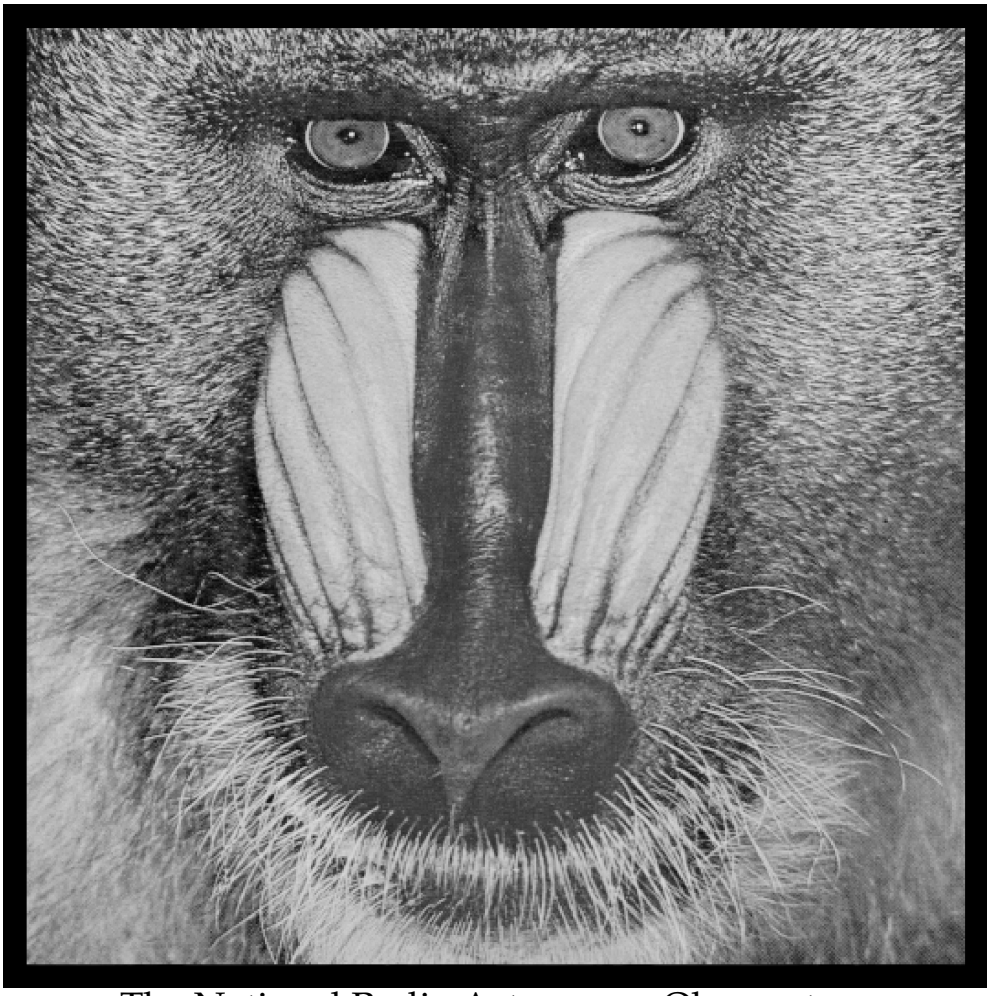
AIPS COOKBOOK

31-December-2025 and earlier



The National Radio Astronomy Observatory

Edgemont Road
Charlottesville, VA 22903–2475
A facility of the National Science Foundation
operated under cooperative agreement by
Associated Universities, Inc.

Acknowledgments

The *CookBook* cover design is by Pat Smiley of the NRAO Graphic Arts Department. It is based on a design suggested by John Bally of Bell Labs.

The image on the title page was converted from the *AIPS* television-like display to Encapsulated PostScript by the *AIPS* task TVCPS. It was then included in this TEX document and plotted on a Hewlett Packard PostScript printer. It is the green portion of the digitized image of a Mandrill which has become a standard in the image-processing field. PostScript is a registered trademark of Adobe Systems Incorporated.

The plot in Chapter 4 was generated with TVCPS in a similar manner. The plots at the ends of Chapters 6, 7, 8, and 9 were generated by various *AIPS* plotting tasks with task LWPLA used to convert the device-independent plot files into PostScript. The data displayed in Chapters 6 and 7 were provided by Bill Cotton for use with the *AIPS* VLAC test suite and by Alan Bridle for use with the *AIPS* DDT test suite, while the data in Chapter 8 were provided by Don Wells for use in testing spectral-line software. The color plots in Chapter 6 are of data provided by Greg Taylor and by Eric Greisen, Kristine Spekkens, and Gustaaf van Moorsel. The editors thank these people for providing their data for our use.

This *CookBook* itself is based on an early users guide written by Alan Bridle. In 1983, it was typeset and edited by Eric Greisen using the TEX program, initially developed by Donald E. Knuth (*The TEXbook*, 1984, Addison-Wesley Publishing Company, Reading, Massachusetts). There were two editions of the *CookBook* in 1983, one in 1985, and one in 1986. The 1990 edition edition was edited by Bill Junor. For recent editions, Eric Greisen has resumed his rôle as editor, while numerous individuals have contributed to the text. In particular, Glen Langston, Andrea Cox, Lorant Sjouwerman, and Minnie Mao have submitted outline guides to VLA continuum, spectral-line, high-frequency, and low frequency data reduction which appear as Appendices A. B, D, and L, respectively. The output of TEX is now converted to PostScript by `dvips` (from Radical Eye Software). The editors are grateful to Knuth for this program and, especially, for his decision to place it in the public domain.

This *CookBook* is now available on the Internet via the World-Wide Web. The current Table of Contents together with a revision history for the full 31DEC25 *CookBook* is available at

`https://www.aips.nrao.edu/cook.html.`

You should review this Web page occasionally to see if chapters important to you have been altered and, if so, why. You may use your favorite Web browser to click on any chapter you wish to receive and the PostScript version will (eventually) appear on your workstation. Cross-linked html and pdf versions of the full *CookBook* are available from this web page as well. The html version is not shipped with *AIPS*, but the pdf version is shipped to all *AIPS* installations.

Contents

1 Introduction 1 – 1

1.1 The NRAO *AIPS* Project — A Summary . . . 1 – 1

1.2 The *CookBook* . . . 1 – 3

1.3 Organization of the *CookBook* . . . 1 – 4

1.3.1 Contents . . . 1 – 4

1.3.2 Minimum match . . . 1 – 5

1.3.3 Fonts and what they signify . . . 1 – 5

1.4 General structure of *AIPS* . . . 1 – 5

2 Starting Up *AIPS* 2 – 1

2.1 Obtaining access to an *AIPS* computer . . . 2 – 1

2.2 Using the workstation . . . 2 – 1

2.2.1 Logging in to the workstation . . . 2 – 1

2.2.2 Control characters on the workstation . . . 2 – 2

2.2.3 Starting the `AIPS` program . . . 2 – 3

2.2.4 Typing commands to the `AIPS` program . . . 2 – 10

2.3 Managing windows . . . 2 – 10

2.3.1 General window management . . . 2 – 10

2.3.2 Managing the *AIPS* TV window called XAS . . . 2 – 12

2.4 Additional recipe . . . 2 – 12

3 Basic *AIPS* Utilities 3 – 1

3.1 Talking to *AIPS* . . . 3 – 1

3.1.1 *POPS* and *AIPS* utilities . . . 3 – 1

3.1.2 Tasks . . . 3 – 1

3.1.3 Verbs . . . 3 – 2

3.1.4 Adverbs . . . 3 – 3

3.2 Your *AIPS* message file . . . 3 – 4

3.3 Your *AIPS* data catalog files . . . 3 – 5

3.3.1 Speedy data file selection . . . 3 – 6

3.3.2 Catalog entry status . . . 3 – 6

3.3.3 Renaming data files . . . 3 – 6

3.3.4 Header listings . . . 3 – 7

3.4 Your AIPS history files . . . 3 – 8
3.5 Saving and restoring inputs . . . 3 – 9
3.6 Monitoring disk space . . . 3 – 11
3.7 Moving and compressing files . . . 3 – 12
3.8 Finding helpful information in AIPS . . . 3 – 12
3.9 Magnetic tapes . . . 3 – 15
3.9.1 Hardware tape mount . . . 3 – 15
3.9.2 Software mounting local tapes . . . 3 – 16
3.9.3 Software mounting REMOTE tapes . . . 3 – 16
3.9.4 Using tapes in AIPS . . . 3 – 16
3.10 AIPS external disk files . . . 3 – 17
3.10.1 Disk text files . . . 3 – 17
3.10.2 RUN files . . . 3 – 18
3.10.3 FITS-disk files . . . 3 – 19
3.10.4 Other binary data disk files . . . 3 – 20
3.11 The array processor . . . 3 – 21
3.12 Additional recipes . . . 3 – 21

4 Calibrating Interferometer Data 4 – 1
4.1 Copying data into AIPS multi-source disk files . . . 4 – 2
4.1.1 Reading EVLA archive files into AIPS . . . 4 – 3
4.1.2 Reading data from FITS files with FITLD . . . 4 – 5
4.2 Record keeping and data management . . . 4 – 6
4.2.1 Calibrating data with multiple FQ entries . . . 4 – 6
4.2.2 Recommended record keeping . . . 4 – 6
4.3 Beginning the calibration . . . 4 – 7
4.3.1 Baseline corrections . . . 4 – 8
4.3.2 Correcting EVLA 3-bit data . . . 4 – 10
4.3.3 Check your data with POSSM . . . 4 – 10
4.3.4 Remove spectral ringing with SPLAT . . . 4 – 11
4.3.5 General considerations in flagging . . . 4 – 11
4.3.6 Correcting delay errors with FRING . . . 4 – 14
4.3.7 Primary flux density calibrators . . . 4 – 15
4.3.8 Calibrating the bandpass shape with BPASS . . . 4 – 16
4.3.9 Calibrating the complex antenna gains with VLACALIB . . . 4 – 19
4.3.9.1 Using calibrator models . . . 4 – 19
4.3.9.2 Calibrating complex gains . . . 4 – 20

4.3.9.3 Flux calibration without calibrator models . . . 4 – 21
4.3.10 Bootstrapping secondary flux-density calibrators . . . 4 – 22
4.3.11 Editing visibility data with EDITA . . . 4 – 23
4.3.12 Applying solutions to the calibration table with CLCAL . . . 4 – 24
4.3.13 Flagging RFI with RFLAG . . . 4 – 28
4.3.14 Restart the calibration . . . 4 – 30
4.3.15 Calibration with the SysPower table . . . 4 – 30
4.3.16 Calibrating polarization . . . 4 – 32
4.3.17 Additional calibrations for EVLA data . . . 4 – 36
4.3.18 Making images from multi-source data with IMAGR . . . 4 – 38
4.3.19 Back up your data with FITTP or FITAB . . . 4 – 39
4.3.20 Creating single-source data files with SPLIT . . . 4 – 39
4.4 Concluding remarks, early science . . . 4 – 40
4.5 Additional recipe . . . 4 – 40

5 Making Images from Interferometer Data 5 – 1
5.1 Preparing *uv* data for imaging . . . 5 – 1
5.1.1 Indexing the data — PRTTP . . . 5 – 1
5.1.2 Loading the data — FITLD and UVLOD . . . 5 – 2
5.1.3 Sorting the data — UVSRT . . . 5 – 3
5.1.4 Combining data sets for oimaging — DBCON and VBGLU . . . 5 – 3
5.2 Basic image making — IMAGR . . . 5 – 4
5.2.1 Making a simple image . . . 5 – 5
5.2.2 Imaging multiple fields and image coordinates . . . 5 – 6
5.2.3 Data weighting . . . 5 – 8
5.2.4 Cell and image size, shifting . . . 5 – 9
5.2.5 Zero-spacing issues . . . 5 – 11
5.3 Deconvolving images . . . 5 – 11
5.3.1 Basic Cleaning with IMAGR . . . 5 – 12
5.3.2 Multiple fields in IMAGR . . . 5 – 14
5.3.3 Clean boxes and the TV in IMAGR . . . 5 – 16
5.3.4 Experimental variations on Clean in IMAGR . . . 5 – 19
5.3.4.1 Clean-component filtering . . . 5 – 19
5.3.4.2 SDI modification of Clean in IMAGR . . . 5 – 20
5.3.4.3 Multi-scale modification of Clean in IMAGR . . . 5 – 20
5.3.4.4 Spectral-index corrections . . . 5 – 21
5.3.5 Data correction options in IMAGR . . . 5 – 21

5.3.5.1 Frequency-dependent primary-beam corrections . . . 5 – 21
5.3.5.2 Frequency-dependent correction for average spectral index . . . 5 – 22
5.3.5.3 Error in the assumed central frequency . . . 5 – 22
5.3.5.4 Array mis-orientation effects . . . 5 – 22
5.3.5.5 Non-coplanar effects . . . 5 – 22
5.3.5.6 Units mismatch of residuals and Clean components . . . 5 – 22
5.3.6 Manipulating Clean components . . . 5 – 23
5.3.7 Image-plane deconvolution methods . . . 5 – 24
5.4 Self-calibration . . . 5 – 26
5.4.1 Self-calibration sequence and SCMAP or SCIMG . . . 5 – 26
5.4.2 Self-calibration with CALIB . . . 5 – 28
5.4.3 Considerations in setting CALIB inputs . . . 5 – 30
5.4.4 Evaluating the quality of the imaging . . . 5 – 32
5.4.5 Experimental extension of multi-field self-calibration . . . 5 – 32
5.5 More editing of *uv* data . . . 5 – 33
5.5.1 General remarks on, and tools for, editing . . . 5 – 33
5.5.2 Baseline-based *uv*-data editing — EDITR . . . 5 – 34
5.6 Imaging with OBIT, Complex imaging . . . 5 – 40
5.7 Additional recipes . . . 5 –40

6 Displaying Your Data 6 – 1
6.1 Getting data into your *AIPS* catalog . . . 6 – 1
6.1.1 IMLOD and FITLD from FITS-disk . . . 6 – 1
6.1.2 Image coordinates . . . 6 – 2
6.2 Printer displays of your data . . . 6 – 3
6.2.1 Printing your visibility data . . . 6 – 3
6.2.2 Printing your image data . . . 6 – 4
6.2.3 Printing your table data . . . 6 – 4
6.2.4 Printing miscellaneous information . . . 6 – 5
6.3 Plotting your data . . . 6 – 5
6.3.1 Plotting your visibility data . . . 6 – 8
6.3.2 Plotting your image data . . . 6 – 9
6.3.2.1 Contour and grey-scale plots . . . 6 – 9
6.3.2.2 Row tracing plots . . . 6 – 13
6.3.2.3 Miscellaneous image plots . . . 6 – 15
6.3.3 Plotting your table data . . . 6 – 16
6.3.4 Plotting miscellaneous information . . . 6 – 17

6.4 Interactive TV displays of your data . . . 6 – 18
6.4.1 Loading an image to the TV . . . 6 – 18
6.4.2 Manipulating the TV display . . . 6 – 19
6.4.3 Intensity and color transfer functions . . . 6 – 20
6.4.4 Setting parameters with the TV . . . 6 – 21
6.4.5 Reading image values from the TV . . . 6 – 22
6.4.6 Labeling images on the TV . . . 6 – 23
6.4.7 Comparing images on the TV . . . 6 – 24
6.4.8 Slice files and the TV display . . . 6 – 26
6.4.9 Other functions using the TV . . . 6 – 27
6.4.10 Capturing the TV . . . 6 – 28
6.5 Graphics displays of your data . . . 6 – 29
6.5.1 Plotting data and setting values with the graphics display . . . 6 – 29
6.5.2 Slice files and the graphics display . . . 6 – 30
6.5.3 Data analysis with the graphics display . . . 6 – 30
6.6 Examples of color plotting . . . 6 – 30

7 Analyzing Images 7 – 1
7.1 Combining two images (COMB) . . . 7 – 1
7.1.1 Subtracting a continuum image from an image cube . . . 7 – 1
7.1.2 Polarized intensity and position angle images . . . 7 – 1
7.1.3 Other image combination options . . . 7 – 2
7.1.4 Considerations in image combination . . . 7 – 3
7.2 Combining more than two images (CMPLX, SUMIM, SPIXR, STACK) . . . 7 – 3
7.3 Image statistics and flux integration . . . 7 – 4
7.4 Blanking of images . . . 7 – 5
7.5 Fitting of images . . . 7 – 6
7.5.1 Centroid fits (IMCENTER) and parabolic fit to maximum (MAXFIT) . . . 7 – 7
7.5.2 Two-dimensional Gaussian fitting (IMFIT, JMFIT) . . . 7 – 7
7.5.3 Source recognition and fitting (SAD, TVSAD) . . . 7 – 8
7.5.4 Gaussian fits to slices (SLFIT) . . . 7 – 9
7.5.5 Spectral parameter fitting (TVSPC, XGAUS, ZEMAN, AGAUS, ZAMAN, RMFIT, FARS) . . . 10
7.6 Image analysis . . . 7 – 12
7.6.1 Geometric conversions . . . 7 – 12
7.6.2 Mathematical operations on a single image . . . 7 – 13
7.6.3 Primary beam correction . . . 7 – 15
7.6.4 Changing the resolution of an image . . . 7 – 15

7.6.5 Filtering . . . 7 – 16
7.7 Modeling in the image and uv planes . . . 7 – 17
7.7.1 Analysis in the uv plane . . . 7 – 17
7.8 Additional recipes . . . 7 – 18

8 Spectral-Line Software 8 – 1
8.1 Data preparation and assessment . . . 8 – 1
8.2 Editing and self-calibration . . . 8 – 4
8.3 Continuum subtraction . . . 8 – 5
8.4 Imaging . . . 8 – 6
8.5 Display and manipulation of data cubes . . . 8 – 7
8.5.1 Building and dismantling data cubes . . . 8 – 7
8.5.2 Transposing the cube . . . 8 – 9
8.5.3 Modifying the image header . . . 8 – 9
8.5.4 Displaying the cube . . . 8 – 10
8.6 Analysis . . . 8 – 14
8.7 Additional recipes . . . 8 – 17

9 Reducing VLBI Data in AIPS 9 – 1
9.1 VLBI data calibration recipe . . . 9 – 2
9.2 Pipeline for the VLBA . . . 9 – 5
9.3 Loading, fixing and inspecting data . . . 9 – 7
9.3.1 Loading data from the VLBA correlator . . . 9 – 7
9.3.1.1 Running FITLD . . . 9 – 7
9.3.1.2 Calibration transfer . . . 9 – 11
9.3.1.3 Repairing VLBA data after FITLD . . . 9 – 12
9.3.1.4 Sorting and indexing VLBA correlator data . . . 9 – 12
9.3.1.5 Subarraying VLBA correlator data . . . 9 – 13
9.3.1.6 Indexing VLBA correlator data . . . 9 – 14
9.3.1.7 Concatenating VLBA correlator data . . . 9 – 14
9.3.1.8 Labeling VLBA correlator polarization data . . . 9 – 15
9.3.1.9 Ionospheric corrections . . . 9 – 16
9.3.1.10 Corrections to the Earth Orientation Parameters . . . 9 – 17
9.3.1.11 Preparing the OB table for SVLBI data . . . 9 – 18
9.3.1.12 Loading the time corrections file for SVLBI data . . . 9 – 18
9.4 Tools for data examination . . . 9 – 19
9.4.1 Textual displays . . . 9 – 19

9.4.2 Spectral displays: POSSM . . . 9 – 20
9.4.3 Time displays: VPLOT, CLPLT, and CAPLT . . . 9 – 21
9.4.4 EDITR . . . 9 – 22
9.4.5 SNPLT . . . 9 – 23
9.4.6 COHER . . . 9 – 23
9.4.7 FRPLT . . . 9 – 43
9.5 Calibration strategy . . . 9 – 25
9.5.1 Incremental calibration philosophy . . . 9 – 26
9.5.1.1 Smoothing and applying corrections in SN and CL tables . . . 9 – 26
9.5.1.2 Running CLCAL for phase referencing observations . . . 9 – 28
9.5.2 Processing observing log and calibration information . . . 9 – 29
9.5.2.1 Automatic formatting of VLBA and VLBA-like log files . . . 9 – 29
9.5.2.2 Manual formatting of log files . . . 9 – 30
9.5.2.3 Loading calibration log information . . . 9 – 30
9.5.2.4 Generating VLA amplitude calibration with VLAMP . . . 9 – 31
9.5.3 Data editing . . . 9 – 32
9.5.4 Amplitude and instrumental delay calibration . . . 9 – 33
9.5.4.1 Parallactic angle correction . . . 9 – 33
9.5.4.2 Digital sampler bias corrections for VLBA correlator data . . . 9 – 34
9.5.4.3 Instrumental phase corrections . . . 9 – 34
9.5.4.4 "Manual" instrumental phase corrections . . . 9 – 36
9.5.4.5 Bandpass calibration . . . 9 – 38
9.5.4.6 Continuum amplitude calibration . . . 9 – 40
9.5.5 Spectral-line Doppler correction . . . 9 – 41
9.5.6 Spectral-line amplitude calibration . . . 9 – 42
9.5.7 Phase calibration . . . 9 – 44
9.5.7.1 Special considerations: SVLBI . . . 9 – 44
9.5.7.2 Special considerations: spectral-line . . . 9 – 45
9.5.7.3 Special considerations: polarization . . . 9 – 45
9.5.7.4 Special considerations: phase-referencing . . . 9 – 46
9.5.7.5 Correcting for atmospheric delays . . . 9 – 46
9.5.7.6 Finding multi-band delays . . . 9 – 47
9.5.7.7 Antenna-based fringe-fitting . . . 9 – 48
9.5.7.8 Baseline-based fringe-fitting . . . 9 – 55
9.5.7.9 SVLBI-specific techniques . . . 9 – 57
9.5.7.10 Spectral-line fringe-fitting . . . 9 – 57

9.5.7.11 Polarization-specific fringe-fitting . . . 9 – 59
9.5.7.12 R-L delay calibration . . . 9 – 60
9.5.7.13 Feed D-term calibration . . . 9 – 61
9.5.8 Complex Bandpass . . . 9 – 62
9.5.9 Baseline-based errors . . . 9 – 63
9.6 After initial calibration . . . 9 – 63
9.6.1 Applying calibration . . . 9 – 63
9.6.2 Time averaging . . . 9 – 64
9.6.3 Verifying calibration . . . 9 – 65
9.7 Self-calibration, imaging, and model-fitting . . . 9 – 67
9.7.1 CALIB . . . 9 – 67
9.7.2 IMAGR, SCIMAG, and SCMAP . . . 9 – 69
9.7.3 Non-conventional methods of imaging . . . 9 – 69
9.8 Summary of VLBI calibration tables . . . 9 – 71
9.9 Additional recipes . . . 9 – 72

10 Single-Dish Data in AIPS 10 – 1
10.1 AIPS format for single-dish data . . . 10 – 1
10.1.1 On-the-fly data from the 12m . . . 10 – 2
10.1.1.1 Listing OTF input files . . . 10 – 2
10.1.1.2 Reading spectral-line OTF files into AIPS . . . 10 – 3
10.1.1.3 Reading continuum OTF files into AIPS . . . 10 – 4
10.1.2 Other input data formats . . . 10 – 4
10.2 Single-dish data in the "uv" domain . . . 10 – 5
10.2.1 Using PRTSD, UVPLT, and POSSM to look at your data . . . 10 – 5
10.2.2 Using UVFLG, SPFLG, and EDITR to edit your data . . . 10 – 7
10.2.3 Using CSCOR and SDCAL to calibrate your data . . . 10 – 11
10.2.4 Using SDLSF and SDVEL to correct your spectral-line data . . . 10 – 12
10.2.5 Using SDMOD and BSMOD to model your data . . . 10 – 13
10.3 Imaging single-dish data in AIPS . . . 10 – 14
10.3.1 Normal single-dish imaging . . . 10 – 14
10.3.2 Beam-switched continuum imaging . . . 10 – 16
10.4 Analysis and display of single-dish data . . . 10 – 20
10.4.1 Spectral baseline removal . . . 10 – 20
10.4.2 Using WTSUM and BSAVG to combine images . . . 10 – 21
10.4.3 Spectral moment analysis . . . 10 – 22
10.4.4 Source modeling and fitting . . . 10 – 22

10.4.5 Image displays . . . 10 – 23
10.4.6 Backing up your data . . . 10 – 24
10.5 Combining single-dish and interferometer data . . . 10 – 24
10.6 Additional recipes . . . 10 – 25

11 Exiting from, and Solving Problems in, *AIPS* 11 – 1
11.1 Helping the *AIPS* programmers . . . 11 – 1
11.2 Exiting from *AIPS* . . . 11 – 1
11.2.1 Backups . . . 11 – 2
11.2.2 Deleting your data . . . 11 – 2
11.2.3 Exiting . . . 11 – 3
11.3 Solving problems in using *AIPS* . . . 11 – 3
11.3.1 "Terminal" problems . . . 11 – 3
11.3.2 Disk data problems . . . 11 – 4
11.3.3 Printer problems . . . 11 – 5
11.3.4 Tape problems . . . 11 – 6

12 *AIPS* for the More Sophisticated User 12 – 1
12.1 *AIPS* conventions . . . 12 – 1
12.1.1 AIPS shortcuts . . . 12 – 1
12.1.2 Data-file names and formats . . . 12 – 2
12.2 Process control features of *AIPS* . . . 12 – 3
12.2.1 RUN files . . . 12 – 3
12.2.2 More about GO . . . 12 – 4
12.2.3 Batch jobs . . . 12 – 5
12.3 AIPS language . . . 12 – 6
12.3.1 Using *POPS* outside of procedures . . . 12 – 8
12.3.2 Procedures . . . 12 – 9
12.3.3 Writing your own programs with *POPS* . . . 12 – 12
12.3.3.1 Special facilities for use in procedures . . . 12 – 13
12.4 Remote use of *AIPS* . . . 12 – 14
12.4.1 Connections via X-Windows . . . 12 – 14
12.4.2 Connections to a terminal . . . 12 – 15
12.4.3 File transfer connections . . . 12 – 16
12.5 Moving data to a new computer . . . 12 – 17
12.6 Adding your own tasks to *AIPS* . . . 12 – 18
12.6.1 Initial choices to make . . . 12 – 18

12.6.2 Getting started . . . 12 – 19
12.6.3 Initial check of code and procedures . . . 12 – 19
12.6.4 Modifying an *AIPS* task . . . 12 – 20
12.6.5 Modifying an *AIPS* template task. . . . 12 – 21
12.6.6 Further remarks . . . 12 – 22

13 Current *AIPS* Software 13 – 1
13.1 ADVERB . . . 13 – 1
13.2 ANALYSIS . . . 13 – 10
13.3 AP . . . 13 – 13
13.4 ASTROMET . . . 13 – 13
13.5 BATCH . . . 13 – 14
13.6 CALIBRAT . . . 13 – 15
13.7 CATALOG . . . 13 – 21
13.8 COORDINA . . . 13 – 23
13.9 EDITING . . . 13 – 24
13.10 EXT-APPL . . . 13 – 25
13.11 FITS . . . 13 – 26
13.12 GENERAL . . . 13 – 26
13.13 HARDCOPY . . . 13 – 29
13.14 IMAGE-UT . . . 13 – 29
13.15 IMAGE . . . 13 – 30
13.16 IMAGING . . . 13 – 30
13.17 INFORMAT . . . 13 – 34
13.18 INTERACT . . . 13 – 34
13.19 MODELING . . . 13 – 36
13.20 OBSOLETE . . . 13 – 37
13.21 ONED . . . 13 – 38
13.22 OOP . . . 13 – 39
13.23 OPTICAL . . . 13 – 40
13.24 PARAFORM . . . 13 – 40
13.25 PLOT . . . 13 – 40
13.26 POLARIZA . . . 13 – 44
13.27 POPS . . . 13 – 45
13.28 PROCEDUR . . . 13 – 48
13.29 PSEUDOVE . . . 13 – 49
13.30 RUN . . . 13 – 50

13.31 SINGLEDI . . . 13 – 51
13.32 SPECTRAL . . . 13 – 51
13.33 TABLE . . . 13 – 53
13.34 TAPE . . . 13 – 54
13.35 TASK . . . 13 – 55
13.36 TV . . . 13 – 66
13.37 TV-APPL . . . 13 – 69
13.38 UTILITY . . . 13 – 70
13.39 UV . . . 13 – 71
13.40 VERB . . . 13 – 78
13.41 VLA . . . 13 – 84
13.42 VLBI . . . 13 – 85
13.43 Additional recipes . . . 13 – 87

A Easy Continuum UV-Data Calibration and Imaging A – 1
A.1 VLARUN . . . A – 1
A.2 Basic calibration . . . A – 2
A.3 Polarization calibration . . . A – 5
A.4 Backup and imaging . . . A – 5
A.5 Additional recipes . . . A – 6

B A Step-by-Step Guide to Spectral-Line Data Analysis in *AIPS* B – 1
B.1 Editing and calibrating spectral-line data . . . B – 1
B.1.1 Loading the data . . . B – 1
B.1.2 Inspecting and editing the data . . . B – 2
B.1.3 Calibrating the data . . . B – 3
B.2 Making and Cleaning image cubes . . . B – 5
B.3 Moment analysis and rotation curve of galaxies . . . B – 6
B.4 Multi-frequency observations . . . B – 7
B.4.1 General frequency information . . . B – 7
B.4.2 Multi-frequency *uv* files . . . B – 7
B.5 Additional recipes . . . B – 8

C A Step-by-Step Recipe for VLBA Data Calibration in *AIPS* C – 1
C.1 Quick Guide . . . C – 1
C.2 Table Philosophy . . . C – 2
C.3 Data set assumed in this Appendix . . . C – 2
C.4 VLBA Utilities . . . C – 3

C.5 VLBA Pipeline . . . C – 4
C.6 Data Loading and Inspection . . . C – 6
C.7 Amplitude and Delay Calibration . . . C – 7
C.8 Rate and Phase Calibration . . . C – 10
C.9 Final Calibration Steps . . . C – 11
C.10 Incorporating non-VLBA antennas . . . C – 13
C.10.1 Loading $T_{\rm sys}$ and Gain Curves . . . C – 12
C.10.2 Pointing Flags . . . C – 13
C.11 Pre-EVLA VLA data . . . C – 12
C.11.1 Single VLA Antenna . . . C – 13
C.11.2 Phased VLA . . . C – 13
C.12 Summary for non-VLBA antennas . . . C – 14
C.13 Some Useful References . . . C – 15
C.14 Additional recipes . . . C – 15

D Hints for Reducing High-Frequency VLA Data in AIPS D – 1
D.1 Complications at higher frequencies . . . D – 1
D.2 Additional recipes . . . D – 7
D.1 Additional recipes . . . D – 7
O.1 The Historical VLA . . . O – 1
O.1.1 Reading from VLA archive files using FILLM . . . O – 2
O.1.2 Reading old spectral-line data . . . O – 6
O.1.3 Applying nominal sensitivities to historic VLA data . . . O – 6
O.1.4 Calibrating historic VLA data . . . O – 7
O.1.5 Editing with LISTR and UVFLG . . . O – 7
O.1.6 Editing with TVFLG . . . O – 9
O.1.7 Spectral-line calibration . . . O – 15
O.1.7.1 Spectral-line aspects of SETJY . . . O – 16
O.1.7.2 Editing the spectral data . . . O – 16
O.1.7.3 Calibrating the spectral data . . . O – 17
O.1.8 Solar data calibration for the historic VLA . . . O – 18
O.1.8.1 Reading solar data from the VLA archive . . . O – 18
O.1.8.2 Using SNPLT and LISTR to assess the nominal sensitivities . . . O – 19
O.1.8.3 Using SOLCL to apply the system-temperature correction . . . O – 20
O.2 Old VLBI format data . . . O – 20
O.2.1 Loading data from a MkIII/MkIV correlator . . . O – 20
O.2.1.1 Running MK3IN . . . O – 20

O.2.1.2 Sorting MkIII/IV data . . . O – 22
O.2.1.3 Concatenating MkIII/IV data . . . O – 22
O.2.1.4 Labeling correlator polarization data . . . O – 23
O.2.1.5 Merging MkIII/IV data . . . O – 23
O.2.1.6 Correcting MkIII/IV sideband phase offsets . . . O – 24
O.2.1.7 Indexing MkIII/IV data . . . O – 24

L Special Considerations for EVLA P-band Data Processing in *AIPS* L – 1
L.1 P-band calibration and editing in *AIPS* . . . L – 1
L.2 P-band imaging in *AIPS* . . . L – 12
L.3 Additional recipes . . . L – 14

F File Sizes F – 1
F.1 Visibility (*uv*) data sets . . . F – 1
F.1.1 *uv* database sizes . . . F – 1
F.1.2 Compressed format for *uv* data . . . F – 2
F.2 Image files . . . F – 3
F.3 Extension files . . . F – 3
F.4 Storing data on tape . . . F – 4
F.4.1 DAT and Exabyte tapes . . . F – 5
F.5 Very large data sets . . . F – 6
F.6 Additional recipes . . . F – 6

V VLA Maintenance Tasks in *AIPS* V – 1
V.1 Real-time calibration tables: TLCAL . . . V – 1
V.2 Correcting antenna positions: LOCIT . . . V – 2
V.3 Improving antenna pointing: PEEK . . . V – 5
V.4 Improving antenna surfaces: holography . . . V – 6
V.5 Antenna polarization: PDPLT . . . V – 11

Z System-Dependent *AIPS* Tips Z – 1
Z.1 NRAO workstations — general information . . . Z – 1
Z.1.1 The "midnight" jobs . . . Z – 1
Z.1.2 Generating color hard copy . . . Z – 1
Z.1.2.1 Color printers . . . Z – 1
Z.1.2.2 Software to copy your screen . . . Z – 2
Z.1.3 Gripe, gripe, gripe, ... Z – 3
Z.1.4 Solving problems at the NRAO . . . Z – 3

Z.1.4.1 Booting the workstations . . . . . Z – 3
Z.1.4.2 Printout fails to appear . . . . . Z – 3
Z.1.4.3 Stopping excess printout . . . . . Z – 4
Z.1.4.4 CTRL Z problems . . . . . Z – 4
Z.1.4.5 "File system is full" message . . . . . Z – 5
Z.1.4.6 Tapes won't mount . . . . . Z – 5
Z.1.4.7 I can't use my data disk! . . . . . Z – 5
Z.2 AIPS at the NRAO AOC in Socorro . . . . . Z – 5
Z.2.1 Reserving public-use workstations at the AOC . . . . . Z – 5
Z.2.2 Using Linux workstations at the AOC — introduction . . . . . Z – 6
Z.2.3 Starting AIPS . . . . . Z – 7
Z.2.3.1 Starting AIPS on another machine . . . . . Z – 7
Z.2.3.2 On-line FILLM . . . . . Z – 7
Z.3 AIPS at the NRAO in Charlottesville and Green Bank . . . . . Z – 8
Z.4 Additional recipes . . . . . Z – 7

G Glossary G – 1

I INDEX I – 1

List of Figures

1.1 Basic *AIPS* flow diagram . . . 1 – 6
4.1 3C48 at X-band . . . 4 – 21
4.2 EDITA display . . . 4 – 25
4.3 RFLAG display . . . 4 – 29
4.4 Example antenna D term spectrum . . . 4 – 35
4.5 Orion hot core at K band . . . 4 – 42
5.1 Affect of ROBUST parameter on beams. . . . 5 – 10
6.1 UVPLT displays . . . 6 – 7
6.2 Contour and grey-scale plots of an image . . . 6 – 12
6.3 GREYS with profiles . . . 6 – 13
6.4 Row plots of an image . . . 6 – 14
6.5 Statistical plots of an image . . . 6 – 15
6.6 Slice and table plots . . . 6 – 16
6.7 OFMPL step wedge display . . . 6 – 21
6.8 KNTR with pseudo-coloring, contours and polarization vectors in bright and dark lines . . . 6 – 31
6.9 KNTR shows true-color image . . . 6 – 32
6.10 PCNTR with polarization position angle shown with color . . . 6 – 33
6.11 PCNTR with contours colored by velocity . . . 6 – 34
6.12 KNTR with contours colored with RGBLEVS . . . 6 – 35
6.13 SNPLT with IFs/polarizations colored . . . 6 – 36
7.1 Geometric and other functions on an image. . . . 7 – 14
8.1 KNTR contours of spectral channels . . . 8 – 11
8.2 TVCUBE display of transposed cube. . . . 8 – 12
8.3 PLCUB display of spectra from a cube . . . 8 – 13
8.4 Images of line-cube moments. . . . 8 – 14
9.1 POSSM and VPLOT displays of uncalibrated VLBI data . . . 9 – 50
9.2 Spectrum before and after calibration . . . 9 – 54
9.3 An example of a fringe rate image . . . 9 – 70
10.1 UVPLT displays of single-dish data. . . . 10 – 6
10.2 POSSM display of single-dish data. . . . 10 – 7
10.3 SPFLG display. . . . 10 – 10
10.4 Convolving function Fourier transforms. . . . 10 – 16
10.5 Images at various beam throw rotations. . . . 10 – 19
10.6 Images at various beam throw corrections. . . . 10 – 20
A.1 Visibility data before and after calibration . . . A – 4

D.1 VPLOT of "bad" phases . . . . . D – 2
D.2 VPLOT of "okay" phases . . . . . D – 3
O.1 TVFLG display . . . . . O – 12
V.1 Example LOCIT plot . . . . . V – 4
V.2 Example holography raster . . . . . V – 7
V.3 Example HOLOG images . . . . . V – 9
V.4 Example PANEL images . . . . . V – 10
V.5 Example PBEAM plots . . . . . V – 11
V.6 Example PDPLT plot . . . . . V – 12

List of Recipes

2.4.1 Banana daiquiri . . . . . 2 – 12
3.12.1 Columbian fresh banana cake with sea foam frosting . . . . . 3 – 21
3.12.2 Banana sweet potato puff casserole . . . . . 3 – 22
3.12.3 Banana poundcake . . . . . 3 – 22
3.12.4 Banana July cocktail . . . . . 3 – 22
4.5.1 Chewy banana split dessert . . . . . 4 – 40
4.5.2 Mexican bananas . . . . . 4 – 40
4.5.3 Sopa de Plátano . . . . . 4 – 41
4.5.4 Almond fudge banana cake . . . . . 4 – 41
4.5.5 Delightful banana cheesecake . . . . . 4 – 41
5.7.1 Cranberry banana bread . . . . . 5 – 40
7.8.1 Golden mousse . . . . . 7 – 18
7.8.2 Coriander banana nut bread . . . . . 7 – 18
8.7.1 Hot banana soufflé . . . . . 8 – 17
8.7.2 Banana crunch cake . . . . . 8 – 18
8.7.3 Panecillos de Plátano . . . . . 8 – 18
8.7.4 Banana-Rhubarb Crisp . . . . . 8 – 18
9.9.1 Bananes rôties . . . . . 9 – 72
9.9.2 Banana curried chicken . . . . . 9 – 72
10.6.1 Chicken salad with banana mayonnaise and grapes . . . . . 10 – 25
10.6.2 Curried bananas . . . . . 10 – 25
10.6.3 Hawaiian banana cream pie . . . . . 10 – 26
10.6.4 Banana Dream Pizza . . . . . 10 – 26
13.43.1 Sautéd sole tobago with bananas, pecans and lime . . . . . 13 – 87
13.43.2 Banana Bombay salad . . . . . 13 – 87
13.43.3 Banana mandarin cheese pie . . . . . 13 – 88
13.43.4 Cream of banana soup . . . . . 13 – 88
13.43.5 Banana relish . . . . . 13 – 88
13.43.6 Banana coffeelate . . . . . 13 – 88
13.43.7 Banana pick-me-up . . . . . 13 – 88
A.5.1 Banana colada . . . . . A – 6
A.5.2 Breaded chicken and bananas . . . . . A – 6
A.5.3 Banana-pineapple bread . . . . . A – 6
B.5.1 Banana breeze pie . . . . . B – 8
B.5.2 Banana cutlets . . . . . B – 8

C.14.1 Orange baked bananas . . . . . C – 15
C.14.2 Mexican chicken vegetable soup with bananas . . . . . C – 15
C.14.3 Orange gingered bananas . . . . . C – 16
C.14.4 Frozen Push-Ups . . . . . C – 16
C.14.5 Banana mallow pie . . . . . C – 16
D.2.1 Chocolate chip banana bread . . . . . D – 7
D.2.2 Banana bran muffins . . . . . D – 8
D.2.3 Banana-pineapple rum bread . . . . . D – 8
D.2.4 Dulce Zacatecaño . . . . . D – 8
D.2.5 Virginia's instant banana pie . . . . . D – 8
L.3.1 Banana storage . . . . . L – 14
F.6.1 Banana stuffing . . . . . F – 6
F.6.2 Banana nut bread . . . . . F – 6
Z.4.1 Going bananas with bananas . . . . . Z – 7
Z.4.2 Banana-chocolate tea bread . . . . . Z – 8
Z.4.3 Banana caramel pie . . . . . Z – 8
Z.4.4 Curried shrimp . . . . . Z – 8

1 Introduction

1.1 The NRAO $\mathcal{AIPS}$ Project — A Summary

The NRAO Astronomical Image Processing System ($\mathcal{AIPS}$) is a software package for interactive (and, optionally, batch) calibration and editing of radio interferometric data and for the calibration, construction, display and analysis of astronomical images made from those data using Fourier synthesis methods. Design and development of the package began in Charlottesville, Virginia in 1978. It presently consists of over 1,785,000 lines of code, 234,000 lines of on-line documentation, and 497,000 lines of other documentation.[1] It contains over 635 distinct applications "tasks," representing very approximately 100+ man-years of effort since 1978. The $\mathcal{AIPS}$ group, now solely in Socorro, has one full-time scientist/programmer, and a few other scientific staff with some low-level responsibility to the $\mathcal{AIPS}$ effort. The "group" is responsible for the code design and maintenance, for documentation aimed at users and programmers, and for exporting the code to hundreds [2] of non-NRAO sites that have requested copies of $\mathcal{AIPS}$. It currently offers $\mathcal{AIPS}$ installation kits for a variety of UNIX systems (Linux and Mac OS/X), with updates available annually.[3]

In 1983, when $\mathcal{AIPS}$ was selected as the primary data reduction package for the Very Long Baseline Array (VLBA), the scope of the $\mathcal{AIPS}$ effort was expanded to embrace all stages of radio interferometric calibration, both continuum and spectral line. The $\mathcal{AIPS}$ package contains a full suite of calibration and editing functions for both VLA and VLBI data, including interactive and batch methods for editing visibility data. For VLBI, it reads data in MkII, MkIII and VLBA (FITS IDI) formats, performs global fringe-fitting by two alternative methods, offers special phase-referencing and polarization calibration, and performs geometric corrections, in addition to the standard calibrations done for connected-element interferometers. The calibration methods for both domains encourage the use of realistic models for the calibration sources and iterated models using self-calibration for the program sources.

$\mathcal{AIPS}$ has been the principal tool for display and analysis of both two- and three-dimensional radio images (*i.e.*, continuum "maps" and spectral-line "cubes") from the NRAO's Very Large Array (VLA) since early in 1981. It has also provided the main route for self-calibration and imaging of VLA continuum and spectral-line data. It contains facilities for display and editing of data in the aperture, or u-v, plane; for image construction by Fourier inversion; for deconvolution of the point source response by Clean and by maximum entropy methods; for image combination, filtering, and parameter estimation; and for a wide variety of TV and graphical displays. It records all user-generated operations and parameters that affect the quality of the derived images, as "history" files that are appended to the data sets and can be exported with them from $\mathcal{AIPS}$ in the IAU-standard FITS (Flexible Image Transport System) format. $\mathcal{AIPS}$ implements a simple command language which is used to run "tasks" (*i.e.*, separate programs) and to interact with text, graphics and image displays. A batch mode is also available. The package contains nearly 11.4 Mbytes of "help" text that provides on-line documentation for users. There is also a suite of printed manuals (available on-line) for users and for programmers wishing to code their own applications "tasks" within $\mathcal{AIPS}$.

An important aspect of $\mathcal{AIPS}$ is its portability. It has been designed to run, with minimal modifications, in a wide variety of computing environments. This has been accomplished by the use of generic FORTRAN wherever possible and by the isolation of system-dependent code into well-defined groups of routines. $\mathcal{AIPS}$ tries to present as nearly the same interface to the user as possible when implemented in different

[1] Counted on 15-December-2025 and omitting the GNU copyrights, PostScript and PDF files, and obsolete areas.

[2] The `15JAN96` release alone was shipped to 225 sites and installed on well over 800 computers. The `15APR99` release was shipped to 344 non-NRAO sites. In 2009, more than 2400 different IP addresses installed or updated copies of $\mathcal{AIPS}$. In 2019, this number had dropped to 1155.

[3] The `TST` version, currently `31DEC24`, will be available continuously. It remains under development and sites may update their copies at will. Binary versions are available for Linux (32- and 64-bit) and Max OS/X (Intel and ARM) and text versions for these and other operating systems.

computer architectures and under different operating systems. The NRAO has sought this level of hardware and operating system independence in AIPS for two main reasons. The first is to ensure a growth path by allowing AIPS to exploit computer manufacturers' advances in hardware and in compiler technology relatively quickly, without major re-coding. AIPS was developed in ModComp and Vax/VMS environments with Floating Point Systems array processors, but was migrated to vector pipeline machines in 1985. Its portability allowed it to take prompt advantage of the new generation of vector and vector/parallel optimizing compilers offered in 1986 by manufacturers such as Convex and Alliant. It was extended in simple ways in 1992 to take full advantage of the current, highly-networked workstation environment. The second is to service the needs of NRAO users in their home institutes, where available hardware and operating systems may differ substantially from NRAO's. By doing this, the NRAO supports data reduction at its users' own locations, where they can work without the deadlines and other constraints implicit in a brief visit to an NRAO telescope site. The exportability of AIPS is now well exploited in the astronomical community; the package is known to have been installed at some time on a large number of different computers, and is currently in active use for astronomical research at way more than 140[4] sites worldwide. AIPS has been run on Cray and Fujitsu supercomputers, on Convex and Alliant "mini-supercomputers," on the full variety of Vaxen and MicroVaxen, and on a wide range of UNIX workstations and laptops including Apollo, Data General, Hewlett Packard, IBM, MassComp, Nord, Silicon Graphics, Stellar and SUN products. It is available for use on personal computers under the public-domain Linux operating system and, since 2003, On MacIntosh OS/X computers. In late 1990[1], the total computer power used for AIPS was the equivalent of about 6.5 Cray X-MP processors running full-time.

Similarly, a wide range of digital TV devices and printer/plotters has been supported through AIPS's "virtual device interfaces". Support for such peripherals is contained in well-isolated subroutines coded and distributed by the AIPS group or by AIPS users elsewhere. Television-like interactive display is now provided directly on workstations using an AIPS television emulator and X-Windows. Hardware TV devices are no longer common, but those used at AIPS sites have included IIS Model 70 and 75, IVAS, AED, Apollo, Aydin, Comtal, DeAnza, Graphica, Graphics Strategies, Grinnell, Image Analytics, Jupiter, Lexidata, Ramtek, RCI Trapix, Sigma ARGS, Vaxstation/GPX and Vicom. Printer/plotters include Versatec, QMS/Talaris, Apple, Benson, CalComp, Canon, Digital Equipment, Facom, Hewlett-Packard, Imagen, C.Itoh, Printek, Printronix and Zeta products. Generic and color encapsulated PostScript is produced by AIPS for a wide variety of printers. The standard interactive graphics interface in AIPS is the Tektronix 4012, now normally emulated on workstations using an AIPS program and X-Windows.

The principal users of AIPS are VLA, VLBA, and VLBI Network observers. A survey of AIPS sites carried out in late 1990[1] showed that 61% of all AIPS data processing worldwide was devoted to VLA data reduction. Outside the NRAO, AIPS is extensively used for other astronomical imaging applications, however. 56% of all AIPS processing done outside the U.S. involved data from instruments other than the VLA. The astronomical applications of AIPS that do not involve radio interferometry include the display and analysis of line and continuum data from large single-dish radio surveys, and the processing of image data at infrared, visible, ultraviolet and X-ray wavelengths. About 7% of all AIPS processing involved astronomical data at these shorter wavelengths, with 7% of the computers in the survey using AIPS more for such work than for radio and *another* 7% of the computers using AIPS exclusively for non-radio work.

Some AIPS use occurs outside observational astronomy, *e.g.*, in visualization of numerical simulations of fluid processes, and in medical imaging. The distinctive features of AIPS that have attracted users from outside the community of radio interferometrists are its ability to handle many relevant coordinate geometries precisely, its emphasis on display and analysis of the data in complementary Fourier domains, the NRAO's support for exporting the package to different computer architectures, and its extensive documentation.

As well as producing user- and programmer-oriented manuals for AIPS, the group publishes a newsletter semi-annually that is sent to subscribing libraries and is available on the Web. There is also a mechanism whereby users can report software bugs or suggestions to the AIPS programmers and receive written responses from them; this provides a formal route for user feedback to the AIPS programmers and for the

[4] "The 1990 AIPS Site Survey", AIPS Memo No. 70, Alan Bridle and Joanne Nance, April 1991

programmers to document difficult points directly to individual users. Much of the *AIPS* documentation is now available to the "World-Wide Web" so that it may be examined over the Internet (start with "URL" `https://info.aips.nrao.edu/`). The NRAO knows of over 230 *AIPS* "tasks," or programs, that have been coded within the package outside, and not distributed by, the observatory.

The *AIPS* group has developed a package of benchmarking and certification tests that process standard data sets through the dozen most critical stages of interferometric data reduction, and compare the results with those obtained on the NRAO's own computers. The "Y2K" package is used to verify the correctness of the results produced by *AIPS* installations at new user sites or on new types of computer, as well as to obtain comparative timing information for different computer architectures and configurations. It has been extensively used as a benchmarking package to guide computer procurements at the NRAO and elsewhere. Two other packages, "VLAC" and "VLAL", are less widely used to verify the continued correctness of calibration and spectral-line reductions.

In 1992, the NRAO joined a consortium of institutions seeking to replace all of the functionality of *AIPS* using modern coding techniques and languages. The "`aips++`" project, now named "`casa`," is expected to provide the main software platform supporting radio-astronomical data processing sometime in the future. Further development of the original ("Classic") *AIPS* will therefore be limited mostly to calibration of VLBI data, general code maintenance with minor enhancements, and improvements in the user documentation.

Further information on *AIPS* can be obtained by writing by electronic mail to `daip@nrao.edu` or by paper mail to the AIPS Group, National Radio Astronomy Observatory, P. O. Box O, Socorro, NM, 87801-0387, U.S.A.

1.2 The *CookBook*

This *CookBook* is intended to help beginning users of the NRAO *A*stronomical *I*mage *P*rocessing *S*ystem (*AIPS*) by providing a recipe approach to the most basic *AIPS* operations. While it illustrates some aspects of *AIPS*, it does not pretend to be complete. However, it does include detailed instructions for running many important items of *AIPS* software. With these as a model, the user should be able to run other *AIPS* software aided by the `EXPLAIN`, `HELP` and `INPUTS` files and the complete index of software given in Chapter 13 of the *CookBook*. In this edition, some of the chapters have matured into something more like a users' manual, than a beginners' cookbook. These sections provide an overview of a few less basic, but nonetheless interesting, programs which often seem to be forgotten even by experienced *AIPS* users. To assist the beginning and infrequent user, appendices have been added to provide outlines of continuum, spectral-line, and high-frequency calibration procedures, primarily geared to users of the VLA, and a simplified outline of VLBA data reduction. A guide to reduction of data from the historic VLA appears in Appendix O. To assist in finding information in this now large document, an index has been added.

AIPS software is changing and growing continually. This edition of the *CookBook* describes the `31DEC24` (aka "*AIPS* for the Ages (Aged), version 25") release of *AIPS*. Some chapters have information only from earlier releases. When something only applies to fairly recent versions of *AIPS*, a comment to that effect is made. Features remain in later releases even if the particular comment does not say as much. There were many changes in *AIPS* software between the seventh (`15JAN94`) and the sixth (`15JUL90`) edition of the *CookBook*. The chapter on the *AIPS* calibration package for continuum, spectral-line, solar and VLBI data (Chapter 4) was revised in 2018 to describe calibration and editing of the Expanded VLA. Material applying to the old VLA was moved to Appendix O (old). The list of current *AIPS* tasks (Chapter 13) has been updated and reflects the extensive improvement and expansion of *AIPS* software. The chapters on imaging and improving images were merged as were the chapters on interactive and hard-copy displays. These mergers reflect in part the mergers of these operations. The chapter on spectral-line imaging (Chapter 8) has been revised with the assistance of Elias Brinks. Phil Diamond, John Conway, Athol Kemball, and Ketan Desai have rewritten the chapter on VLBI calibration and imaging (Chapter 9).

Appendix Z contains instructions and advice peculiar to the individual *AIPS* sites of the NRAO. This has been revised extensively to reflect the migration of much of the data reduction at NRAO sites away from VAXes and Convex computers to Sun and Linux workstations. The ninth edition has an Index which is current and updates concerning editing, calibration, imaging and single-dish processing in the `15APR98` and later versions of *AIPS*. This edition still contains, essentially unchanged, the helpful glossary of astronomical and computing terms written by Fred Schwab.

Paper copies of recent editions of the *CookBook* are no longer available from NRAO. However, much of the *AIPS* documentation, including the *CookBook*, is now available on the "World-Wide Web" so that it may be examined and retrieved over the Internet (start with "URL" `https://www.aips.nrao.edu`). This edition of the *CookBook* is issued in a ring-binder format with a chapter-based page numbering system. This allows us to update individual chapters without altering the pagination of others and to make each chapter available individually over the Internet. The documentation is also included with every copy of *AIPS* shipped.

Additional written documentation on *AIPS* is available in several forms. A programmers' reference manual called *Going AIPS* is available in two volumes. This was revised completely for the `15APR90` release due to the upgrading of the *AIPS* code to FORTRAN-77 and to reflect the extensive additions and improvements to the software. Unfortunately, it has not been revised since but it continues to be quite useful. The first volume is intended for applications programmers, while the second volume is needed by programmers developing *AIPS* for new peripheral devices or computers. *Going AIPS* may be obtained from the *AIPS* web site.

AIPS provides run-time documentation in the form of `HELP` and `EXPLAIN` files which may be viewed at the terminal or printed. (See § 3.8 for explicit instructions.) Should these not suffice, consult your local *AIPS* Manager and then, if needed, e-mail the *AIPS* programmer at `daip@nrao.edu`. Although individual *AIPS* programs have often been written, and are best understood, by a single programmer, the *AIPS* group as a whole assumes responsibility for all released software. Anyone in the group will attempt to help you or, at least, to identify another member of the group better able to help you.

Finally, users are encouraged to recommend new and better analysis and display tools and to help debug the existing software by entering "Gripes" (see § 11.1). Please note that examples of bugs that are documented by printouts of inputs, message logs, *etc.* are most useful to the programmers. Also note that written bug reports are *much* more effective than verbal reports. E-mail to `daip@nrao.edu` reaches everyone in the group.

1.3 Organization of the *CookBook*

1.3.1 Contents

Chapter 2 of the *CookBook* describes in general terms how to get started in *AIPS* — signing up, logging in, mounting tapes, *etc.* Appendix Z gives details of these operations specific to NRAO's *AIPS* sites. Your local *AIPS* Manager may be able to provide a version of this appendix appropriate to your system. Chapter 3 introduces the basic *AIPS* utilities. Chapter 4 leads you through the basics of reading in and calibrating your modern VLA *uv* data. Chapter 5 explains the basic operations required to make and improve images. Appendices A, B, and C provide simpler recipe-like approaches to calibration and imaging which beginning users may wish to try. Chapter 6 introduces the basic *AIPS* tools for making interactive and hard-copy displays of images and other data and Chapter 7 describes tools for analyzing them. Chapter 8 and Appendix B contain hints and further *AIPS* tools of particular interest, but not restricted, to spectral-line users and other observers who have images of more than 2 dimensions. Appendix O is designed for users of the historic VLA archive and for older VLBI formats. Similarly, Chapter 9 and Appendix C are aimed primarily at users of VLB interferometers. Chapter 10 deals with single-dish data reduction with *AIPS*. Chapter 11 describes how to help the *AIPS* programmers, to backup your data, and to exit from *AIPS*. It also suggests some cures for common hang-ups and miscellaneous "disasters"

which seem to afflict *AIPS* users. No such list can be made comprehensive or sufficiently general to cover all the computer systems now running *AIPS*. You will need to consult with your local *AIPS* Manager or other users if you encounter an unlisted problem.

Chapter 12 is intended for the "mature" *AIPS* user who wishes to learn about data formats, procedures, RUN files, and various subtleties of AIPS syntax. We recommend that you read this after becoming familiar with the operations described in Chapters 3 through 7. Chapter 13 contains lists of all available routines broken down by categories. Appendix G presents Fred Schwab's Glossary of radio astronomy data processing terminology. Appendix F gives some useful recipes for estimating disk files sizes and for saving data and images on tape. Appendix V contains information about tasks used to maintain the VLA, including holography of dish surfaces. Appendix I contains the index.

1.3.2 Minimum match

In this *CookBook*, we use the minimum-matching capability of *AIPS* to abbreviate the instructions needed to run the programs. This speeds up your activity at the terminal while working in *AIPS*. However, the full names of some of the AIPS instructions may be easier to learn and to remember. They are given in Chapter 13.

1.3.3 Fonts and what they signify

Throughout this *CookBook*, RESPONSES TO BE TYPED BY THE USER APPEAR IN THE PRESENT FONT. Prompts provided by the operating or *AIPS* systems are left-justified on the same line, *e.g.*, system prompts $ on VAXes or % on UNIX systems , AIPS prompt >. THIS IS THE FONT USED FOR SAMPLE OUTPUTS FROM THE COMPUTER and for program names such as PRTUV. A lower-case italic font, *such as this*, is used for numeric and character parameter values which must be supplied by the user. The symbol AIPS refers to the program which you will use to communicate with the computer. The symbol *AIPS* refers to the full system, made up of the AIPS program, numerous other programs which may be run from AIPS, and the hardware configuration. The symbol $\subset_R$ means "hit the RETURN or Enter key on the terminal."

The symbol § means Section and refers to the various chapters and sub-chapters of this *CookBook*. Except in the values assigned to character string variables, *AIPS* is case insensitive. We use upper-case letters in this *CookBook* to differentiate *AIPS* symbols from ordinary words visually. This usage also allows us to generate html and pdf capable versions of the *CookBook* from the basic TEX files automatically.

1.4 General structure of *AIPS*

The diagram on the next page is an attempt to show the general structure of the *AIPS* software package. You may wish to refer to it as you read the remaining chapters. Input to *AIPS* is either interactive or batch via the main AIPS program. It uses the *POPS* language processor and symbol table to set "adverb" values and cause application "verbs" to be executed. Chief among these, the verb GO causes independent programs called "tasks" to run. Sequences of commands may be run in batch by SUBMITting them to a batch "checker" and running them using the batch version of AIPS. All programs access and modify disk data files, and interactive ones may access tapes and TV- and Tektronix-like display devices.

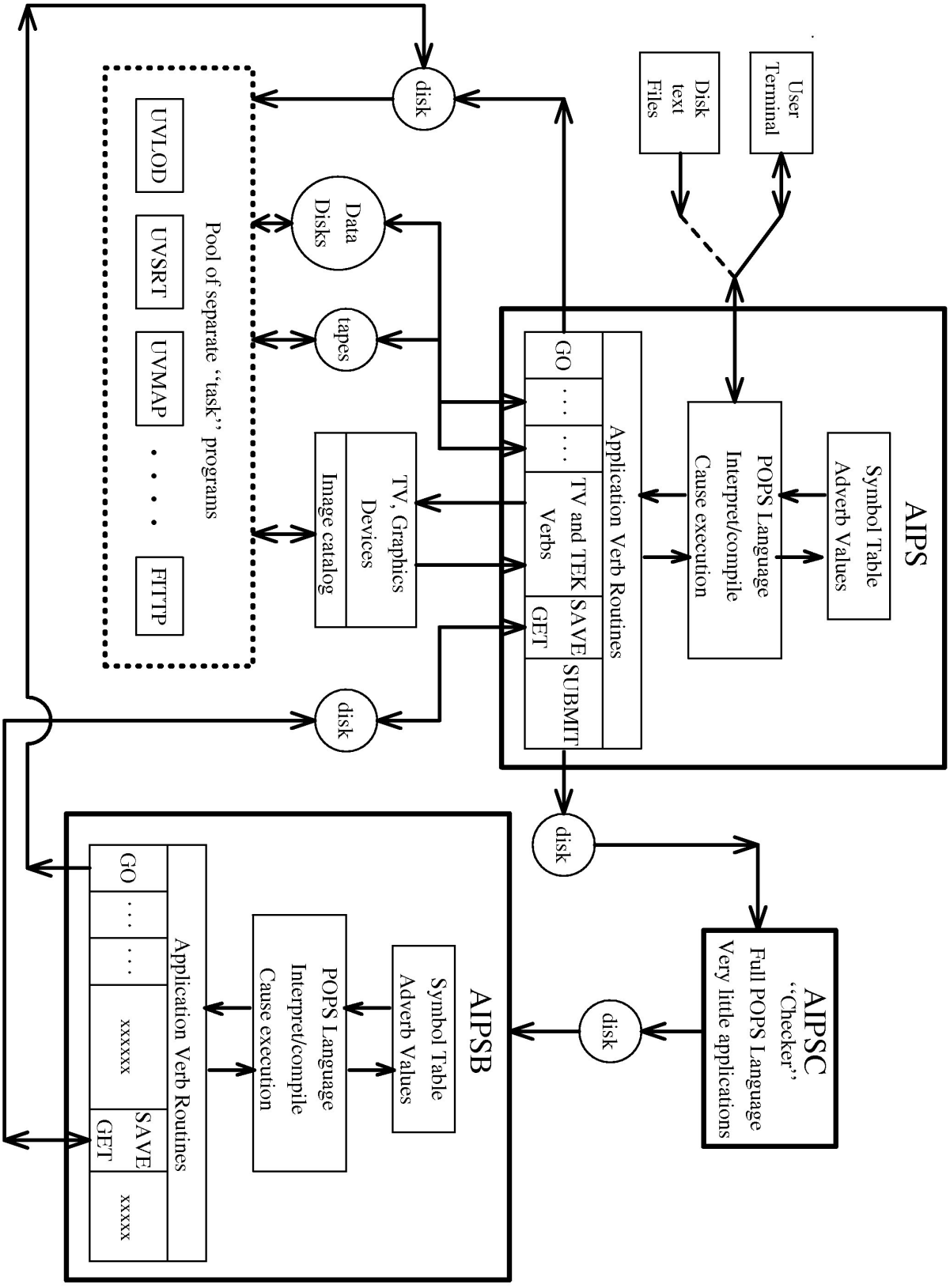


Figure 1.1: Basic AIPS flow diagram

2 Starting Up $\mathcal{AIPS}$

This chapter contains general information concerning the steps needed to obtain access to, and use, an $\mathcal{AIPS}$ system. It attempts (as does the design and coding of $\mathcal{AIPS}$ itself) to avoid specific references to particular computer devices and to the peculiarities of any one $\mathcal{AIPS}$ installation. We will assume, for the most part, that you will be running $\mathcal{AIPS}$ on a Unix workstation although $\mathcal{AIPS}$ should still work in more classical environments. Even for workstations of the "same" operating system, some installation-specific practicalities remain. For the NRAO installations, these are described in Appendix Z.

2.1 Obtaining access to an $\mathcal{AIPS}$ computer

Most $\mathcal{AIPS}$ sites now possess a number of computers which are networked together and are each individually capable of running $\mathcal{AIPS}$ while sharing both disk resources. Most such computers cannot support more than a few simultaneous users (or simultaneous incarnations of the same user) of $\mathcal{AIPS}$. Thus, most locations are obliged to institute a mechanism for distributing the available $\mathcal{AIPS}$ time to the people desiring it. At NRAO, some of the computers are assigned to individual staff members and are normally used only by them. Other computers, including all of the most powerful ones, are for "public" use, but are mostly still on an assigned basis. You should arrange to have a workstation assigned to you for your $\mathcal{AIPS}$ processing. A few of the computers are available on a first-come, first-served basis, and are often used remotely. There may be sign-up sheets and rules for their use posted in or near the principal "$\mathcal{AIPS}$ Caige" (user-terminal room). To promote fair and efficient use of the system, there are often restrictions on the amounts of time that any one user or user group may reserve.

$\mathcal{AIPS}$ can support several simultaneous users which it calls `AIPS1`, `AIPS2`, *etc.* In the workstation environment, this is used primarily to allow one user to have separate simultaneous $\mathcal{AIPS}$ sessions using multiple windows. This also allows users to log in to remote computers (*e.g.*, with the Unix tool `slogin`) and run $\mathcal{AIPS}$ while remaining comfortably ensconced in their offices in front of their own (presumably lesser) workstations. You should not do this, of course, without permission.

2.2 Using the workstation

The way that a workstation behaves is a function of the type of workstation, the computer operating system, the window manager program, and the set-up files for the specific computer account being used. Given all these variables, it is hard to give detailed usage instructions. Nonetheless, it is important for beginning users to master the foibles of the workstation(s) they will be using.

2.2.1 Logging in to the workstation

Find your assigned computer in the appropriate $\mathcal{AIPS}$ caige or office, or an available one intended for general use (checking any sign-up sheets for it). Typing $^{\text{C}}_{\text{R}}$ on the keyboard will reveal the current state of the workstation. If you see a message prompting you to log in (*e.g.*, `AOC RedHat Linux, [monkey] login` on a Socorro Linux workstation named monkey), then the computer is ready for you to log on. Type the account name you are supposed to use for $\mathcal{AIPS}$ followed by a $^{\text{C}}_{\text{R}}$ (use `Tab` in forms) and then type the password (it will not be visible on the screen) followed by another $^{\text{C}}_{\text{R}}$. See your $\mathcal{AIPS}$ Manager for the account to use and its password (which should change with time). Many sites will assign an account to you

personally, while some use a more generic AIPS account. The login scripts should start the window system automatically and produce one or more `xterm` or `aixterm` windows that you can use for starting `AIPS`.

If the initial $^{C}_{R}$ produces instead a set of windows (and/or icons), the computer is already being used. If these windows include the AIPS TV and possibly the `TEKSRV` and `MSGSRV` server windows, it is being used for `AIPS`. Check with other possible users before proceeding. If it's okay to use the system, you should log the previous user out and log in for yourself, restarting the window system. If you are patient, you can open each iconified window (by clicking on it once or twice), see what it's doing and finish up and/or exit. If the prompt is > in any text window, `AIPS` is running there and you should type:

> KLEENEX $^{C}_{R}$

which will get out of `AIPS` and kill the servers. Then once at the system prompt (Unix), you can type exit $^{C}_{R}$ (lowercase!) to make the window go away. If the `XAS` *AIPS* TV server is still running, just press the escape key while the cursor is in the TV window. For the `MSGSRV` message server, move the cursor into the window and press `CTRL C`. Finally for the Tek server, hold the control key down while you press the left mouse button, and choose the `QUIT` option.

The procedure for exiting from the windowing system will depend on what window manager you use. If your system uses `KDE`, there will be an icon on the icon bar with a large `K` superimposed on a globe. If the system uses Gnome, then the magic icon is an image of a foot. Move the mouse to the icon and hold down the left button. A pull-down menu will appear; select the `Logout` function.

2.2.2 Control characters on the workstation

To correct characters which you have typed, you may have to press either the `BackSpace` key or the `Delete` (or `DEL`) key. Unfortunately, which is required varies with the application you are using and how the AIPS account (or your personal account) has been set up. For details, see the manual page on `stty` with particular note of the `erase` function.

A control character is produced by holding down the `CTRL` (or `Control`) key while hitting another key. Some control characters under Unix have characteristics that may confuse users more used to other environments (VMS, MS-DOS). In particular, `CTRL D`, `CTRL T`, `CTRL Y` and `CTRL Z` behave much differently under Unix than under VMS. `CTRL D` is used in Unix as a signal to logout, unless otherwise inhibited. If you use the AIPS accounts at either Charlottesville or the AOC, this feature is automatically disabled. While in `AIPS`, `CTRL D`s are interpreted from the > prompt as an `EXIT` command. `CTRL T` (under GNU `readline`) transposes two characters, while `CTRL Y` inserts characters previously saved in the "kill" buffer. `CTRL Z` suspends the current process, printing `Stopped` on your window and leaving you at the Unix prompt level. The `Stopped` message does *not* mean that the process has been terminated. It simply means the process has been suspended and placed in the background. For *AIPS* users, the suspended process is normally `AIPS`*n*. Users who do not understand this state often start up another *AIPS* session. In doing so, they are tying up a second *AIPS* number. If a user does this enough times, s/he can eventually tie up all available `AIPS`*n*'s. If you are unfamiliar with the use of `CTRL Z` (suspend) in Unix systems, it's best not to use them from *AIPS*, unless expert advice is close at hand. With an X-Window display, it is preferable to pop up a new `xterm` or other window and do whatever you want in it, leaving the *AIPS* session undisturbed. (You can get a new `xterm`, usually, by moving the cursor into the root (background) window, pressing the right mouse button, and selecting the appropriate option.) If you have suspended the current process (usually `AIPS`) with `CTRL Z` to get to monitor level (for instance, to edit a `RUN` file), then you can bring the suspended process back into the foreground with the command `fg` $^{C}_{R}$.

To abort any execution in your window, type `CTRL C`. Using `CTRL C` while in `AIPS` will unceremoniously eject you to the Unix prompt. You will have to restart `AIPS` with all the input parameters having been lost. In some cases, any *AIPS* tasks running in the background, and maybe even the TV and other servers, will also be "killed" and will disappear from the screen. Aborting *AIPS* "tasks" (sub-processes) is usually done

from within AIPS with the command ABORT *taskname* ↵ (see § 3.1.2) rather than with CTRL C's or Unix-level system commands. Not only does this avoid killing AIPS, but it even allows for orderly deletion of scratch files.

During execution, scrolling of output lines out of the window can be halted by typing CTRL S and resumed by typing CTRL Q. If you are using an xterm (or cmdtool or aixterm) window with a scroll bar, you probably won't have to worry too much about doing this; use the scroll bar to review lines which have rolled off the visible part of the window. You can specify how many lines these terminal emulator windows remember, *e.g.*, for xterm with the -ls option or with the X resource xterm*saveLines (in your .Xdefaults or .Xresources file).

2.2.3 Starting the AIPS program

As you enter the commands needed to log in to your system and start AIPS, please read all messages which appear. They are often important and relate to current system, disk, and *AIPS* problems which may affect your reductions.

To begin AIPS, enter

% aips ↵ with no options initially

You will then be shown a list of printer devices and be prompted to Enter your choice:. You will then be told about the assigned printer queue and data disks. If all is going well it will then tell you

```
You seem to be at a workstation called monkey
Starting local TV servers on monkey
```

where *monkey* is the name of your workstation. Any news messages about your *AIPS* installation will then appear. Read them; they might be important. Finally, you should see the messages:

```
Starting up 31DEC24 AIPS with normal priority
BEGIN THE ONE TRUE AIPS NUMBER n (release of 31DEC24) at priority 0
```

where 31DEC24 identifies the release of *AIPS* and n is a number between 1 and 6 (typically). If this is the only *AIPS* session on the computer, you should be assigned $n = 1$, with higher numbers used for additional sessions. If you start with $n > 1$, someone else may be using your computer remotely. AIPS will then tell which TV and graphics devices have been assigned to you:

```
AIPS n:  You are assigned TV device nn
AIPS n:  You are assigned graphics device mm
```

where *nn* and *mm* are numbers assigned to your workstation (or, rarely now, to real TV and graphics devices). AIPS will now ask you for your user number and provide a ? prompt:

```
AIPS n:  Enter user ID number
?  uuuu ↵
```

where *uuuu* is the number assigned to you for the local *AIPS* system (in decimal form). The AIPS prompt > should now appear.

There is more. Notice the line above that says "starting local TV servers on monkey"? At that point, the process of figuring out what computer you're running on and what display you're sitting at (they may be different) is shed in an asynchronous way while the main process of starting the AIPS program proceeds. Then, sometime later, you will see the following messages appear in the same window:

```
XASERVERS: Start TV LOCK daemon TVSERV on monkey
TVSERVER: Starting AIPS TV locking, Inet domain
XASERVERS: Start XAS on monkey, DISPLAY monkey:0
XAS: ** TrueColor FOUND!!!
```

```
XAS: *** Using shared memory option for speed ***
XAS: Using screen width height 2540 1920
     max grey level 8191 in 16 grey-scale memories
     characters multiplied by 2
XASERVERS: Start graphics server TEKSRV on monkey, DISPLAY monkey:0
XASERVERS: Start message server MSGSRV on monkey, DISPLAY monkey:0
```

Each of the first four messages should announce the starting of one of the servers. The Tek server will appear in iconified form somewhere on the screen, while the message server will appear opened (not iconified) somewhere else. Finally, the XAS TV server appears in opened form. An environment variable can be set to have the Tek server appear in open form and a `.Xdefaults` (or `.Xresources`) file option may be set to have the TV appear in iconified form. If your X-Windows supports 24-bit TrueColor, then XAS will use it. Otherwise, XAS will use 8-bit PseudoColor which is faster but less flexible. In this case, if you have a lot of colors in your X11 display (*e.g.*, an image on the root window displayed with `xv`), you may also get the message:

```
XAS: Using screen width height 1142 800, max grey level 189
XAS: Warning -- creating virtual colormap
```

which means XAS wasn't able to find enough free colors in the main colormap (189 in the above example) and had to create its own. In this case, the colors of every other window will "flash" when you move the mouse cursor into the opened XAS TV window, and vice versa. You can use `xsetroot -solid navy` command (or other legal X colors) to blank out whatever is on the root window; then restarting AIPS will restart the TV server, hopefully without a virtual colormap. There are a number of X-Window parameters which may be specified in your `.Xdefaults` or `.Xresources` file for these three windows. After AIPS begins, type HELP XAS ↵, HELP MSGSRV ↵, and HELP TEKSRV ↵ for details. Among these is a parameter controlling how many colors XAS tries to use in PseudoColor and whether it tries to use TrueColor or not. Character size multiplication occurs by default when you have a large number of pixels on your screen. It can be controlled by an `.Xresources` parameter and the AIPS verb CHARMULT. See § 2.3.2 for more information about XAS.

There are several options you can use in starting up AIPS. To see them, just enter `man aips` at the Unix command prompt, or if you are already in AIPS, type HELP AIPS ↵. This information is reproduced in part below:

```
aips [OLD, NEW, or TST]
     [TV=[disp][:][host]]
  or [TV=local[:n]]
  or [TV=inet]
  or [NOTV]
     [TVOK]
     [DA=host[,host,...]]
  or [DA=default]
  or [DA=all]
     [PR=#]
     [REMOTE or REM or TEK]
     [DEBUG[=prog][:aips]]
     [LOCAL] [NORL] [NOEX]

DESCRIPTION
     The aips or AIPS command starts up the AIPS command interpreter and
     associated AIPS server processes.

OPTIONS
     All command line options are case insensitive.
```

```
AIPS allows up to three versions to co-exist (disk space permitting) in one
installation.  They are identified either by date (e.g. 31DEC24) or name
(OLD, NEW, or TST).  On most installations, these will all be the same.

OLD     Start the OLD version of AIPS.  For NRAO this is a frozen version
        which has been distributed worldwide.  Serious bugs are rarely
        corrected.

NEW     Start the NEW version of AIPS.  For NRAO this is the most recently
        released version and is "frozen" at the date in its name.
        Serious bugs are corrected over the next year.

tst     Start the TST version of AIPS.  For NRAO this is the development
        version which is also available to non-NRAO sites.  This is the
        default.

TV=[tvdisp][:][tvhost] or TV=local[:n] or TV=inet

        The default TV display server is now (31DEC24) TV=local:0.

        TV display server to use instead of the default.  The AIPS startup
        script tries to deduce which host the user is sitting in front of
        (this may not work; it is often difficult or impossible to determine
        this  information).  This may not be the same as the machine on which
        AIPS is to be run if, for example, the user has remotely logged in to
        another machine within a terminal emulator window.

        The  "TV=local" option allows use of Unix based sockets for the TV and
        other servers.  If you choose this option, you MUST run the XAS server
        and any AIPS sessions that will use it on the same host, though the
        DISPLAYs can be the same or different.  Also, no remote AIPS sessions
        will be able to talk to this local TV.

        If you instead use "TV=local:0", it will attempt to start a new
        instance of the TV and ancillary servers.  This can be used to have
        multiple TV's on the same host, and is useful in a compute server
        environment with X terminals.  If you have multiple Unix-socket based
        TV's already started, you can choose which one a new AIPS session will
        use by, e.g. "TV=local:2" to choose the second one.  Note that the
        n in ``TV=local:n'' is in extended HEX or 0 to 9, A to Z.

        NOTE: The default TV behavior is no longer to use internet
        based sockets.  Such use is mostly when more than one
        computer must talk to the same incarnation of XAS, which is
        an uncommon usage.  Internet sockets also require compatible
        entries in the system file /etc/services.

        For the use of internet sockets, the full syntax of the TV= option
        is TV=tvdisp:tvhost, where tvhost is the name of the machine on which
        the TV display server (usually XAS), Tektronix  graphics  server
        (TEKSRV), message server (MSGSRV), and TV Lock server (TVSERV) are to
        run, and tvdisp indicates the machine to which the DISPLAY environment
        variable should point for XAS.  Do NOT specify TV=hostname:0.0!  Both
```

TVHOST and TVDISP can be different from the machine that AIPS itself is running on. See the section on X Window System servers below for more information on how to control the servers.

The default behavior of this option if only one of tvdisp and tvhost is specified is

TV=tvhost tvdisp defaults to tvhost.

TV=tvdisp: tvhost defaults to the host AIPS is running on.

TV=:tvhost tvdisp defaults to the host AIPS is running on.

For the remote TV options to work, you must be able to use the rsh or remsh command; see the notes on it under the tp= heading below. Also see the notes on environment variable AIPSREMOTE. By default, if you do not specify any tv= option, you will only get a TV if your current TERM environment variable matches sun*, *xterm*, *hpterm, dtterm, or iris*. The DISPLAY environment variable is used if set, otherwise the who am i (on HP-UX, with the -R option) is used to make a guess at "where" you really are.

NOTV Prevents automatic activation of the TV server if no display is wanted. This option also disables the Tektronix graphics server, the message server and the TV lock server. See the section on X Window System servers below for information on how to control the Tektronix and message servers.

TVOK Assume that the TV display servers are already running; the particulars (display, host) are still worked out -- from the TV=... argument (see above) if necessary -- but no servers will be started.

DA=host[,host,...] or DA=default or DA=all
Select user data areas (directories, or "disks" in AIPSpeak) that are local to the (comma separated) list of machines. Data areas from "required" hosts and those on the local machine are always added, regardless of the list of hosts.

All disks from each named host will be assigned. Use the FREE command within AIPS to see the disk assignments you end up with. They are also shown on startup.

AIPS has a limit of 35 disks in any one session. The limit on the number of disks that can be defined for any given site is 512. Disk 1 is special in that it stores the AIPS message and save/get files. The system is designed so that one particular required disk will almost always be assigned as disk 1. For performance reasons, this may be undesirable if the filesystem in question is mounted via NFS. See the description of personal .dadevs files below, as it can be used to customize the list of possible user data areas.

Selecting DA=ALL will try to include every area defined in the startup file, up to the session limit. Bear in mind that most AIPS tasks only

```
have 10 slots for "BADDISK".   Selecting DA=DEFAULT will completely
bypass the configurable data areas and choose only those data areas
pre-configured by the AIPS manager; THIS IS NOT NORMALLY ENABLED, CHECK
WITH YOUR AIPS MANAGER BEFORE USING DA=DEFAULT.

There is a hierarchy of data area "lists" that AIPS will look for on
startup.  These are:

$HOME/.dadevs       This would be in your private login area (what
                    $HOME points to).   It need not exist.  If it
                    doesn't, AIPS looks for the next file:

$DA00/DADEVS.LIST   This is a host-specific file possibly set up by
                    the AIPS manager.  If it doesn't exist, AIPS
                    finally looks for:

$NET0/DADEVS.LIST   which is the site-wide data area configuration
                    file.
The  normal state of affairs is to have just one place for disks to be
defined, namely $NET0/DADEVS.LIST.  Your AIPS manager can choose to
install host-specific list files, and you can choose (if you run AIPS
from your own private account) to override both of these two with your
own private  version.   This  allows for considerable flexibility but
moves the onus of maintenance of these files to the  user.   In other
words,  if  you  have your own .dadevs file, you have to keep track of
your site's disk configuration!

If your AIPS installation supports multiple  sites,  e.g.  to support
both little-endian (Intel, Mac) and big-endian (Sparc) systems, you
can have any of these files refer to one or the othe site by appending
the site name, e.g. $HOME/.dadevs.VCOARN for SITE=VCOARN.

The  format for these files is all the same: a list of directory names
preceded by a "+" for required or a "-" for optional.  There should be
two (2) spaces between the "+" or "-" (in the leftmost column) and the
directory name.

In addition to all of the above,  you may define a list of  data areas
in an optional $HOME/.dadevs.always file.  This is used in addition to
whichever of the DADEVS files have been selected by the rules above.
The data areas that you will be assigned  start with any required data
areas in the $HOME/.dadevs.always file followed by any required data
areas in the selected DADEVS file regardless of name.   The 3rd group
of data areas are those optional ones in the selected DADEVS file
containing a string matching the user's host name.  Then come the
optional data areas in the  $HOME/.dadevs.always file regardless of
name.  Finally, the optional data areas in the selected  DADEVS file
with strings matching the names of any hosts given in the
DA=host[,host,...] command-line  option.  The order of data areas
within each group,  i.e. which is disk 1, etc.,  is determined by the
order in the files.

There is also a $NET0/NETSP file that is maintained by the AIPS manager
```

```
          and controls aips user-number access  to the disks.  Regardless of
          the number of sites in your installation, there is only one NETSP file.

       TP=host[,host,...]

              Make sure tape servers are running on the comma separated list of
              machines.  While the AIPS account is usually set up so that it can
              perform remote shell (rsh) commands, your personal account may not.
              Check with your system administrator or network guru for details.
              Also check the Unix manual pages on rsh (remsh on HP-UX), rhosts, and
              hosts.equiv.  The tp= option uses rsh to issue commands to remote hosts.

       TPOK   Do NOT check or launch the TPMON tape daemons on the local host.  The
              default is no longer to check or to start the tape daemons.  Therefore,
              if you need them, use TP= above even for the current computer.

       PR=#    Select printer number (e.g., pr=2).  If this option is not specified,
               the user will be presented with a menu of available printers and
               prompted to enter a choice.  If there is only one printer configured,
               no menu will be presented.  You may change the selected printer within
               AIPS via the PRINTER adverb.

       REMOTE or REM or TEK
               Any one of these indicates that the user is running from a terminal
               with Tektronix display capability.   Graphics output will be sent
               directly to this terminal.  NOTE: AIPS will not switch from text to
               graphics mode on terminals with a separate graphics "screen".

       DEBUG[=prog][:aips]
               Start AIPS in debug  mode.   With no arguments, the user will be
               prompted for the name of the debugger (e.g. gdb, dbx, adb, csd, xde,
               dbxtool, debugger, xxgdb) and also whether to run AIPS itself under
               the  debugger.  If you answer no, only AIPS tasks will be run in debug
               mode.  If =prog is specified, this suppresses the prompt for the name
               of the debugger program.  If :aips is specified, this suppresses the
               prompt for whether to run AIPS itself in debug mode and assumes it
               will.   Use of both these options is useful in speeding up the startup
               of the system when debugging a program or AIPS itself.

       LOCAL   Start a local copy of AIPS.EXE  residing  in  the  current  directory.
               Usually used by programmers for debugging purposes.

       NORL    Disable  GNU  readline library and command-line editing.  This is pri-
               marily useful for running backgrounded  AIPS  sessions,  running  AIPS
               from "here-document" shell-scripts, and for debugging.

       NOEX    This defers AIPS execution and is not normally used directly by users.


X WINDOW SYSTEM SERVERS
       If you are running under the X Window System, AIPS will open up to three win-
       dows: a TV window (normally XAS), a message window (MSGSRV) and a graphics
       window (TEKSRV).  If you specify the notv option on the command line, none of
```

```
       these will be started.

       MSGSRV and TEKSRV are actually simple programs running inside a terminal  emu-
       lator.   You  may use any terminal emulator that you would normally use on the
       machine on which you are running AIPS for the MSGSRV window.  Examples include
       xterm  (the  sample  vt100/Tektronix emulator that comes with the MIT X Window
       System code); cmdtool and shelltool (the standard terminal emulators for Open-
       Windows) and AIXterm (the standard terminal emulator on RS/6000s).  You can
       choose which one to use by setting the environment variable AIPS_MSG_EMULATOR
       to the name of the terminal emulator you wish to use.  For example, if you
       want to use cmdtool you would type

       setenv AIPS_MSG_EMULATOR cmdtool

       if you use the C or TC Shell, or

       AIPS_MSG_EMULATOR=cmdtool; export AIPS_MSG_EMULATOR

       if you use Korn, BASH, or Bourne shells before you start up AIPS.   You  could
       also add these  commands to your .login file (C Shell) or .profile
       (Korn/BASH/Bourne Shells) to make the assignment more permanent.  You can also
       give AIPS_MSG_EMULATOR the special value of "none" which will disable the mes-
       sage window without affecting the Tektronix window or the TV.  If
       AIPS_MSG_EMULATOR is not set, the default is xterm.

       You may choose the terminal emulator used for the Tektronix window using the
       environment variable AIPS_TEK_EMULATOR in the same way that you use
       AIPS_MSG_EMULATOR to choose the terminal emulator, but it must support Tek-
       tronix graphics codes.  On most machines the only values of AIPS_TEK_EMULATOR
       that  make any sense are xterm and none.  If AIPS_TEK_EMULATOR is not set AIPS
       will behave as if it were set to xterm.  (Note: dxterm, aixterm, and cmdtool
       are not "xterm"; they cannot display Tek graphics).

       You can set preferences for positions and colors for all three servers using
       the standard X Window System mechanisms.   Further information is available
       through the AIPS HELP system (subjects MSGSRV, TEKSRV, XAS and XVSS).

       Note that AIPS expects that a terminal emulator can start a program that is
       specified using a -e flag on the command line.  This is true of all of the
       terminal emulators we know about but if you find one that requires a different
       flag you can specify the flag as AIPS_TEK_EXE_FLAG or AIPS_MSG_EXE_FLAG.

ENVIRONMENT VARIABLES
       In addition to the Message and Tek server customizations, you may choose to
       set a variable AIPSREMOTE to indicate your choice of remote shell command.  It
       is strongly recommended that the secure shell (ssh) be used in place of the
       traditional Berkeley rsh or remsh command:
              setenv AIPSREMOTE "ssh -n"
       for csh or tcsh shells, or
              export AIPSREMOTE="ssh -n"
       for bash, korn, zsh and other bourne-like shells.
```

If you do not specify a printer (by number) on the command line when starting AIPS, you will get a menu showing you all the alternative printers available. You should omit the PR option until you are familiar with

the choices. The OLD version of *AIPS* is likely to be relatively free of bugs (provided the *AIPS* version in NEW does not prescribe format changes which prevent OLD from working), but the NEW version will contain improvements and will be mostly debugged. The TST version is a debugging area recommended for NRAO staff and those few users who may require the most recent software. (Note that this choice affects only the version of the AIPS program itself. You may choose TST, NEW or OLD versions of the *AIPS* reduction programs at a later time — see § 3.5.)

2.2.4 Typing commands to the AIPS program

As of the 15JUL95 release, *AIPS* is available to users under a GNU-style license. This has numerous benefits, one of which is that it allows us to incorporate other GNU-style code within our system. The first of these is the GNU readline library which provides the user-input interface for AIPS under Unix beginning with the 15JAN96 release. The GNU readline library gives the user the ability to use the cursor-arrow keys, as well as various "control" and "escape" key sequences, to recall previously-entered commands, to edit the current command line (without having to back-space and re-type the entire line), to search the command history for previously-executed commands, to define customized key bindings for executing commands and macros, and much more. The full information may be obtained with the command man readline from the system command line (not inside AIPS). There is even "tab completion" based on the list of *AIPS* help files and on context. At any point, when typing a symbol, you may hit the TAB key. The symbol name will be completed if it is unique or the screen will flash (or the bell sound) is it is not. A second hit on the TAB key will produce a list of the possible completions. Since a task name cannot be the first symbol on a line, tasks are included in the possible completions only after some other symbol appears on the line.

The default key bindings should be very familiar to users of emacs and/or the bash shell; many of them should also be recognizable to users of the Korn and tcsh shells. Hard-core vi users can put AIPS into "vi-mode" and use vi-like key bindings instead. (The basic emacs-like key bindings will be outlined below; it will be assumed that those who are using the non-default vi-like key bindings already know what they are doing.)

Your command-line history is automatically saved between sessions, unique to both the user number and the "*AIPS* number" of the session, and then recovered at the next AIPS startup.

Use of the GNU readline library for input can be disabled on a per-session basis by starting AIPS with the "norl" option. This can prevent problems under some operating systems (most notably HP/UX) with putting AIPS into the background, when running with input "fed" from a script, or when debugging AIPS itself.

The key bindings are given below. Key sequences/bindings using the CONTROL key will be prefixed below with "C-." Those using the ESCAPE key (or "META" key — often available as the ALT key on PC keyboards and as the "diamond" key on Sun keyboards) will be prefixed with "M-." The basic cursor-movement key bindings are:

```
C-b     backward-character      [also: left-arrow]
C-f     forward-character       [also: right-arrow]
C-p     previous-command        [also: up-arrow]
C-n     next-command            [also: down-arrow]
M-b     backward-word
M-f     forward-word
C-a     beginning-of-line
C-e     end-of-line
C-r     incremental-search backward
C-p     previous-history (move backward in history list)
C-n     next-history (move forward in history list)
```

The basic editing key bindings are:

```
C-d      delete-character (under cursor)
M-d      delete-word (to right of cursor)
M-DEL    delete-word (to left of cursor)
C-t      transpose-characters (left with under cursor)
```

`DELETE` and `BACKSPACE` work as expected.

2.3 Managing windows

Unfortunately, the management of windows on a workstation screen depends heavily on the type of window manager and on the setup files defined for your login. At best, we can only be approximate here and try to describe general characteristics of normal setups.

2.3.1 General window management

Most window managers allow multiple windows to be created on the screen at the same time. These windows can either be closed in a small "iconified" form or opened in a larger and more usable form. Windows are normally opened by positioning the cursor on the icon with the mouse and clicking either once or twice with the left button. You can type only into open windows. An open window can be re-sized usually by "grabbing" (position the cursor with the mouse and then hold down the mouse button) one of its corners with the left mouse button. Windows under `twm` have a widget in the upper right corner which must be grabbed with any of the buttons. Positioning the cursor in the top bar of a window border and holding down a mouse button will do something. Usually, the left button moves the window, the middle button puts the window above or below other windows, and the right button gets you a pull-down menu of all the window manipulation options. Under `Motif` the middle and right buttons are switched. In the upper left corner of the top bar is a special button widget. Under `Openlook` and `twm`, clicking on this widget iconifies the window. Under `Motif` the iconify widget is shown as a dot and is usually in the upper right corner. The widget in the upper left corner under `Motif` offers a pull-down menu of window options, but is dangerous since a double click on that widget with the left button destroys the window (and any programs running in it).

Positioning the cursor in the root window (the background) and holding down a mouse button usually gets a pull-down menu with programs that can be run and various other options including exiting from the system. Well-configured systems offer a separate menu with each button. This is usually the way to get more windows if you need them.

When encountering a system for the first time, you should explore what the various controls have to offer. Position in the background, press each button in turn, and follow up what is offered by the pull-down menus which appear. Many menu items may themselves have a menu which you get by dragging the cursor to the right. Usually there is an arrow at the right of the menu item to indicate this. Then, try to open some icons with a single click or a quick double click. Then try the various mouse buttons in the top bar, the corners, and any special widgets visible in the window. Some windows also have scroll bars along the left or right side. Experiment with the various mouse buttons, clicking or dragging, in the scroll-bar region to see how to scroll back to previous text or forward to the last line. It pays to master all this slight of hand to allow you rapid access to multiple windows, previous text, and the like. It is very painful to click the wrong button and destroy a program that has been running for a few hours already!

2.3.2 Managing the *AIPS* TV window called XAS

On workstations, *AIPS* simulates a real TV display with a program called XAS. The program starts when you start AIPS and comes up in an open form. Its icon shows a cute drawing of an ape along with words like AIPSTV UNIX for local display and AIPSTV INET for Internet-connectivity. In many ways, this is a normal window which can be re-sized, moved, iconified, and destroyed like any other. However, when the window is open and the cursor positioned inside the window, XAS offers some additional features. The cursor changes shape and color in the window to indicate this fact. To get XAS to treat the cursor position as a "TV cursor" position, you must hold down the left mouse button. This allows the cursor to fill two rôles at nearly the same time, that of a workstation cursor and of a TV cursor. You do not have to hold the button down for long to register a TV position and, in fact, it is more efficient in interactive TV operations simply to click the left button at the desired locations. When you drag the cursor, numerous intermediate values are read with consequent extra computation. Note that the TV cursor position is read by XAS whenever the cursor is in the XAS window with the left button down. However, that position is only used when some verb or task reads it from XAS and uses it for some purpose, *e.g.*, to select image coordinates or to control image enhancement.

AIPS TV functions refer to "buttons" A, B, C, and D for the purpose of signaling conditions to the software. In the XAS simulation, these buttons are the keys a, A, or F3 for button A, keys b, B, or F4 for button B, keys c, C, or F5 for button C, and keys d, D, or F6 for button D. The F2 and F7 buttons toggle the size of the display from full screen to whatever size you set the window. XAS simulates a TV with a number, usually sixteen, of grey-scale memories and eight one-bit graphics overlays. The x and y dimensions of the memories adapt to the display area of your workstation less some room for window borders and, sometimes, for a few lines of a type-in window as well. XAS has the ability to display full-color (256 levels for each of red, green, and blue) on terminals capable of supporting full "TrueColor visuals." The internal dynamic range of images in XAS is actually 0:8191, but displays are limited to 0:255. You pay for this capability with a small reduction in speed for ordinary enhancements, blinks, and the like. You may select to limit your XAS to a "PseudoColor visual" which is all that is available on some older workstations. In that mode, the higher the number of grey levels the greater the dynamic range is available in the display of images. The maximum allowed maximum grey level is 235, but this will use all 256 levels of a "colormap" and therefore force XAS to use its own colormap. When the cursor enters the XAS window, the computer switches to that special colormap changing all of the other colors in the other windows (often in ways that are very undesirable). The default number of grey levels is 199 which may be small enough to avoid this effect or to manage to leave the colors of your most basic windows unaffected. Type HELP XAS ↵ when in AIPS to see how to control the number of levels, the colors of the graphics overlay planes, and numerous other parameters.

If the XAS window has closed for any reason, you may restart it without exiting AIPS with the verb STARTTV or TVRESTOR. If these do not work, exit AIPS and look for (and delete) left over LOK*, MSS*, TKS*, and XAS* files in /tmp. The TV server process named TVSRV*n* may also still be running and need to be killed.

2.4 Additional recipes

2.4.1 Banana daiquiri

1. Combine in an electric blender: 2 ounce **light rum**, 0.5 ounce **banana liqueur**, 0.5 ounce **lime juice**, 1/2 small **banana** peeled and coarsely chopped, and 1/2 cup crushed **ice**.
2. Blend at high speed until smooth.
3. Pour into large saucer champagne (or similar) glass. Serves one.

3 Basic *AIPS* Utilities

This chapter reviews some basic *AIPS* utilities with which you should be familiar before you start calibrating data or processing images in *AIPS*. Many of these utilities will appear in later chapters on calibration, image making, and so on. However, in those chapters, these utilities will be explained only briefly.

3.1 Talking to *AIPS*

3.1.1 *POPS* and *AIPS* utilities

When using the *AIPS* system, you talk to your computer through a command processor called *POPS* (for People Oriented Parsing System) that lives in the program `AIPS`. The steps needed to start this basic program are discussed in § 2.2.3. The copy of this program that you get will be called `AIPS`*n* where *n* is often referred to as the "*POPS* number" of your session.

The *POPS* command processor is not unique to *AIPS*. It has been present in other programs at the NRAO for many years, and will be familiar to users of the NRAO single-dish telescopes. Chapters 4 to 11 of this *CookBook* give explicit examples of most of the *POPS* commands that a new *AIPS* user needs to know, so we will not give a separate *POPS* tutorial here. The command HELP POPSYM ↵ will list the major *POPS* language features on your terminal, and Chapter 12 below reviews some advanced features of *POPS*.

As well as providing a command processor, *AIPS* replaces many features of your computer's operating system with its own utilities. This may seem inconvenient at first — you will have to learn the *AIPS* utilities as you go along. You will see the advantage of this approach when you use *AIPS* in a computer that has a different operating system. Your interface to *AIPS* will be almost identical on a VAX, or a Convex C-1, or a Unix-based workstation, or a Cray X-MP. Once learned, your *AIPS* skills will therefore be highly portable.

Lists of the important *AIPS* utilities can be obtained at your terminal by typing ABOUT CATALOG ↵ and ABOUT GENERAL ↵. See also Chapter 13 for a relatively recent version of all such category lists.

3.1.2 Tasks

AIPS provides a way for you to set up the parameters for, and then execute, many applications programs sequentially or in parallel. The more computationally intensive programs may take many minutes, hours (or even days) of CPU time to run to completion. They are therefore embodied in *AIPS* "tasks" — programs that are spawned by the `AIPS` program to execute independently and asynchronously (unless you choose to synchronize them). This lets you get on with other work in `AIPS`, while one or more tasks are running. You may spawn, however, only one copy at a time of each task from a given *AIPS* session (*i.e.*, *POPS* number.)

A typical task setup will look like:

> TASK ‘*task_name*‘ ↵	to make *task_name* the default for later commands; note the quote ‘ marks.
> HELP ↵	to write helpful text on your terminal about the purpose of the task and about its input parameters.

You will then spend some time setting up parameter values, as in § 3.1.4 below. Then, type

> INP ↵ to review the parameter values that you have set and
> GO ↵ to send the task into execution.

You may also specify which task you want to execute by an immediate argument, *e.g.*, GO UVSRT ↵ to execute the task UVSRT. After the GO step, you will watch for messages saying that the task has started executing normally, has found your data, etc., while you get on with other work in *AIPS*.

If you discover that you have started a task erroneously, you may stop it abruptly with

> ABORT ↵ to kill the task named by TASK, or
> ABORT *task_name* ↵ to kill *task_name*.

This will stop the job quickly and delete any standard scratch files produced by it. However, input data files — and output data files that are probably useless — may be left in a "busy" state in your data catalog. The catalog file is described in § 3.3, including methods to clear the "busy" states and to delete unwanted files. Note that you may also stop a task running under another *AIPS* number by setting adverb NUMTELL to that number.

Similarly, you may tell the current AIPS session to wait for the completion of a task before doing something else using

> WAIT ↵ to wait on the task named by TASK, or
> WAIT *task_name* ↵ to wait on *task_name*.

You may also wait on a task running under another *AIPS* number by setting adverb NUMTELL to that number. Note that AIPS will continue after the task finishes even if it finishes badly; see § 12.2.2.

The current full list of tasks may be obtained on your terminal (or workstation window) by typing ABOUT TASKS ↵. Since this list runs for many pages, you may wish to direct the output to the line printer (with DOCRT = -2 ↵) or to consult the list in Chapter 13 of this *CookBook*.

3.1.3 Verbs

Some of the smaller *AIPS* utilities run quickly enough to be run inside the AIPS program rather than being spawned. These "verbs" include simple arithmetic and *POPS* operations, the HELP, ABOUT, INP, and GO commands mentioned already, interactive manipulations of the TV-like display, and many more. Verbs are sent into action simply by setting their input parameters and typing the name of the verb followed by ↵. (The sequence GO *verb_name* ↵ will also work, but a bit more slowly since it also saves the input parameters of the verb for you; see § 3.5 for a further discussion of saved parameters. The sequence TASK '*verb_name*' ; GO ↵ will not work, however.) While a verb is executing, AIPS will not respond to anything you type on the terminal (but it will remember what you type for later use). Just watch out for messages and do what is called for with the TV cursor or terminal. You may, of course, think about what you will do next.

You can list all the verbs in *AIPS* on your terminal by typing HELP VERBS ↵, but the output lists only the names. To find out more, type ABOUT VERBS ↵ which describes what the verbs do. Since this output fills several pages, you may wish to direct it to the line printer (set parameter DOCRT to -2), or to consult the (perhaps dated) list printed in Chapter 13 of this *CookBook*.

3.1.4 Adverbs

AIPS uses "adverbs" (which may be real numbers or character strings, scalars or arrays) to pass parameters to both "verbs" and "tasks." A significant part of your personal time during an *AIPS* session will be spent setting adverbs to appropriate values, then executing the appropriate verbs or tasks. Examples of adverb-setting commands in *AIPS* are:

> CELL 0.5 C_R	to set a single scalar CELL to 0.5
> CELL 1/2 C_R	alternate for above with *POPS* in-line arithmetic
> IMSIZE 512,256 C_R	to set a two-element array IMSIZE to IMSIZE(1)=512, IMSIZE(2)=256
> IMSIZE 512 256 C_R	an alternate for the above if both values are positive
> IMSIZE 256+256,256 C_R	an alternate for the above using *POPS* in-line arithmetic
> UVWTFN 'NA' C_R	to set a string variable UVWTFN to the value NA
> LEVS 0 C_R	to set all elements of the 30-element array LEVS to zero
> LEVS = -2,-1,1,2,3,4,5 C_R	to set LEVS(1)=-2, LEVS(2)=-1, LEVS(3)=1, etc. The = avoids in-line arithmetic that would otherwise subtract 2 from LEVS(1)
> LEVS = -2,-1 1 2 3 4 5 C_R	an alternate for the above; the comma avoids in-line arithmetic that would otherwise set LEVS(1) = -3

Many *AIPS* tasks will assume sensible "default" values for adverbs that you choose not to (or forget to) specify. Some adverbs cannot be sensibly defaulted; these should be clearly indicated in the appropriate help information. You may review the current input parameters for any *AIPS* task or verb by typing

> INP C_R	to review the parameters for task TASK, or
> INP *task_name* C_R	to review the adverbs for task *task_name*.

Any adverbs which you have set to *à priori* unusable values will be followed on the next line by a row of asterisks and an informative message. Details of the input parameters used by any *AIPS* verb or task can be obtained on your terminal by typing:

> HELP C_R	to review the parameters for task TASK, or
> HELP *task_name* C_R	for task *task_name*
> HELP *verb_name* C_R	for verb *verb_name*
> HELP *adverb_name* C_R	for adverb *adverb_name*

See § 3.8 below for more methods of obtaining on-line help with *AIPS*.

You can list all the adverbs in *AIPS* on your terminal by typing HELP ADVERBS C_R, but the output lists only the names. To find out more, type ABOUT ADVERBS C_R which describes what the adverbs do. Since this output fills several pages, you may wish to direct it to the line printer (set parameter DOCRT to -2), or to consult the (perhaps dated) list printed in Chapter 13 of this *CookBook*.

3.2 Your *AIPS* message file

AIPS and all tasks talk back to you by writing messages to a disk file called the "message file" and/or by sending them to you on the appropriate "message monitor." Simple instructions and progress messages usually go only to the monitor; very few (if any) messages go only to the file. For AIPS itself, the message monitor is always the workstation window or terminal into which you are typing your commands. For the tasks, the monitor can also be a separate terminal (on well-equipped, but old, systems) or a second workstation window under control of the *AIPS* dæmon process MSGSRV. You can control whether or not you get the message server window by the setting of a Unix environment variable (AIPS_MSG_EMULATOR). Enter

> HELP MSGSRV ↵ for details.

At most *AIPS* installations, you get a message server by default. You may also control the size and appearance of the message server with parameters in the X-Windows .Xdefaults file. These parameters are also listed in by HELP MSGSRV ↵. If the message server window has quit, you can restart it with the verb STARTTV.

You may review the contents of the message file by typing PRTMSG ↵ at the > prompt at your terminal. PRTMSG is an example of an *AIPS* "verb" — it does not need a GO from you to execute, and it is *not* shed from your terminal. Each message in the file has, associated with the text, the time, task name, *POPS* number, and the priority of the message. The priority codes range from 0 for user input to 2 for "unimportant" messages to 5 for "answers" and other significant normal messages to 8 for serious error messages. The PRTMSG verb has adverbs to let you select either the printer or your window or terminal for the display and to let you control which messages will be displayed. For example, to set the minimum priority level for messages to be displayed, type:

> PRIORITY *np* ↵ where *np* is the desired minimum level,

before running PRTMSG; then only messages at this level or above will be listed on the printer or terminal. If *np* is ≤ 5, then messages at level 0 are also shown. PRTMSG has further adverbs to limit the output by program name (PRTASK, uses minimum match), message age (PRTIME as upper limit to the age), and *AIPS* number (PRNUM). Note that PRNUM must be your *AIPS*, *i.e.*, *POPS*, session number, not your user identification number. The choice of the output device is made with

> DOCRT -1 ↵ to select the line printer or a text file under control of OUTPRINT.

> DOCRT 1 ↵ to select the terminal at its current width ≥ 72 characters

> DOCRT *nc* ↵ to select the terminal at width *nc* characters: $72 \leq nc \leq 132$.

The wider you can make your window display, up to 132 characters, the more information *AIPS* can put on a line. You may change the line printer selection with PRINTER. If the line printer is selected (DOCRT=-1; OUTPRINT=''), PRTMSG will ask for permission to proceed if more than 400 lines will be printed.

PRTMSG does not delete messages from your message file. Use:

> CLRMSG ↵ to delete messages and to compress the message file.

CLRMSG supports adverbs like those of PRTMSG, except that the deletion is of messages older than PRTIME and the printing is of messages younger than PRTIME seconds ago. Old messages are automatically deleted from your message file when you EXIT (or KLEENEX)from AIPS. (The time limit for "old" messages is set by your local *AIPS* Manager. Usually, it is about 5 days.)

3.3 Your AIPS data catalog files

Your *uv* data sets and images are your largest inputs to, and outputs from, AIPS. A summary record of all your disk data sets (*uv* data, images, beams and temporary "scratch" data created by active tasks) is kept in your disk catalog files (one per disk). To interrogate this catalog file, use:

> INDI 0 ; MCAT ↵ to list all images on all disks, or
> INDI 0 ; UCAT ↵ to list all *uv* data sets on all disks, or
> SCAT ↵ to list all scratch files on all disks.

MCAT and UCAT list files based on the INDISK adverb, SCAT always lists all disks. M2CAT and U2CAT list files based on the IN2DISK adverb. M3CAT, U3CAT, M4CAT, U4CAT, OCAT, and UOCAT are similar for the IN3DISK, IN4DISK, and OUTDISK adverbs, respectively.

A complete listing of the catalog file, which may be printed with PRTMSG, can be generated by:

> CLRNAME ↵ to reset INNAME, INCLASS, INSEQ, INTYPE, and INDISK,
> CATALOG ↵ to generate the listing.

which will list all of your disk data sets. To limit the listing to a particular name, class, sequence number, type, and/or disk, use a combination of the adverbs INNAME, INCLASS, INSEQ, INTYPE, and INDISK. Verbs CAT2LOG, CAT3LOG, CAT4LOG, CAT5LOG, and CATOLOG are similar but use the matching set of input or output name adverbs. The INNAME and INCLASS type adverbs allow a rather powerful wild-card grammar; type HELP INNAME ↵ for details. Unless you want a hard copy, it is faster to use MCAT and UCAT, although they respond only to the INDISK adverb. A typical listing looks like:

```
CATALOG ON DISK  1
CAT USID MAPNAME      CLASS   SEQ  PT     LAST ACCESS      STAT
 18   76 3C166L5OK   .IIM001.    1 MA 27-OCT-1996 22:30:18
 19   76 3C166L5OK   .IBM001.    1 MA 27-OCT-1996 23:02:14
 22   76 3C166L5OK   .IIM001.    2 MA 28-OCT-1996 15:30:45
CATALOG ON DISK  2
CAT USID MAPNAME      CLASS   SEQ  PT     LAST ACCESS      STAT
 22   76 1200+519    .IIM001.    1 MA 01-NOV-1996 23:50:10
 23   76 1200+519    .IBM001.    1 MA 01-NOV-1996 23:59:58
 24   76 1200+519    .QIM001.    1 MA 28-OCT-1996 00:10:10
 25   76 1200+519    .UIM001.    1 MA 28-OCT-1996 00:19:19
 28   76 1200+519    .ICL001.    1 MA 02-NOV-1996 00:35:20 WRIT
 31   76 SCRATCH FILE.IMAGR1.    1 SC 02-NOV-1996 00:35:37 WRIT
 32   76 SCRATCH FILE.IMAGR1.    2 SC 02-NOV-1996 00:35:39 WRIT
CATALOG ON DISK  3
CAT USID MAPNAME      CLASS   SEQ  PT     LAST ACCESS      STAT
  2   76 3C138 A C   .UVSRT .    1 UV 22-OCT-1996 12:56:50
 36   76 1200+519    .UVXY  .    1 UV 02-NOV-1996 00:32:50 READ
 37   76 1200+519    .IMAGR .    1 UV 02-NOV-1996 00:34:25 WRIT
```

This user (identification number 76) has eight image files, three on disk 1 and six on disk 2. He also has two sorted *uv* data sets and an IMAGR *uv* work file on disk 3. There are two scratch (temporary) files on disk 2 which were created by IMAGR running out of AIPS1 (this determines their IMAGR1 classname). Image data files (images and beams) are distinguished by the type code MA. The *uv* data files are distinguished by the type code UV and scratch files by type SC.

Note that this user has encoded useful information other than the source name into the image file names on disk 1. These images were of 3C166 at L band with 50 kilo-wavelength (*uv*) taper. Such information is also carried in AIPS history files (see § 3.4 below), but it is often useful to place it at a level where CAT can see it. The user also gave the UVSRT file in slot 2 on disk 3 a name that encodes the source name (3C138), the VLA configuration (A), and the observing band (C). Careful choice of AIPS file names can save much

other bookkeeping. The file name can be any valid string up to 12 characters long. Also note how SEQ numbers distinguish different versions of a file with the same name; this and the global variables in *AIPS* are helpful features when doing iterative computations such as self-calibration.

3.3.1 Speedy data file selection

Each catalog entry has an identification number called the "catalog slot number". The CAT column at the left of the listing above shows these catalog numbers. They can be used to set up inputs quickly for *AIPS* programs that read cataloged disk data sets. Use:

> INDI *n1* ; GETN *ctn1* ↵ where *n1* selects the disk and *ctn1* is the catalog slot number.

The verb GETNAME (abbreviated through minimum match as GETN above) sets the adverbs INNAME, INCLASS, INSEQ, and INTYPE used by many tasks and verbs. Some tasks require a second, a third and even a fourth set of input image name adverbs. For these, use:

> IN2D *n2* ; GET2N *ctn2* ↵ to set the second set, and
> IN3D *n3* ; GET3N *ctn3* ↵ to set the third set.
> IN4D *n4* ; GET4N *ctn4* ↵ to set the fourth set.

The verb GETONAME (GETO for minimum match) sets the adverbs OUTNAME, OUTCLASS and OUTSEQ to those of a pre-existing output file. GETO is particularly useful with calibration tasks that copy extension tables (*e.g.*, CL or FG tables) from one database to another or for restarting an image deconvolution.

3.3.2 Catalog entry status

Note that several catalog slots on disks 2 and 3 in our sample catalog listing above do not have blank entries in the STAT column. This listing was made while the user was running a Clean deconvolution with IMAGR on the sorted *uv* data set in slot 36 — this *uv* data file is opened for READing. The Clean image file, ICL001 in slot 28, and the scratch and IMAGR files are opened for WRITing. Procedures that attempt to read files which are opened for writing, or vice versa, will be rejected with appropriate error messages. You must therefore note any non-blank entries in the STAT column carefully. In some situations (mainly involving system crashes or abortion of tasks [§ 3.1.2]) files may be left in READ or WRIT status indefinitely. If this happens, you may reset the file status with CLRSTAT ↵ after issuing the appropriate INDISK and GETNAME. Note that a WRIT status on a file which is not, in fact, being used at present probably indicates that the data in the file have been corrupted. Such files should usually be removed from your catalog by first clearing the file status with GETN *nn*; CLRST ↵ then deleting them with ZAP ↵.

Before using a data set as input to an *AIPS* task, check that the data set has a clear status. (It is possible to let two tasks read the same data at the same time, but this is not recommended as it will usually slow execution.) Also note the data set's disk number and its ordinal number in the catalog, as these are useful for GETN, GET2N, etc.

3.3.3 Renaming data files

Files may be renamed, after they have been cataloged, using the *AIPS* verb RENAME. Typical inputs might be:

> INDI 2 ; INNA '1200+519' ↵ to select disk 2 and set the input (old) name.
> INCL 'IIM001' ; INSEQ 1 ↵ to set the rest of the input name adverbs, *i.e.*, to select the file in slot 22 on disk 2 in the example above.

> OUTN '1200+51 15K' ; OUTSEQ 2 ↵	to set desired output name and sequence number.
> INP RENAME ↵	to review the inputs.
> RENAME ↵	to rename the I image to '1200+51 15K' and reset its sequence number to 2.

Two verbs can be used to alter the catalog numbers of files. RENUMBER moves a file to an empty, user-specified slot; a one-line command to do this would be SLOT *n*; RENUM ↵ where *n* is the new slot number. Note that *n* may now be higher than any slot numbers currently in use. RECAT compresses the catalog (*i.e.*, it removes gaps in the catalog numbers) without changing the order of the entries in the catalog.

3.3.4 Header listings

Every image or *uv* data set in *AIPS* has an associated header file that contains information needed to describe the data set in detail.

The header also contains information on the number of extension files of each type that have been associated with the data set. The most important file extensions that can be associated with *AIPS* image data are the HIstory file described below, the CC or Clean component files (see Chapter 5) and the PLot files and SLice files (see Chapter 6).

Multi-source *uv* data files may have many extensions (see Chapter 4). The most important are the HIstory file, the ANtennas file (subarray geometric data, date, frequency and polarization information, *etc.*), the BP (bandpass) file for bandpass calibration data, the CL (calibration) file for calibration and model information, the FQ (frequency) file for frequency offsets of the different IFs, the FG (flag) file for editing information, the NX (index) file (which assists rapid access to the data), the SN (solution) file for gain solutions from *AIPS* calibration routines, and the SU (source) file with source-specific information such as name, position, and velocity. Chapter 4 describes the use of these extensions in some detail.

You can list the header file of any catalog entry on your terminal by following the GETNAME step above with

> IMHEAD ↵	for a detailed listing, or
> QHEADER ↵	for a shorter listing.

The output of IMHEAD and QHEADER can also be printed using PRTMSG (at PRIORITY 2).

Output from IMHEAD on a multi-source *uv* data set might look like:

```
Image=3C345     (UV)         Filename=Z17G1_A     .MULTI .   1
Telescope=SBLNKGYO           Receiver=VLBI
Observer=FAP                 User #= 1353
Observ. date=27-FEB-1991     Map date=13-JUN-1995
# visibilities    112813     Sort order  TB
Rand axes: UU-L  VV-L  WW-L  BASELINE  TIME1  WEIGHT  SCALE
           SOURCE
-----------------------------------------------------------
Type    Pixels   Coord value  at Pixel    Coord incr   Rotat
COMPLEX      1   1.0000000E+00    1.00 1.0000000E+00    0.00
STOKES       4  -1.0000000E+00    1.00-1.0000000E+00    0.00
FREQ       128   2.2228990E+10   63.50 5.0000000E+05    0.00
RA           1    16 41 17.608    1.00      3600.000    0.00
DEC          1    39 54 10.820    1.00      3600.000    0.00
-----------------------------------------------------------
Maximum version number of extension files of type SU is   1
```

```
Maximum version number of extension files of type CL is   3
Maximum version number of extension files of type HI is   1
Maximum version number of extension files of type AN is   1
Maximum version number of extension files of type NX is   1
Maximum version number of extension files of type FG is   1
Maximum version number of extension files of type SN is   1
```

Output from IMHEAD on an image file might look like:

```
Image=3C219     (MA)         Filename=3C219-BC-6  .ICL001.   1
Telescope=VLA                Receiver=
Observer=BRID                User #=   76
Observ. date=06-SEP-1992     Map date=18-APR-1994
Minimum=-1.89720898E-04      Maximum= 5.05501366E-02 JY/BEAM
------------------------------------------------------------
Type    Pixels   Coord value  at Pixel    Coord incr   Rotat
RA---SIN   510    09 17 50.662  263.00     -0.300000    0.00
DEC--SIN   640    45 51 43.555  294.00      0.300000    0.00
FREQ         1   4.8726000E+09    1.00 2.5000000E+07    0.00
STOKES       1   1.0000000E+00    1.00 1.0000000E+00    0.00
------------------------------------------------------------
Map type=NORMAL              Number of iterations=   50000
Conv size=   1.40 X   1.40   Position angle=   0.00
Observed RA   09 17 50.600    DEC  45 51 44.00
Maximum version number of extension files of type HI is   1
Maximum version number of extension files of type PL is   5
Maximum version number of extension files of type SL is   1
```

Both QHEADER and IMHEAD list the maximum version numbers of the table extension files associated with a data set. Because you may acquire many versions of such tables during calibration, these verbs are often invoked during calibration in *AIPS*. Verbs IM2HEAD. IM3HEAD, IM4HEAD, and IMOHEAD, as well as Q2HEADER, Q3HEADER, Q4HEADER, and QOHEADER appeared to make comparable listings for the second, third, and fourth input and the output file naming adverb groups, respectively.

3.4 Your *AIPS* history files

Every *uv* and image file has an associated "history", or HI, file. This HI "extension" of the data set stores important information about the processing done so far on the data in the file. Every *AIPS* task and verb that alters either the data or the file header will record its key parameters in the history file. The history file is written to the disk file when you use FITS format, so you can preserve it for reference in later *AIPS* sessions or when sending data to colleagues.

In general, each "card" in the history file begins with the task or verb name. It then gives one or more of the input adverb values it used (*i.e.*, the defaults are filled in). All or parts of the file may be displayed on your terminal or printed on the line printer. For example, use:

> INDISK *n* ; GETN *ctn* ↵	to select the file to be displayed.
> PRTASK 'IMAGR' ↵	to examine only history information from IMAGR.
> DOCRT 1 ↵	to direct the display to your terminal, using its full width.
> PRTHI ↵	to print the IMAGR history.
> KEYSTRNG 'robust'; PRTHI ↵	to display only IMAGR history lines containing the string ROBUST.

> PRTASK ' '; DOCRT FALSE ↵ — to select all history cards and direct the output to the line printer.

> PRTHI ↵ — to print the full history file.

There are several (legitimate) reasons why you might wish to edit your history files. Repetitive self-calibration cycles, or image combinations, can lead to very long and very repetitive histories which could be substantially shortened with no real loss of information. Also some entries in the history file may become obsolete by, say, the deletion of plot files. The verb STALIN allows you to send a range of history lines to Siberian salt mines (*i.e.*, delete) by number with optional selection by task and, optionally, interactive confirmation of each deletion. You may, of course, simply wish to add information to the history file. The verb HINOTE can be used to append one line, given by the adverb COMMENT, or many lines, typed in interactively, to the history file. Even more powerfully, the verb HITEXT allows you to write your history file to an external text file (see § 3.10.1). You may edit that file with your favorite Unix file editor and then read it back, writing your edited file into any *AIPS* history file you want (with verb HINOTE).

3.5 Saving and restoring inputs

All input and output parameters ("adverbs") are global throughout AIPS. When an adverb value is specified for, or set by, a task or verb, it remains at that value for any other task or verb that uses an adverb of the same name (until you change it). This global nature of the *AIPS* adverbs is useful in most cases. It can, however, be inconvenient — especially if you are taken by surprise because you have not reviewed the adverb values before running a task. Before running any task or verb, check your current input adverbs carefully with:

> INP *name* ↵ — where *name* is the program name, or

> INPUTS *name* ↵ — to write the input values to the message file.

> QINP ↵ — to resume the previous INP or INPUTS with the page last displayed.

Some tasks have multiple pages of input parameters. QINP allows you to change a parameter on a page, review that page and then go on to the next page without having to view the first pages over again. Some verbs and a few tasks have *output* adverbs. Unless they are also used on input, they will not appear when you do INP or INPUTS. After running such verbs and tasks, do

> OUTPUTS *name* ↵ — to view the output values and write them to the message file.

To reset all adverbs for a particular task or verb to their initial values, without changing any other adverbs or procedures, enter

> DEFAULT *name* ↵ — to reset the values for *name*.

> DEFAULT ↵ — to reset the values for the verb or task named in the TASK adverb.

You can save all adverbs you have specified for *AIPS* to disk at any time by typing:

> SAVE *aaaaa* ↵ — where *aaaaa* is any string of up to 12 characters.

> GET *aaaaa* ↵ — will restore these inputs later.

These commands save or restore your entire AIPS "environment". For this reason, GET must be the only command on the input line; SAVE may appear with other commands, but will be executed before *any* of the other commands on the line. Thus, the sequence INNAME '3C123' ↵ INNAME 'BLLAC'; SAVE BLLAC ↵ will save a 3C123 environment, not a BLLAC one. AIPS automatically saves your environment in a disk area called LASTEXIT whenever you use the EXIT, KLEENEX, or RESTART commands. The command GET LASTEXIT is automatically executed whenever you start up the AIPS program again on the same machine. Thus, you retain your own *AIPS* environment from one use of AIPS to the next. To obtain a null version of the adverb values and of the rest of the *AIPS* environment, type:

> RESTORE 0 ↵

There is also one temporary area for saving your AIPS environment. To save your inputs temporarily, type:

> STORE 1 ↵ to save your inputs in area 1, and

> RESTORE 1 ↵ to recover the inputs you previously stored in area 1.

When new verbs and adverbs are created at your site, your old SAVE files will not know about them. Beginning with the 15JAN96 release, you may update the old files with the sequence:

> GET *aaaaa* ↵ to recover the old SAVE area.

> COMPRESS ↵ to get the new basic vocabularies without losing your adverb values and procedures.

> SAVE *aaaaa* ↵ to save the updated area for later; use the full name of the SAVE area here.

The list of SAVE areas may be reviewed with the verb SGINDEX. A SAVE area may be written as a RUN file (§ 3.10.2) if you first GET the area and then use SG2RUN.

The input adverb values associated with a task or a verb can be stored by the command:

> TPUT *name* ↵ where *name* is the verb, task, or procedure name.

and retrieved by the command:

> TGET *name* ↵

TPUT and TGET allow you to avoid, to some extent, the global nature of the adverb values in AIPS. This is sometimes advantageous. Whenever a task (or a verb, for that matter) is executed by the verb GO, TPUT runs automatically. TGET will therefore recover the last set of input adverbs used to execute the task, unless you deliberately overwrite them with a TPUT of your own. Note that *AIPS* will complain if you try to TGET input adverbs for a task for which no TPUT has previously been run (either manually or automatically). You must "put" before you can "get." TGINDEX will show you what tasks have been TPUT and when. VPUT, VGET, and VGINDEX allow you to save, recover, and list task-specific adverbs from up to 35 completely user-controlled storage areas. All TGET and VGET files have a new format which includes adverb names. Then, when the adverbs to a task or verb change, TGET and VGET will still work. In 31DEC23, the command TUGET will do a TGET but from a different user number. This may be convenient when the user uses multiple user numbers on his/her computer. In 31DEC24, TUGET was enhanced to read not only from the current user's *AIPS* disk 1, but also from another disk which was USERID's disk 1. New verbs TGUINDEX and SGUINDEX were written to list those tasks and SAVE/GET files, respectively, available from another user number and disk. New verb SGUGET allows you to GET a saved *POPS* environment from another user number and disk.

PLGET lets you recover the adverbs used to construct a user-selected plot file. As of 31DEC23, these are known by name. Older plot files depend on the list of adverbs having not changed. Check the INPUTS carefully after a PLGET.

You can change between versions of *AIPS* software once you are inside AIPS by typing

> VERSION '*version*' ↵ where *version* is one of OLD, NEW or TST

Alternatively, you may use this command to access a private version of a program in some other area — see § 12.2.2. Note that toggling between different versions of *AIPS* is possible only when the data formats are the same. Unfortunately, recent versions are not fully compatible with previous versions of *AIPS*. (The antenna file format changed in 2009, the internal *UV* format changed in 2015.) Note also, that you are toggling between different versions of tasks, not the verbs within the AIPS program. That version is selected when you start the program (§ 2.2.3) and can be changed only by exiting and start anew.

3.6 Monitoring disk space

Since the 15APR92 release of *AIPS*, the availability of data areas via NFS has vastly increased the amount of disk space accessible from a given *AIPS* session. The da= command line option to the aips command allows you to specify "disks" (data areas) from many hosts in addition to the current host, subject to a maximum of 35 disks per session. Note, however, that the BADDISK adverb has a limit of 10 disks. Thus, if more than 10 disks are accessed via NFS, you will not be able to prevent one or more from being used for scratch files. This can be important. Reading data over NFS is relatively efficient, but writing data is not. Even file creations (under Unix) require the writing of zeros to the whole file in order to guarantee later access to the requested space. Over NFS, this can be a slow process. For example, if user disk 1 is accessed via NFS, then every line of the message file must be written with NFS, a process which has been observed to require about one second of real time per message!

Another aspect of the new disk allocation system is a scheme by which the local *AIPS* Manager may restrict the availability of some disk areas to a set of user numbers, specified on a disk-by-disk basis. Managers usually use this tool to set aside most disks on a staff member's workstation for his/her sole use and to reserve space for visitors or other special projects on "public" workstations on a case-by-case basis. Use the FREESPAC verb within AIPS to show you the space used and available on all disks for your session and also to show whether or not that space is reserved. The right-most column of FREESPAC's output will show Alluser if the space is not reserved, Resrved if you are one of the users for which the space is reserved, Not you if you are not allowed to use the space, and Scratch if the space is to be used only for scratch files. Use FREESPAC often to keep track of how much space is available and where the space can be found.

Disk space is still generally at a premium. If more than one user has access to the disk areas you are using, or you use more than one user number to separate projects, then another useful tool for monitoring disks is the *AIPS* task called DISKU. To run it, type

> USER 32000 ; INDISK 0 ↵	to get all disks and users.
> BADDISK 0 ↵	to allow all disks; can be used to skip chosen disk numbers in 31DEC23.
> DOALL 0 ; GO DISKU ↵	to run the *AIPS* disk user task.

This will (eventually) list on the AIPS monitor (and the message file) the amount of data space in use by each user for all *AIPS* disks. Identify the worst disk hogs and apply appropriate peer pressure. If you are, mysteriously, the culprit on some disk, then

> USER 0 ; INDISK n ↵	where n is the mysteriously eaten disk
> DOALL m ; GO DISKU ↵	to run the job, listing all catalog entries requiring more than m Megabytes.

will give you the size of your larger files on the specified disk. Armed with this information, you may be able to take appropriate action upon your own data.

Sometimes the available disk space has been eaten up by *AIPS* scratch files that are no longer in use. Tasks that abort while executing (and other mysterious events) may produce this situation. To delete all your scratch files, except those for tasks which are still running, type:

> SPY ↵	to see which tasks are running.
> SCRD ↵	to delete the files.

SCRDEST is run automatically whenever EXIT, KLEENEX, RESTART, or ABORT *task_name* are executed. Note that the imaging and deconvolution tasks IMAGR, APCLN, and VTESS, the data editor TVFLG, the sorter UVSRT and other tasks may create large scratch and "work" files, so you should watch for "dead" copies of scratch and work files from these programs in your disk catalog. Both MCAT and UCAT will show scratch files as well as the requested file type, but SCAT will be easier to use. Note too that, if you are using more than one computer on a given disk area, only those scratch files created by your current computer will be deleted when you run the SCRD verb. Work files have to be deleted individually since they can be still of use after

the task which created them has finished.

Chapter 11 of this *CookBook* tells you how to backup or delete your own data to relieve disk crowding. At present, all other methods for managing disk space involve system-dependent commands of one sort or another. Since these may have unexpected consequences they are not recommended.

3.7 Moving and compressing files

Two *AIPS* tasks are frequently used to move files from one disk to another with options to reduce the file size. They are SUBIM, used on images, and UVCOP, used on *uv* data sets. SUBIM uses the adverbs BLC and TRC to select a portion of the input image and XINC and YINC to select a pixel increment through the portion. If these adverbs are defaulted (set to 0), the entire image is copied. Clean component, history, and other table extension files are copied as well, but plot and slice extensions are not. Similarly, UVCOP uses a wide range of adverbs to select which IFs, channels, frequency IDs, times, antennas, and sources are to be copied. If all of these adverbs are defaulted (set to 0 or blank), then all data are copied except (optionally) for completely flagged records. A flag table may also be applied to the data. including flag tables too large to be handled by most tasks. With extensive data editing, UVCOP may produce a rather smaller data set even with no other selection criteria. Antenna, gain, and other table extension files are copied, but plot files are not. The task MOVE may be used to copy all files associated with a catalog number (without modification) to another disk or to another user number.

3.8 Finding helpful information in *AIPS*

Much *AIPS* documentation can be displayed on your terminal by typing HELP *word* C_R, where *word* is the name of an *AIPS* verb, task or adverb. The information given will supplement that given in the INPUTS for a verb or task. It is the only source of information on the adverbs. Type XHELP *word* C_R to display the help file in your WWW browser with links to adverbs from task help files.

To print the HELP information on your line printer, set DOCRT = -1 and enter EXPLAIN *word* C_R instead. (Using DOCRT = 1 with EXPLAIN will send the output to your terminal screen.) For the more important verbs and tasks, EXPLAIN will print extra information, not shown by HELP about the use of the program, with detailed explanations, hints, cautions and examples.

HELP may also be used to list the names of all *POPS* symbols known to AIPS by category, an operation helpful when you can't remember the name of something. Type:

> HELP ADVERBS C_R	to get a list of all adverbs in the symbol table
> HELP ARRAYS C_R	to get a list of all array adverbs in the symbol table
> HELP REALS C_R	to get a list of all real adverbs in the symbol table
> HELP STRINGS C_R	to get a list of all character string adverbs in the symbol table
> HELP VERBS C_R	to get a list of all verbs, pseudo-verbs, and procedures in the symbol table
> HELP PSEUDOS C_R	to get a list of all pseudo verbs in the symbol table
> HELP PROCS C_R	to get a list of all procedures in the symbol table

In the past, *AIPS* contained a range of general HELP files which purported to list all verbs and tasks in various categories. Since these were maintained by hand, they were essentially never current and complete. That entire system has been replaced by the verbs ABOUT and APROPOS to be discussed below. A few general help files do remain, and they may even be relatively current. A list of these may be found by typing:

> HELP HELP ↵ for help on HELP.

A few general help files remain. They are POPSYM (symbols used in *POPS* interpretive language), NEWTASK (writing and incorporating a new task into *AIPS*), and PANIC (solutions to common problems).

The HELP verb is very useful, but only if you know that the function you want exists in *AIPS* and know its name. Two functions have appeared in AIPS to assist you in this search. The first of these, APROPOS, searches all of the one-line summaries and keywords of all *AIPS* help files for matches to one or more user-specified words. For example, type

> APROPOS CLEAN ↵ to display all keyword and 1-line summaries of help files containing words beginning with "clean" (in upper and/or lower case), and

> APROPOS 'UV PLOT' ↵ note the quote marks which are required if there are embedded blanks, or

> APROPOS UV,PLOT ↵ to display all keyword and 1-line summaries of help files containing *both* words beginning with "uv" and words beginning with "plot."

The text files used by APROPOS are maintained by the *AIPS* source code maintenance (check-out) system itself. As a result, they should always be current. Of course, the quality of the results depends on the quality of the programmer-typed one-line and keyword descriptions in the help files. These were not regarded as important previously and hence are of variable quality.

The second new method for finding things in *AIPS* is the verb ABOUT. Type

> ABOUT *keyword* ↵ to see a list of all *AIPS* tasks, verbs, adverbs, etc. which mention *keyword* as one of their "keywords."

You need only type as many letters of *keyword* as are needed for a unique match. The source-code maintenance system is used to force all help files to use only a limited list of primary and secondary keywords. Software tools to update the list files have also been written, and are used at least once with every *AIPS* release. The list of categories recognized is as follows (where only the upper-case letters shown in the name are actually used):

```
ADVERB        POPS symbol holding real or character data
ANALYSIS      Image processing, analysis, combination
AP            Tasks using the "array processor"
ASTROMETry    Accurate position and baseline measurements
BATCH         Running AIPS tasks in AIPS batch queues
CALIBRATion   Calibration of interferometer uv data
CATALOG       Dealing with the AIPS catalog file
COORDINAtes   Handling image coordinates, conversions
EDITING       Editing tables, uv and image data.
EXT-APPL      Access to extension files (tables)
FITS          FITS format for data interchange
GENERAL       General AIPS utilities
HARDCOPY      Creating listings and displays on paper
IMAGE-UTil    Utilities for handling images
IMAGE         Transforming of images
IMAGING       Creation of images: FFT, Clean, ...
INFORMATion   General lists and user help functions
INTERACTive   Functions requiring user interaction
MODELING      Model fitting to uv or image data
OBSOLETE      Functions slated for removal
ONED          Functions for one-dimensional image slices
OOP           Tasks coded with object oriented principles
OPTICAL       Functions of interest for optical astronomy data
```

```
PARAFORM      Skeleton tasks for use in building new tasks
PLOT          Displays of image and uv data
POLARIZAtion  Calibration, analysis, display of polarization
POPS          Aspects of the AIPS' user language POPS
PROCEDURe     Creation of and available procedures
PSEUDOVErb    Pseudo-verbs in the POPS language and AIPS
RUN           Creation of and available RUN files
SINGLEDIsh    Functions of interest for single-disk radio data
SPECTRAL      Functions for spectra-line and other 3D data
TABLE         AIPS table extension files
TAPE          Use of magnetic tapes (obsolete)
TASK          AIPS tasks - available asynchronous functions
TV-APPL       Tasks using the TV display
TV            Basic functions on the TV display
UTILITY       Basic functions on tables, uv and image data
UV            Functions dealing with interferometer uv data
VERB          Synchronous functions inside the AIPS program
VLA           Functions of particular interest for the VLA
VLBI          Functions of particular interest for very long
              baseline data.
```

A variety of synonyms are also recognized. Besides those that are merely spelling variants, the currently accepted synonyms are

```
FILES       ->  CATALOG          POSITION   ->  COORDINATES
FLAGGING    ->  EDITING          EXTENSION  ->  EXT-APPL
PRINTING    ->  HARDCOPY         PRINTER    ->  HARDCOPY
MAP         ->  IMAGE            MAP-UTIL   ->  IMAGE-UTIL
MAPPING     ->  IMAGING          LANGUAGE   ->  POPS
CUBE        ->  SPECTRAL         LINE       ->  SPECTRAL
VISIBILITY  ->  UV               VLBA       ->  VLBI
PARAMETERS  ->  ADVERB           HELPS      ->  INFORMATION
SLICE       ->  ONED
```

More detailed descriptions of new developments in *AIPS* can be found in the *AIPSLetter* published by the NRAO every six months and tied to each *AIPS* software release. It is available from the web site `https://www.aips.nrao.edu/` and is included with *AIPS* distributions. An *AIPS* Memo series is published by the NRAO with details of various aspects of the implementation of, and planning for, *AIPS*. Advanced users may also wish to receive, and contribute to, the *AIPS* electronic mail forum — `BANANAS`. There is also an electronic news group called `alt.sci.astro.aips` devoted to *AIPS* matters. This *AIPS CookBook*, many of the *AIPS* Memos, and various other publications of the *AIPS* group are available via anonymous `ftp` (at `ftp.aoc.nrao.edu`) and via the Internet and the "World-Wide Web" starting with "URL" (Universal Resource Location) `https://www.aips.nrao.edu/`).

Your local *AIPS* Manager probably receives the *AIPSLetter*, *AIPS* Memos, and `BANANAS` and can make information from them available at your site. He/she should also be aware of the electronic means of information retrieval, and be able to help you use them. If this is not the case, write to the *AIPS* Group (at NRAO, P. O. Box O, Socorro, NM 87801-0387) or send electronic mail to **daip@nrao.edu** for further information about these services.

3.9 Magnetic tapes

Large volumes of data were once brought into, and taken away from, AIPS using magnetic tape. Disk files are now much more frequently used; see § 3.10. Tape usage in AIPS is described here, in case you still have need of it, but the descriptions of tapes elsewhere in this CookBook have been greatly reduced. The tape drives assigned to you are displayed as you start up AIPS, *e.g.*,

```
Tape assignments:
Tape 1 is IBM 9-track model 9348-012 on LEMUR
Tape 2 is HP 9-track model 88780B on LEMUR
Tape 3 is IBM 7208/001 Exabyte 8200 (external) on LEMUR
Tape 4 is ZZYZX 1.3Gb DAT (left, Model# ZW/HT1420T-CC6) on LEMUR
Tape 5 is ZZYZX 1.3Gb DAT (right; both 150mb personality) on LEMUR
Tape 6 is IBM Exabyte 8200 (internal) on LEMUR
Tape 7 is REMOTE
Tape 8 is REMOTE
```

for the heavily loaded, and now obsolete, IBM called `lemur`. The tape numbers you see above correspond to AIPS adverb INTAPE values of 1, 2, 3, and so on. The description is meant to give you some idea of which box or slot is to receive your tape. Most of the drives will have a label on them identifying their AIPS tape number. If in doubt, ask a local guru for help. The last two tape "drives," called REMOTE, will be discussed separately below.

In case you forget this list, the verb TAPES will show it to you. TAPES is even capable of going out on the Internet and asking what devices are available to an AIPS user at the computer specified by the REMHOST adverb (if it is running TPMON)! Note however that instances of TPMON are no longer started automatically when AIPS is started.

3.9.1 Hardware tape mount

On some AIPS systems, tapes are handled by designated operators. Before mounting tapes, read Appendix Z (for NRAO sites) or obtain directions from your local AIPS Manager or operators for methods by which tapes are to be handled. Most AIPS systems, however, are on the self-service plan. In that case, the simplest thing to do is to find a drive of the required type without a tape in it. There is no way in most Unix systems (certainly not in Linux or Mac OS/X) of reserving a tape drive globally for your exclusive use, though once you have it MOUNTed from within AIPS, no other AIPS user can access it. It is most efficient to use a tape drive directly connected to your computer (and hence listed as you started up AIPS). However, any "AIPSable" drive will do. Mount the tape physically on the drive following the mounting instructions in Appendix Z or those posted at your installation for the particular kind of tape drive. For half-inch (nine-track) tapes, don't forget to insert a write ring if you intend to write on the tape or to remove any write ring if you intend only to read the tape. Exabyte and DAT tapes have a small slide in the edge of the tape which faces out which takes the place of the write ring of 9-track tapes. For 8mm (Exabyte) tapes push the slide to the right (color black shows) for writing and to the left (red or white shows) for reading. With 4mm DAT tapes, the slide also goes to the right for writing (but white or red shows) and to the left for reading (black shows). Note the identification number *m* marked on the drive you are using, as you will need to provide that number to the software for mounting and dismounting the tape and for executing AIPS tasks which read or write tape.

3.9.2 Software mounting local tapes

After you have the tape physically mounted on the tape drive, AIPS must also be told that you have done this and which tape drive you have chosen. This step is called a "software tape mount." It is necessary to wait until the mechanism in the drive has "settled down", *i.e.*, when the noises and flashing lights have stopped, before you can do the software mount. This operation is done from inside AIPS by typing:

> INTAPE *m* ↵ to specify the drive labeled *m*.
> DENSITY *dddd* ↵ to set the density to *dddd* bpi if needed.
> MOUNT ↵ to mount the tape in software.

Read any messages which appear on your terminal carefully since they report the success, failure, and/or limitations of the operation. The meaning of "density" with modern magnetic tape devices is mostly a matter of convention. With half-inch, 9-track tapes, AIPS understands the usual 800, 1600, and 6250 bytes per inch densities. A special value for density, 22500, is taken to mean high density (5-Gbyte) mode on 8mm (Exabyte) tapes. You must set the DENSITY adverb to one of these magic values, but in many cases it does not matter which one you use.

Please dismount the tape as soon as you are finished with it, using:

> INTAPE *n* ; DISMO ↵ to dismount a tape from the drive labeled *n*.

The dismount verb should cause the tape to be rewound and, in most cases, ejected from the drive. Please remove the tape from the tape drive promptly so that others may use the drive. Note that exiting AIPS under most circumstances — even with CTRL C — will cause your mounted tapes to be dismounted automatically.

3.9.3 Software mounting REMOTE tapes

On all AIPS systems, the last two tape drives are indicated as REMOTE. This means you can use two additional adverbs in AIPS to access tape drives on other computers. It doesn't matter where the computer is, as long as it's connected via Internet and has AIPS installed on it in the conventional way. For example, if you wanted to use AIPS tape drive 2 on remote host rhesus, you would type:

> REMHOST 'RHESUS' ; REMTAPE 2 ↵
> DENSITY *dddd* ↵ to set the density to *dddd* bpi if needed.
> INTAPE *n* ; MOUNT ↵ set local "tape" number and software mount

where *n* is the number of one of the REMOTE tape assignments in the list of tape drives you see on AIPS startup. If you know which computers are to provide remote tape services for you, it is a good idea to specify them when you start AIPS using the tp=*hostname* option (see § 2.2.3). In this way, you make certain that the AIPS dæmon tasks TPMON*n* which provide the remote service are running where they are needed. Note that this option is commented out of START_AIPS in 31DEC24.

3.9.4 Using tapes in AIPS

AIPS provides a number of basic tools for managing magnetic tapes. It is very helpful to have a list of the contents of magnetic tapes you intend to read. To list the contents of a tape on the line printer:

> TASK 'PRTTP' ; INP ↵ to review the inputs.
> NFILES 0 ↵ to list all files on the tape.
> PRTLEV 0 ↵ to list the image headers but not the details — both more and less detailed listings are available.
> DOCRT FALSE ↵ to print on the line printer.
> GO ↵ to run the task.

It is also a good idea to run PRTTP on your data tapes after you have written them, but before you have deleted the data from disk. PRTTP reads the the tape record by record to test for tape errors as well as to check the data format.

The AIPS program has a number of verbs to position and check magnetic tapes. These include

> REWIND ↵	to rewind the tape, *e.g.*, after running PRTTP.
> NFILES n ; AVFILE ↵	to advance the tape $n > 0$ file marks.
> NFILES -n ; AVFILE ↵	to move the tape backwards to the $n^{\rm th}$ previous file.
> NFILES 0 ; AVFILE ↵	to position the tape at the start of the current file.
> AVEOT ↵	to advance the tape to the end of information, usually for the purpose of adding more data at the end.
> TPHEAD ↵	to display the contents of the data file at the current tape position.

Users are encouraged to treat magnetic tapes with some caution. The tapes themselves can have — or develop — errors which render the data in the file unavailable. Furthermore, there are no generally accepted standards governing magnetic tape software in the industry. As a consequence, each Unix operating system handles them differently and each can change over time. This creates great difficulties in *AIPS* and may cause your version not to handle all tape devices in a fully compatible manner.

AIPS tasks are still able to handle tapes, should the need arise. Output tapes may be written with FITTP, FITAB, and TCOPY. Input tapes are read by IMLOD, UVLOD, FITLD, FILLM, M3TAR, MK3IN, and MK3TX.

3.10 *AIPS* external disk files

AIPS maintains a wide range of disk files for its own use internally. Unless you intend to write programs for *AIPS* you need not be concerned about their formats or, in many cases, even their existence. However, recent versions of *AIPS* also support "external" disk files to be read from and written to disk directories controlled by you. You may read and write from/to binary "FITS-disk" files with TPHEAD, UVLOD, IMLOD, FITLD, FITTP, and FITAB.. AIPS and some tasks also allow text files to be read or written from/to disk. For example, all print tasks can be instructed to append their output to user-specified text files. These can be examined later with an editor or written to tape with standard tape utilities. The two PostScript tasks, LWPLA and TVCPS, can be instructed to write their output plots in user-specified text files for later processing and, for example, inclusion in manuscripts. And AIPS itself can be instructed to take its input commands from user-created text files.

3.10.1 Disk text files

The most significant user control over external files is the specification of the file's full name, *i.e.*, its directory path and its name in that path. You specify the directory path by creating an environment variable ("logical name" in *AIPS*peak) *before* starting AIPS. The simplest way is to change directory (cd Unix utility) to the area you wish to use and enter

% setenv MYAREA 'pwd' ↵

where *MYAREA* is a logical name of your choosing (but all in *upper case*). Note that the pwd is surrounded by backward single quote marks. The grammar above is for users of c-shell and tc-shell. Users of korn, bourne, and bash shells would type:

$ MYAREA='pwd'; export MYAREA ↵

also with backward single quote marks. If you are going to read a text file into AIPS, its name must also be in upper-case letters. Finally, inside AIPS, you specify the file with, *e.g.*,

```
> OUTPRINT  = 'MYAREA:3C123.PRT' CR
```

where 3C123.PRT is any all upper-case file name of your choosing. Note the surrounding quote marks and the colon that separates the logical name and the file name portions. You may put the file anywhere under any name you choose, but we request that you put it an area owned by you, if you have one, or that you use an identifying name and a standard AIPS area set aside for the purpose. Files left around in the AIPS directories are subject to summary deletion. Be sure that AIPS has the privilege to write into your directory; use chmod to allow appropriate write privilege on the directory file (try to avoid world write!). On Unix systems, duplicate file names are not allowed and AIPS tasks will usually die when trying to write a file name that already exists. Print tasks will append to pre-existing files, however. The verb FILEZAP allows you to delete external files from inside AIPS. The name of the file to be deleted is given as an immediate argument.

File names may also be entered as complete path names, so long as they do not require more than 48 characters, the length of the adverb data values. Thus

```
INTEXT = '/home/primate2/egreisen/AIPS/Text.prt
```

Note that the trailing quote mark is left off and this is the last command on the input line so that the case is preserved.

Ordinary text files are used in AIPS for a variety of purposes. Every print task offers the option of saving the output in a file specified by OUTPRINT rather than immediately printing and discarding it. Similarly, output PostScript files from LWPLA and TVCPS may be saved in files specified by OUTFILE rather than immediately printing and discarding them. They may be used later in larger displays, or even enclosed as figures in a TEX document such as this *CookBook*. OUTTEXT is used by numerous other tasks, such as SLICE and IMEAN, to write output specific to the tasks which may be of use to other programs. AIPS tables may even be written as text files by task TBOUT, edited by the user, and then read back in by task TBIN. History files may be revised in a similar manner. Adverbs INFILE, INTEXT, CALIN, and INLIST may be used by a number of tasks to specify source models, lists of "star" positions, holography data, and the like. Television color tables are read from and written to disk text files specified with the OFMFILE adverb.

3.10.2 RUN files

RUN files are ordinary text files containing AIPS commands to be executed in sequence in a batch-like manner. They are often used to define procedures which you save in your own area or in an AIPS-provided public area with the logical name $RUNFIL. The name of the file must be all upper case letters, followed by a period, followed by your user number as a three-digit "extended-hexadecimal" number with leading zeros. (To translate between decimal and extended hexadecimal, use the AIPS procedures or the AIPS verbs called EHEX and REHEX.) The files are edited from Unix level using emacs, vi, textedit or your other preferred text editor. For example, log in to the aips (or your own) account. From Unix level, type:

```
% cd  $RUNFIL                          to change to RUN area.
% emacs  MAPIT.03D CR
```

to edit with emacs a file called MAPIT for user 121. You may now also use any area of your choosing instead of the public $RUNFIL area. For instructions on the individual editors, consult the appropriate Unix Manuals. Instruction manuals for the GNU emacs editor are available from local computer staff. A SAVE area (§ 3.5) may be written as a RUN file if you first GET the area and then use SG2RUN.

To use the RUN file, define a logical name as in the previous Section. Then start up AIPS under your user number and enter

```
> VERSION  = 'MYAREA' CR                where MYAREA is your disk area, or
```

> VERSION = ' ' ↵ if $RUNFIL is to be used
> RUN *FILE* ↵ to execute the file named *FILE.uuu*

where *uuu* is your user number if extended hexadecimal with leading zeros to make three digits.

3.10.3 FITS-disk files

FITS is an IAU-endorsed binary format standard for astronomical data heavily used by *AIPS* for almost all of its data on disk and magnetic tape. In fact, it is the only format written by *AIPS* except for simple tape copying. The basic FITS paper (by Wells, Greisen, and Harten) appeared in *Astronomy & Astrophysics Supplement Series*, Volume 44, pages 363–374, 1981. The newsgroup `sci.astro.fits` is devoted to discussion of FITS. World-wide web users can access the FITS home page at

`https://fits.gsfc.nasa.gov/fits_home.html`

AIPS also supports the FITS format written to disk in exactly the same form as it is written to magnetic tape. The tasks FITTP and FITAB may be instructed to write their output files on disk rather than on tape. Likewise, TPHEAD, FITLD, UVLOD, IMLOD, and PRTTP can read from disk. To write to a FITS-disk file, specify:

> DATAOUT *'filename'* ↵ where *filename* is the name of the desired output file.

and to read from a FITS-disk file, you specify:

> DATAIN *'filename'* ↵

where you must specify *filename* with environment variables ("logical names" in *AIPS*peak), *e.g.*,

> DATAOUT = 'MYDATA:3C123.FIT' ↵

in exactly the same way as described for text files in § 3.10.1. There is a standard public area, called logically FITS, which you may use for reading and writing FITS-disk files. FITTP will use this area if you do not specify a logical name. Be aware that older files will be purged from this public area when space is needed. Note too that FITTP will write only one disk file per execution; the DOALL option is disabled when writing to disk.

There is a package of procedures to assist in writing and reading more than one FITS-disk file at a time. Enter RUN WRTPROCS to define the procedures. The procedure FITDISK will write a single disk catalog file to a disk file using a name based on the *AIPS* file name parameters. You may then construct loops invoking FITDISK to write multiple files. For example:

> FOR I=1:10; GETN(I); FITDISK; END ↵

Such file names are useful for their mnemonic content, but must be read back one at a time. The procedure WRTDISK will dump a range of catalog numbers to disk under names that allow the procedure READISK to read them back as a group. These two procedures are particularly useful when moving your data between computer architectures (*e.g.*, from a Solaris to a Linux computer). Note that there is a stand-alone program REBYTE which can do the conversion directly including files of all types.

FITLD can read multiple disk files in either the normal FITS format (as written by FITTP) or the special FITS format written by the VLBA correlator. The only requirement for this operation is that file names end in sequential numbers beginning with 1. FITAB has the ability to write special FITS files with visibility data in tables. These files may be broken up into multiple files, called "pieces," for size and reliability considerations. These pieces, when written to disk, have names ending in sequential numbers. Special code in FITLD and UVLOD recognize these pieces and read the requested number of them as if they were in one file.

Remote FITS-disk files may be read in much the same manner as remote magnetic tapes. Type HELP DATAIN ↵ or HELP DATAOUT ↵ for details.

FITS-disk files are written as Fortran files and hence are available also to user-coded programs. The Fortran specifications for the file are `ACCESS='DIRECT', RECL=2880, FORM='UNFORMATTED'` in the `OPEN` statement

for Unix systems. Most Fortrans cannot read or write files larger than 2 Gigabytes, so *AIPS* now reads and writes these files with C subroutines. Users may also, of course, code programs to create such files to be read by FITLD, IMLOD or UVLOD. Consult *GOING AIPS*, Volume 2, Chapter 13 for details on how to do this.

One of the main uses for FITS-disk files is to transfer data over the Internet between computers. For example, to transfer a file from `rhesus` (in Charlottesville) to `kiowa` (at the AOC), log in to `rhesus`, change to the directory in which you wish to store the file (for example, cd $FITS ↵), and enter:

% ftp kiowa ↵	to start `ftp` to the remote system.
`Name (kiowa:...):` *loginame* ↵	to log in to account *loginame*.
`Password:` *password* ↵	to give the account's password.
`ftp>` cd *directory* ↵	to change to the *directory* name containing the file.
`ftp>` binary ↵	to allow reading of a binary file.
`ftp>` hash ↵	to get progress symbols as the copy proceeds.
`ftp>` put *filename* ↵	to send the file
`ftp>` quit ↵	to exit from `ftp`.

The file should then be in the desired directory. You may have to rename it, however, to a name in all upper-case letters since that may be required by *AIPS*. (See § 3.10.1 for a trick that allows you to use lower-case letters in file names.) The file format will be correct. In general it is better to use the ftp program to "get" files instead of "put"ting them; things tend to go faster that way.

An alternative to using `ftp` is to use the `scp` (remote secure copy) Unix utility to write the output file directory in the appropriate area on the other computer. The syntax for the remote copy is:

% scp $FITS/MYFILE.FITS *username*@kiowa:/AIPS/FITS/MYFILE ↵

(this shows how you would copy it from `rhesus` to `kiowa`).

If you wish to copy a FITS-disk file from one machine to another within a site, check if you can just use the Unix `cp` command; this is often possible if the remote disk is mounted (or can be auto-mounted) via NFS (the Network File System).

FITS files may be compressed with standard utility programs such as `gzip`. This does not produce much compression for files written with full dynamic range and floating-point format. However, FITAB offers the option of writing images (not *uv* data) which are quantized at some suitable level. These are capable of significant compression even if they are in floating-point format.

3.10.4 Other binary data disk files

Data written by the on-line system of the VLA are now often found in disk files rather than on tape. These data are available from an archive of all VLA data. See

`https://data.nrao.edu/`

for information on how to access your current data and all data for which the proprietary period has expired. FILLM and PRTTP can read the disk files produced from the archive, including reading more than one such file in a single execution. In this case, the file names must end in consecutive numbers beginning with NFILES + 1.

3.11 The array processor

In running numerous important tasks in AIPS you will notice references to an "array processor." This used to be an expensive device attached to computers which allowed them to run some AIPS tasks 100 times faster than they would run without the device. To support our less fortunate colleagues, we wrote a software emulation of the array processor which we call the pseudo array processor. This uses highly optimized routines running on data stored in the "AP memory." By now, the hardware devices are all gone and only the software emulation remains. We have found that this is still a good model to obtain highly optimized software performance.

The pseudo-AP now uses dynamic memory allocated by the calling task as needed rather than a fixed amount of memory that would be much too large for some problems and too small for others. The largest AIPS tasks, including IMAGR and CALIB in self-cal mode, have changed to use rather large amounts of memory if needed to reduce the number of times the visibility data need to be read. Two new verbs appeared at this time: SIZEFILE returns the size of a disk file and SETMAXAP sets the maximum computer memory that may be used for the pseudo-AP. The latter allows the user to limit the AP on smaller or busier machines or to permit really large memory usage when large amounts of memory are available. See HELP SETMAXAP for a discussion of the essential considerations.

The pseudo-AP has been changed to use only double precision integers and floats. This avoids serious overflow issues with the large number of data samples from modern telescopes and provides much greater accuracy especially in the gridding and Fourier transforming of visibility data.

3.12 Additional recipes

3.12.1 Columbian fresh banana cake with sea foam frosting

1. Open an 18.5 oz **yellow cake mix** into a large mixxing bowl; do not use a mix that contains pudding or requires oil. Combine with 1/8 teaspoon **baking soda**.
2. Stir 3/4 cup **Coca-Cola** briskly until foaming stops. Add to cake mix and blend until just moistened. Then beat at high speed for 3 minutes, scraping the bowl often.
3. Combine 2 teaspoons **lemon juice** with 1 cup mashed **bananas** and then add to batter. Add 1/3 cup finely chopped **nuts** and beat for 1 minute at medium speed.
4. Turn the batter into a well greased, lightly floured 9x13 baking dish. Bake in a preheated 350° F oven for about 40 minutes or until the cake tests done. Cool on a rack for 15 minutes, remove cake from pan and turn right side up on a rack to finish cooling.
5. In the top of a double boiler, combine 2 large **egg whites**, 1 1/2 cups packed **light brown sugar**, 1/8 teaspoon **cream of tartar** (or 1 tablespoon corn syrup), and 1/3 cup **Coca-Cola**. Beat at high speed for 1 minute with an electric mixer.
6. Place over boiling water — the water should not touch the bottom of the top half of the double boiler. Beat on high speed for about 7 minutes until the frosting forms peaks when the mixer is raised. Remove from boiling water. Empty into a large bowl.
7. Add 1 teaspoon **vanilla extract** and continue beating on high speed until thick enough to spread, about 2 minutes. Spread on the sides and top of the cold banana cake.

3.12.2 Banana sweet potato puff casserole

1. In a large bowl, combine 2 cups mashed **sweet potatoes**, 1 cup mashed ripe **bananas** (3 medium), 3/4 teaspoon **curry powder**, 1/3 cup **sour cream**, 1/2 teaspoon **salt**, and 1 **egg**.
2. Beat with electric mixer until light and very fluffy. Turn into 1 quart casserole dish.
3. Bake at 350° F for 20 minutes or until puffed and lightly browned.

Thanks to Turbana Corporation (`www.turbana.com`).

3.12.3 Banana poundcake

1. Mix in large bowl until blended:
   - $1\frac{1}{3}$ cups mashed **bananas** (4 medium)
   - 1 pkg. ($18\frac{1}{2}$ oz.) **yellow cake mix**
   - 1 pkg. ($3\frac{3}{4}$ oz.) instant **vanilla pudding mix**
   - $\frac{1}{3}$ cup **salad oil**
   - $\frac{1}{2}$ cup **water**
   - $\frac{1}{2}$ teaspoon **cinnamon**
   - $\frac{1}{2}$ teaspoon **nutmeg**
   - 4 **eggs** at room temperature
2. Beat at medium speed for 4 minutes.
3. Turn batter into greased and lightly floured 10-inch tube pan.
4. Bake in 350° F oven for 1 hour or until cake tester inserted in cake comes out clean.
5. Cool in pan 10 minutes, then turn out onto rack and cool completely.
6. If desired, dust with confectioners sugar before serving.

Thanks to the United Fresh Fruit and Vegetable Association.

3.12.4 Banana July cocktail

1. Sprinkle 3 sliced **bananas** with 1 tablespoon **lemon juice**.
2. Mix with 1 1/4 cans drained and flaked **tuna**, 1/2 **onion** chopped, and 2 tablespoons chopped **gherkins** or **olives**.
3. Spoon into 7 cocktail shells.
4. Melt 2 tablespoons **butter** in a saucepan. Add 2 tablespoon **cake flour** and salt and pepper to taste.
5. Add 1/4 cup **chicken stock** and 1/4 cup dry **white wine**. Simmer for one minute stirring constantly.
6. Add 1/3 cup grated **cheddar cheese** and allow to cool.
7. Add 1/4 cup fresh cream to sauce and pour over banana-tuna mixture.
8. Sprinkle with 1 tablespoon grated **cheese** and **paprika**. Decorate with a slice of **gherkin** or **olive**.
9. Bake 15–20 minutes at 350° F; serve warm.

Thanks to Turbana Corporation (`www.turbana.com`).

4 Calibrating Interferometer Data

This chapter focuses on ways to do the initial calibration of interferometric fringe-visibility data in *AIPS*. The sections which follow concentrate primarily on calibration for connected-element interferometers, especially the Karl G. Jansky very Large Array, called the EVLA hereafter. However, the information in these sections is useful for data from the historic VLA and other interferometers and to spectral-line, solar, and VLBI observers as well. For specific advice on the historic VLA, consult Appendix O. while the original appendix on EVLA data reduction has been deleted. For the gory details of VLBI, read Chapter 9. After the initial calibration has been completed, data for sources with good signal-to-noise are often taken through a number of cycles of imaging with self-calibration. See § 5.4 for information on these later stages of the reduction process. For accurate calibration, you must have accurate *a priori* positions and structural information for all your calibration sources and accurate flux densities for at least one of them. It is best if the calibration sources are unresolved "point" sources, but it is not required.

For the basic calibrations, visibility ("*uv*") data are kept in "multi-source data sets," each of which contains, in time order, visibility data for one or more "unknown" sources and one or more calibration sources. Associated with these data are "extension" files containing tables describing these data. When VLA archive data are first read into *AIPS* a number of basic tables are created and filled with information describing the data set. A complete set of these would include

1. `AN` (antennas) for sub-array geometric data, date, frequency, polarization information, *etc.*,
2. `CD` (CalDevice) for noise tube values,
3. `CL` (calibration) for calibration and model information,
4. `CT` (Calc) for astrometric data used in the correlation,
5. `FG` (flag) for flagging (editing) information,
6. `FQ` (frequency) for frequency offsets of the different IFs (IF pairs in VLA nomenclature),
7. `GC` (gain curve) for nominal sensitivity and antenna gain functions,
8. `HI` (history) for history records,
9. `NX` (index) to assist rapid access to the data,
10. `SU` (source) for source specific information such as name, position, velocity,
11. `SY` (SysPower) for system gains and measured total power with the noise tubes on and with them off, and
12. `WX` (Weather) for weather data.

An initial `CL` table contains gains due to known antenna functions of elevation and measured atmospheric opacities. VLBI, and especially VLBA, data sets will end up with even more table files. Calibration and editing tasks then create, as needed, other tables including

13. `BL` (baseline) for baseline-, or correlator-, dependent corrections,
14. `BP` (bandpass) for bandpass calibration,
15. `PD` (Polarization) for spectral dependent polarization D terms,

16. SN (solution) for gain solutions from the calibration routines.

All of these tables can be written to FITS files along with the visibility data using FITTP or FITAB. They can be read back in with FITLD. These, and any other, *AIPS* tables can be manipulated and examined using the general tasks PRTAB, TACOP, TABED, TAMRG, TASRT, TAFLG and TAPPE.

The visibility data within the multi-source data set are not normally altered by the calibration tasks. Instead, these tasks manipulate the tabular information to describe the calibration corrections to be applied to the data and any flagging (deletion) of the data.

The *AIPS* programs discussed in this chapter are part of a package that has been developed to calibrate interferometer data from a wide range of connected-element and VLB arrays, especially the VLA and VLBA. These programs therefore support many functions (and inputs) that are not required when calibrating normal VLA data. The examples given below show only the essential parameters for the operation being described, but, to get the results described, it is essential that you check *all* the input parameters before running any task. Remember that *AIPS* adverbs are global and will be "remembered" as you proceed. A list of calibration-related symbols is given in § 13.6, but a possibly more up-to-date list can be obtained by typing ABOUT CALIBRAT in your *AIPS* session. More general information on calibration can be routed to your printer by typing DOCRT FALSE ; EXPLAIN CALIBRAT ↵, while deeper information on a specific task is obtained with EXPLAIN *taskname* ↵.

When you are satisfied with the calibration and editing (or are simply exhausted), the task SPLIT is used to apply the calibration and editing tables and to write *uv* files, each containing the data for only one source. These "single-source" *uv* files are used by imaging and deconvolution tasks that work with only one source at a time. Many of the tasks described in this chapter will also work on single-source files. For VLA calibration, there are several useful procedures described in this chapter and contained in the RUN file called VLAPROCS. Each of these procedures has an associated HELP file and inputs. Before any of these procedures can be used, this RUN file must be invoked with:

> RUN VLAPROCS ↵ to compile the procedures.

There is a "pipeline" procedure designed to do a preliminary calibration and imaging of ordinary VLA data sets. This provides a good first look at the data. Nonetheless, the results are still not likely to be of publishable quality. To run the pipeline, enter

> RUN PIPEAIPS ↵ to compile the procedures.
> INP PIPEAIPS ↵ to review the input adverbs and, when ready,
> PIPEAIPS ↵ to execute the pipeline.

4.1 Copying data into *AIPS* multi-source disk files

There are several ways to write VLA data to *AIPS* multi-source *uv* data sets on disk. They include:

1. For data from the historic VLA, download the files from the archive and run FILLM to load the data. See § O.1 for the details.
2. For data from the new EVLA, download the files from the archive and run BDF2AIPS to load the data. See § 4.1.1 below for the details.
3. For an *AIPS* multi-source data set written to a FITS tape or disk file during an earlier *AIPS* session, use UVLOD or FITLD to read the tape.
4. Single-source data sets that are already on disk may be combined with multiple executions of UV2MS or, if the files are dissimilar, with multiple executions of MULTI followed by DBCON.

5. Data from the Australia Telescope may be loaded from disk files into AIPS using the task ATLOD which is now included with AIPS.

Data from other telescopes can be read into AIPS only if they are written in AIPS-like FITS files already or if you have a special format-translation program for that telescope. The VLBA correlator produces a format which is translated by the standard AIPS task FITLD; see § 4.1.2. A translation task for the Westerbork Synthesis Telescope (WSLOD) is available from the Dutch, but is not distributed by the NRAO with the normal AIPS system.

4.1.1 Reading EVLA archive files into AIPS

The NRAO Archive makes available, among other things, data from the EVLA, which began observing in January 2010. Your EVLA data are stored as an "Science Data Model" (SDM) format file in "SDMBDF" (Science Data Model Binary Data Format) in the NRAO archive. They may be read out of the archive in that format or a CASA measurement set format. Go to the web page

`https://data.nrao.edu/`

and log in with your `my.nrao.edu` account. Select Show Search Inputs to request the data. Fill out enough of the form to describe your data and submit the query. If the data are not yet public, your log in will tell the archive if you are entitled to access particular locked projects. The query will return a list of the data sets which meet your specifications. On this form, enter your e-mail address. If you have disk access at NRAO Socorro and have a world-writable directory, enter that directory as the download destination. Otherwise let the download go to dthe default location and be sure to uncheck the tar file button. Select the SDM-BDF dataset format if you wish to reduce the data fully in AIPS. If you select the CASA MS format, you will be required to use CASA either for all your work or at least to convert the MS format to a UVFITS format which AIPS can read. Sadly, such UVFITS files lack a lot of useful information and so they are not recommended. The archive tool will notify you, by e-mail, when your data set is available and will provide information on how to retrieve the data. Your file will be deleted after 2 days, so you must copy the file to a data directory that you control.

You may now run AIPS and load the data via the verbs BDFLIST and BDF2AIPS. These verbs run programs in the OBIT software package to load your data directly into AIPS including flag (FG), index (NX), calibration (CL), over-the-top (OT), SysPower (SY), CalDevice (CD) and Planet (PO) tables which you will not get from CASA. Note that these verbs require that OBIT be installed on your computer — as it is in Socorro — and that ObitTalk be in your $PATH. OBIT is relatively easy to install and may be obtained from `www.cv.nrao.edu/~bcotton/Obit.html`

SDMBDF files may be read into AIPS using BDFLIST to learn what is in your data set and then BDF2AIPS to translate the data. Thus

> DEFAULT BDF2AIPS; INP ↵	to initialize all relevant adverbs.
> DOWAIT 2 ; DOCRT 1 ↵	to wait for the verbs to finish and to display the log file on the terminal after the OBIT task finishes. DOWAIT 1 displays the messages as they are generated but is insensitive to returned error conditions from the OBIT tasks. Be sure to set DOWAIT -1 after using BDF2AIPS.
> ASDMF(1) = '*path_to_asdm_dir* ↵	to set the path name to your data directory into the adverb. Note the lack of close quote so that case is preserved.
> ASDMF(2) = '*asdm_file_name* ↵	to put the rest of the data file name in the second adverb since the names are almost always longer than 64 characters. Trailing blanks in ASDMF(1) will be ignored.
> BDFLIST ↵	to list the contents of the SDMBDF. Note particularly the "configuration" numbers.

> OUTNA *'myname'* CR — to set the *AIPS* name.
> OUTCL '' CR — to take default (UVEVLA) class.
> OUTDI 3 CR — to write the data to disk 3 (one with enough space).
> DOUVCOMP FALSE CR — to write visibilities in uncompressed format. There are no weights at present, so there is no loss of information in compressed format, but the conversion from compressed format costs more than reading the larger data files.
> FOR CONFIG = 0:100 ; BDF2AIPS; END CR — to load all of the configurations in your data, terminating with error messages on the first configuration number not present in your data (when DOWAIT is 2).

There are other adverbs — NCHAN, NIF, BAND, CALCODE, and DOSTOKES — available if needed to limit which data are read. CONFIG is frequently all that is needed to select data, but these others may be needed if more complicated modes of observing were used. If you running Python 3 rather than 2, set OBITVERS to 3.

If BDF2AIPS is executing correctly, your message terminal will report information about the data being loaded including the data selection, the data scans loaded, and information about some of the tables written. Once BDF2AIPS has completed, you can find the database on disk using:

> INDI 0 ; UCAT CR — to list all cataloged *uv* files

This should produce a listing such as:

```
Catalog on disk  3
Cat Usid Mapname      Class  Seq  Pt     Last access      Stat
  1  103 MyName      .UVDATA.    1 UV 05-FEB-2017 12:34:16
```

You might then examine the header information for the disk data set by:

> INDI 3 ; GETN 1 ; IMHEAD CR — set name adverbs, list header in detail.

This should produce a listing like:

```
AIPS 1: Image=MULTI     (UV)         Filename=MyName      .UVDATA.   1
AIPS 1: Telescope=EVLA               Receiver=EVLA
AIPS 1: Observer=Dr. Crys            User #=  200
AIPS 1: Observ. date=29-OCT-2016     Map date=29-JAN-2018
AIPS 1: # visibilities    876000     Sort order  TB
AIPS 1: Rand axes: UU-L-SIN  VV-L-SIN  WW-L-SIN  BASELINE  TIME1
AIPS 1:            SOURCE  INTTIM  WEIGHT  SCALE
AIPS 1: ----------------------------------------------------------------
AIPS 1: Type    Pixels   Coord value     at Pixel     Coord incr   Rotat
AIPS 1: COMPLEX      1   1.0000000E+00       1.00  1.0000000E+00    0.00
AIPS 1: STOKES       2  -1.0000000E+00       1.00 -1.0000000E+00    0.00
AIPS 1: FREQ       128   4.0393400E+09      65.00  1.0000000E+06    0.00
AIPS 1: IF          31   1.0000000E+00       1.00  1.0000000E+00    0.00
AIPS 1: RA           1    00 00 00.000       1.00       3600.000    0.00
AIPS 1: DEC          1    00 00 00.000       1.00       3600.000    0.00
AIPS 1: ----------------------------------------------------------------
AIPS 1: Coordinate equinox 2000.00
AIPS 1: Maximum version number of extension files of type SU is   1
AIPS 1: Maximum version number of extension files of type AN is   1
AIPS 1: Maximum version number of extension files of type FQ is   1
AIPS 1: Maximum version number of extension files of type NX is   1
AIPS 1: Maximum version number of extension files of type CD is   1
AIPS 1: Maximum version number of extension files of type SY is   1
AIPS 1: Maximum version number of extension files of type GC is   1
AIPS 1: Maximum version number of extension files of type CT is   1
```

```
AIPS 1: Maximum version number of extension files of type FG is   1
AIPS 1: Maximum version number of extension files of type WX is   1
AIPS 1: Maximum version number of extension files of type CL is   1
AIPS 1: Maximum version number of extension files of type HI is   1
```

This header identifies the file as a multi-source data set (Image=MULTI) with 876000 floating-point visibilities in time-baseline (TB) order. There are 128 spectral channels in each of 31 entries on the IF axis. These correspond to "spectral windows" in the correlator and have, at best, only a loose correspondence to the separate electronic channels. The description of the frequency (FREQ) axis shows that the first IF is at 4039 MHz and has 128 MHz total bandwidth. The parameters of the other IFs are determined from the data in the FQ table file and cannot be read directly from this header; these values are shown in the 'SCAN' listing from LISTR. In 31DEC23, task LISUN may be used as an alternative to LISTR SCAN operation. It shows the angular distance to the Sun of each scan and a summary of the antennas used in addition to all of the standard LISTR output. The header shown above indicates that the data are in compressed format since the number of pixels on the COMPLEX axis is 1 and the WEIGHT and SCALE random parameters are present. Uncompressed data does not use these random parameters and has 3 pixels on the COMPLEX axis.

Complex VLA observations may have the spectral windows in a strange frequency order. BDF2AIPS with ORDER=1 can rearrange them. If the groups of windows overlap, it may be best to rearrange them after calibration (particularly FRING which can use groups of adjacent windows) using the new task FIXFR.

If your experiment contains data from several bands BDF2AIPS will place the data from each band in separate data sets. Also, if you observed with several sets of frequencies or bandwidths in a given observing run these will be assigned different FQ numbers by BDF2AIPS. You can determine which frequencies correspond to which FQ numbers from the 'SCAN' listing provided by LISTR or LISUN. If you find multiple FQ numbers in your data set, we strongly advise you to run UVCOP to separate them into different files. This will greatly simplify your data reduction.

4.1.2 Reading data from FITS files with FITLD

FITLD is used to read FITS-format disk files (and tapes) into *AIPS*. It recognizes images, single- and multi-source *uv* data sets, and the special FITS *uv*-data tables produced by the VLBA and DiFX correlators ("FITS-IDI" format). In particular, VLA data sets that have been read into *AIPS* previously with FILLM and then saved to tape (or pseudo-tape disk) files with FITTP and FITAB can be recovered for further processing with task FITLD. (The older task UVLOD will also work with *uv* data sets in FITS format, but it cannot handle image or FITS-IDI format files.)

A multi-source data file with all of its tables can be read from a FITS FILE by:

> TASK 'FITLD' ; INP ⏎	to review the inputs needed.
> DATAIN '*filename*' ⏎	to specify the FITS disk file (see § 3.10.3).
> DOUVCOMP FALSE ⏎	to write visibilities in uncompressed format.
> OUTNA '' ⏎	take default (previous *AIPS*) name.
> OUTCL '' ⏎	take default (previous *AIPS*) class.
> OUTSEQ 0 ⏎	take default (previous *AIPS*) sequence #.
> OUTDI 3 ⏎	to write the data to disk 3 (one with enough space).
> INP ⏎	to review the inputs (several apply only to VLBA format files).
> GO ⏎	to run the program when you're satisfied with inputs.

The data-selection adverbs SOURCES, QUAL, CALCODE, and TIMERANG and the table-control adverbs CLINT and FQTOL are used for VLBA-format data only.. See Chapter 9 for more specific information.

4.2 Record keeping and data management

4.2.1 Calibrating data with multiple FQ entries

An observing run with the VLA may result in a *uv* data file containing multiple FQ entries. This may be convenient, but it has a number of costs. If a file contains multiple, independent frequencies, then it occupies more disk space and costs time in every program to skip the currently unwanted data (either a small cost when the index file is used or a rather larger cost when the file must be read sequentially). Since multiple frequencies are still not handled correctly in all programs (*i.e.*, polarization calibration) and since it is not possible to calibrate all of the different FQ data in one pass, you might consider separating the multiple frequencies into separate files (use UVCOP). In either case, you must calibrate each frequency with a separate pass of the scheme outlined below. There are three adverbs to enable you to differentiate between the different FQ entries: FREQID enables the user to specify the FQ number directly (with -1 or 0 meaning to take the first found); SELFREQ and SELBAND enable the user to specify the observing frequency and bandwidth to be calibrated (the tasks then determine to which FQ number these adverbs correspond). If SELFREQ and SELBAND are specified they override the value of FREQID.

There are certain bookkeeping tasks that must be performed between calibrating each FQ set. First, you must ensure that you have reset the fluxes of your secondary calibrators by running SETJY with OPTYPE = 'REJY' — if not, this will cause the amplitudes of your data to be incorrect. Second, it is wise to remove the SN tables associated with any previous calibration using the verb EXTDEST. Although this is not strictly necessary, it will simplify your bookkeeping.

4.2.2 Recommended record keeping

It is useful to print a summary of the time stamps and source names of the scans in your data set. This reminds you of the structure of your observing program when you decide on interpolation and editing strategies, and may help to clarify relationships between later, more detailed listings of parts of the data set. It is also useful to have a printed scan summary and a map of the antenna layout if you need to return to processing the data months or years later. Finally, it is also making sure that all *AIPS* input parameters have their null (default) values before invoking the parts of the calibration package, such as CALIB, that have many inputs. The null settings of most parameters are arranged to be sensible ones so that basic VLA calibration can be done with a minimum of specific inputs; but some inputs may lose their default values if you interleave other *AIPS* tasks with the calibration pattern recommended below. Therefore, you should *always* review the input parameters with INP *taskname* ↵ before running task *taskname*.

We suggest that you begin a calibration session with the following inputs:

> DEFAULT LISTR ↵	to set all LISTR's inputs to null (default) values, also setting the TASK adverb.
> INP ↵	to review the inputs
> INDI n; GETN m ↵	to select the data set, $n = 3$ and $m = 1$ in BDF2AIPS example above.
> TPUT CALIB ↵	to store null values for later use with CALIB.
> OPTYP 'SCAN' ↵	to select scan summary listing.
> DOCRT -1 ↵	to send the output to the printer.
> INP ↵	to review the inputs for LISTR.
> GO ↵	to run the program when the inputs are set correctly.

Note that the DEFAULT LISTR sets the adverbs to select all sources and all times and to send printed output to the terminal rather than the printer. Task LISUN is a useful alternative to LISTR and includes a summary

of the antennas used. It may be useful to have a printed summary of your antenna locations with more details, especially a list of which ones you actually ended up using. To do this, enter

> NPRINT 0 ↵	to do all antennas
> INVERS *n* ↵	to do sub-array *n*
> GO PRTAN ↵	to print the list and a map of antenna locations. The new plot option is really useful only for ALMA data.

In looking over the output from LISTR or LISUN, you may notice that some of the sources you wish to use as calibrators have a blank "Calcode". To mark them as calibrators, use:

> TASK 'SETJY' ; INP ↵	to select the task and review its inputs.
> SOURCES '*sor1*' , '*sor2*' , '*sor3*' , . . . ↵	to select the unmarked calibrator sources.
> OPTYPE 'RESE' ↵	to reset fluxes and velocities.
> CALCODE 'C' ↵	to mark the sources as "C" calibrators.
> GO ↵	to run the task.

This operation will let you select the calibrators by their Calcodes rather than having to spell out their names over and over again. You may wish to consider separate calibrator codes for primary and secondary gain calibrators to make them easier to separate. You may reset a calibrator code to blank by specifying CALCODE = '----'.

4.3 Beginning the calibration

Handling EVLA data is quite different from the situation with the historic VLA. The principal difference is the very wide bandwidth of the EVLA which makes every data set have a large number of spectral channels in every spectral window. This requires corrections for delay errors and bandpass shape which were not needed with historic VLA continuum data. It also means that almost all observations are contaminated by radio frequency interference (RFI) at a level requiring that some portion of the data be marked as bad ("flagged") and omitted from later use. Writing a general list of calibration steps is rather difficult because the steps needed depend on the problems found in the data set. Low frequency data (bands less than 8 GHz) are likely to be heavily contaminated with RFI, while the highest frequency bands have very little. But the high frequencies suffer greatly from rapidly varying "instrumental" (*i.e.*, atmospheric) phase variation. Nonetheless, we will make a list of the general steps that need to be taken following those already described.

1. Correct the data for antenna locations with VLANT (§ 4.3.1)
2. Correct 3-bit data (only) for post-detector gain with TYAPL (§ 4.3.2)
3. Look at some of your calibrator data with POSSM using the TV display (§ 4.3.3)
4. If there is significant narrow-band RFI causing spectral ringing, copy the data applying a Hanning smooth with SPLAT (§ 4.3.4)
5. If there is significant RFI on your calibrator sources, flag the bad channels and times using UVFLG, FTFLG, RFLAG, or some other flagging task. (§ 4.3.5)
6. If the phases show significant slope with frequency, run FRING to find delay corrections and apply them with CLCAL (§ 4.3.6)
7. Run SETJY with OPTYPE 'CALC' on the primary flux calibrators and set CALCODEs for other calibrators if needed. (§ 4.3.7)
8. Run BPASS to find corrections for bandpass shape and examine them with BPEDT, perhaps flagging some calibrator data and re-running BPASS. (§ 4.3.8)

9. Run VLACALIB separately for each calibration source (§ 4.3.9)
10. Run GETJY to determine the calibration sources' flux from that of the primary flux calibrator (§ 4.3.10)
11. Use EDITA to examine the resulting SN table for bad values and to flag calibrator data as needed. (§ 4.3.11)
12. If significant editing was done, run SETJY with OPTYPE 'REJY' on the secondary calibrators; then re-run VLACALIB and GETJY (§ 4.3.9, § 4.3.10)
13. Run CLCAL to apply the SN table to the CL table (§ 4.3.12)
14. Examine your data to see if there remains significant RFI. Use POSSM or SPFLG on a limited set of baselines or even FTFLG although this last will encourage you to flag too much data.
15. If there is significant RFI, use multiple passes of RFLAG to address the problem. (§ 4.3.13)
16. If you have deleted a bunch of calibrator data, delete the calibration tables except for CL tables 1 and 2. (§ 4.3.14)
17. For 8-bit data (only), examine the SY table with EDITA or SNPLT. Correct it as needed with TYSMO and then apply it to the data with TYAPL or SYSOL for Solar data. (§ 4.3.15)
18. Now return to the FRING step and repeat.
19. If you wish to calibrate polarization, run RLDLY to correct right - left delay, run PCAL to find the antenna D terms, and then RLDIF to correct the right - left phase difference. Note that this requires a calibration source of known linear polarization. (§ 4.3.16)
20. Now examine your target source data with POSSM and edit them with RFLAG
21. Calibrate the weights with REWAY if TYAPL has not already done this. Shift spectral-line data to constant velocity with CVEL. (§ 4.3.17)
22. Try some quick images with IMAGR to check the calibration (§ 4.3.18)
23. Back up your calibrated data with FITTP or FITAB (§ 4.3.19)
24. Apply the calibration and editing to the target sources, writing single-source files for imaging and self-calibration with SPLIT (§ 4.3.20)

4.3.1 Baseline corrections

Sometimes, *e.g.*, during a VLA array re-configuration, your observations may have been made when one or more of the antennas had their positions poorly determined. The positional error is usually less than a centimeter at the VLA, but even this may affect your data significantly. The most important effect is a slow and erroneous phase wind which is a function of source position and time. Since this error is a function of source position, it cannot be removed exactly using observations of a nearby calibrator, although the error will be small if the target source is close to the calibrator. In many observations, the target sources and calibrators are sufficiently close to allow this phase error to be ignored. Self-calibration will remove this error completely *if* you have enough signal-to-noise to determine the correction during each integration.

The maximum phase error introduced into the calibrated visibility data by incorrect antenna coordinates $\Delta\phi_B$, in radians, by a baseline error of ΔB meters is given by

$$\Delta\phi_B \approx 2\pi\Delta\theta\Delta B/\lambda$$

where $\Delta\theta$ is the angular separation between the calibrator and the target source in radians and λ is the wavelength in meters.

Note, however, that the error due to the phase-wind is not the only error introduced by incorrect antenna positions. A further, but much smaller effect, will be incorrect gridding of the data due to the erroneous calculation of the baseline spatial frequency components u, v and w. This effect is important only for full primary beam observations in which the antenna position error is of the order of a meter. It is highly unlikely that such a condition will occur. Note too, that this error *cannot* be corrected by the use of self-calibration. However, after correcting the antenna position with CLCOR, you may run UVFIX to compute corrected values of u, v, and w. The maximum phase error in degrees, $\Delta\phi_G$, caused by incorrect gridding of the *u,v,w* data is

$$\Delta\phi_G \approx 360\Delta\epsilon\Delta\Theta$$

where $\Delta\epsilon$ is the antenna position error in antenna diameters and $\Delta\Theta$ is the angular offset in primary beams.

If baseline errors are significant they need to be removed from your data before calibration. It is important to do this to CL table 1, right after running BDF2AIPS. For the EVLA, use the task VLANT. This task determines and applies the antenna position corrections found by the VLA operations staff after your observation was complete. To run VLANT:

> TASK 'VLANT' ⏎

> INDISK *m* ; GETN *n* ⏎ — to get the correct data set. Note that you don't have to keep doing this unless you switch between different input data files.

> FREQID *1* ⏎ — to choose FQ 1.

> SUBARRAY *x* ⏎ — to choose the antenna table to correct.

> GAINVER 1 ⏎ — to choose the correct version of the CL table to read. A new one will produced.

> GO ⏎ — to run VLANT.

For arrays other than the VLA, use CLCOR to enter the antenna position corrections (in meters) in a new CL table and the old AN table. This must be done for each affected antenna in turn. CLCOR puts the corrections into the AN table as well as the CL table, so it is wise to save the AN table before running CLCOR by running TASAV.

> TASK 'CLCOR' ⏎

> INDISK *m* ; GETN *n* ⏎ — to get the correct data set. Note that you don't have to keep doing this unless you switch between different input data files.

> SOURCES ' ' ;STOKES ' ' ⏎ — to do all sources, all Stokes,

> BIF 0 ; EIF 0 ⏎ — and all IFs.

> SUBARRAY *x* ⏎ — to choose the correct sub-array.

> OPCODE 'ANTP' ⏎ — to select the antenna position correction mode.

> GAINVER 1 ⏎ — to choose the correct version of the CL table to read.

> GAINUSE 0 ⏎ — to have CLCOR create a new table.

> ANTENNA *k* ⏎ — to select antenna.

> CLCORPRM $\Delta b_x, \Delta b_y, \Delta b_z, 0$ ⏎ — to add the appropriate antenna corrections in meters.

> GO ⏎ — to run CLCOR.

The program will need to be run as many times as there are antennas for which positional corrections must be made. Set GAINUSE and GAINVER both to 2 after the first correction. Otherwise, with the above adverbs, CLCOR will make multiple CL table versions each with only one correction in them. Note that subsequent calibration must be applied to CL table 2 to create higher versions of the calibration table. This new CL table (version 2) will replace version 1 in all of the subsequent sections on calibration. Thus, in

subsequent executions of CALIB, you must apply these corrections by specifying DOCALIB TRUE ; GAINUSE 0 (for highest version). Note too that CLCOR and VLANT change the antenna file for the changed antenna location(s). Therefore, it is wise to save the AN table before running CLCOR or VLANT by running TASAV.

Note that NRAO's data analysts use the *AIPS* task LOCIT to determine the antenna position corrections; see Appendix V.. These are available to the general user, but a data set designed to determine antenna corrections is normally required. Such data sets consist of about 100 observations of a wide range of phase calibrators taken as rapidly as possible.

4.3.2 Correcting EVLA 3-bit data

With the historic VLA, the system temperature was measured in real time and a correction for it applied to the data as they were archived. With the EVLA this is not the case. Instead the SY table records the total power when the switched noise tubes are on and the total power when they are off. These values may be used to correct data taken with 8-bit correlation (§4.3.15) to values which would be fully calibrated in Janskys if the noise tube levels and antenna efficiencies were accurately known. For 3-bit data, frequently used to observe very wide bandwidths at high frequency, the the function to correct the visibilities from these measurements is not well known. As a consequence, for such data, we can only correct for any post-detector gain changes. These are also recorded in the SY table, so use

> DEFAULT TYAPL ↵ — to select the task and init the adverbs.
> INDISK *m* ; GETN *n* ↵ — to get the correct data set.
> INEXT 'SY' ↵ — to specify an SY table.
> OPTYPE 'PGN' ↵ — to specify the post-detector gain correction.
> INP ↵ — to check the inputs.
> GO ↵ — to write out a new data set with gain corrections.

Reminder — do this only for 3-bit correlator data. Your 8-bit data will be discussed later.

4.3.3 Check your data with POSSM

The next step is to take a look at your data. Do not be obsessive about this. Choose your main calibration source (that you plan to use for FRING, BPASS, and absolute gain calibration), choose a few baselines (say all baselines to one interior antenna), and use the *AIPS* TV display. At this point you want to look for antennas that appear out of the ordinary (*i.e.*, dead) and for frequencies damaged by RFI. Use

> DEFAULT POSSM ↵ — to select the task and init the adverbs.
> INDISK *m* ; GETN *n* ↵ — to get the correct data set.
> SOURCE '*bandpass_cal*' ↵ — to select the strong bandpass calibrator.
> DOTV 1 ; NPLOTS 1 ↵ — to plot only on the TV, one baseline at a time.
> ANTEN *n1* , 0 ↵ — to select an antenna nearest the center of the array.
> BASELINE 0 ↵ — and all antennas to that antenna.
> DOCAL 1 ; APARM 0 ↵ — to apply the initial calibrations and display vector averaged spectra. Scalar averaged spectra will turn up at the edges reflecting the decreased signal to noise in the outer channels.
> APARM(9)=3 ↵ — to plot all IFs and both polarizations in a single plot.
> GO ↵ — to run the task. Make notes of what you see.

4.3.4 Remove spectral ringing with SPLAT

If there is very narrow-band RFI, the POSSM plots will show ringing - a $\sin(x)/x$ pattern as a function of spectral channel. If this occurs in your data, and it will not occur in some bands, then

> DEFAULT SPLAT ↵	to select the task and init the adverbs.
> INDISK m ; GETN n ↵	to get the correct data set.
> DOCAL 1 ↵	to apply any initial and VLANT calibration.
> SMOOTH 1,0 ↵	to use Hanning smoothing.
> INP ↵	always check the many inputs which should be defaults here.
> GO ↵	to make a new data set with the ringing reduced.

4.3.5 General considerations in flagging

At this point, it is essential only to flag the most egregiously bad data samples. These are, with the EVLA, usually either dead antennas or spectral regions affected more or less uniformly by RFI. We will however use this section for a more general discussion of flagging, most of which will apply more directly to later stages in the reduction. Probably FTFLG may be used carefully at this stage to flag only those channels seen to be generally bad in the POSSM plots.

The philosophy of editing and the choice of methods are matters of personal taste and the advice given below should, therefore, be taken with a few grains of salt. When interferometers consisted of only a couple of movable antennas, there was very little data and it was sparsely sampled. At that time, careful editing to delete all suspect samples, but to preserve all samples which can be calibrated, was probably justified. But modern instruments produce a flood of data, with the substantial redundancy that allows for self-calibration on strong sources. Devoting the same care today to editing is therefore very expensive in your time, while the loss of data needlessly flagged is rarely significant. A couple of guidelines you might consider are:

- Don't flag on the basis of phase. At least with the VLA, most phase fluctuations are due to the atmosphere rather than the instrument. Calibration can deal with these up to a point, and self-calibration (if you have enough signal) can refine the phases to levels that you would never reach by flagging. The exceptions are (1) IF phase jumps which still happen on rare occasions, and (2) RF interference which sometimes is seen as an excursion in phase rather than amplitude.
- Don't flag on minor amplitude errors, especially if they are not common. Except for very high dynamic range imaging, these will not be a problem, and in those cases, self-calibration always repairs or sufficiently represses the problem.
- Don't flag if CALIB reports few closure errors and the SN tables viewed with EDITA, SNPLT, and LISTR and the calibrator data viewed with the matrix format of LISTR show only a few problems.

There are three general methods of editing in *AIPS*. The "old-fashioned" route uses LISTR to print listings of the data on the printer or the user's terminal. The user scans these listings with his eyes and, upon finding a bad point, enters a specific flag command for the data set using UVFLG. While this may sound clumsy, it is in fact quite simple and by far the faster method when there are only a few problems. In a highly corrupted data set, it can use a lot of paper and may force you to run LISTR multiple times to pin down the exact problems. The "hands-off" route uses tasks which attempt to determine which data are bad using only modest guidance from the user. The most general of these are RFLAG and FLAGR mentioned below. The third and "modern" route uses interactive ("TV"-based) tasks to display the data in a variety of ways and to allow you to delete sections of bad data simply by pointing at them with the TV cursor. These tasks are

TVFLG (§ O.1.6) for all baselines and times (but only shows one IF, one Stokes, and one spectral channel at a time), SPFLG (§ 10.2.2) for all spectral channels, IFs, and times (but only shows one baseline and one Stokes at a time), FTFLG for all spectral channels, IFs, and times (but shows all baselines combined, one Stokes at a time), EDITA (§ 4.3.11) for editing based on SY (SysPower), TY (T_{ant}), SN or CL table values, EDITR (§ 5.5.2) for all times (but only shows a single antenna (1–11 baselines) and one channel average at a time) and WIPER for all types of data (but with the time, polarization, and sometimes antenna of the points not available while editing). TVFLG is the one used for continuum and channel-0 data from the VLA, while FTFLG is only used to check for channel-dependent interference. SPFLG is very useful for spectral-line editing in smaller arrays, such as the Australia Telescope and the VLBA, but is tedious for arrays like the VLA. Nonetheless, it may be necessary for the modern EVLA. (The redundancy in the spectral domain on calibrator sources helps the eyes to locate bad data.) EDITR is more useful for small arrays such as those common in VLBI experiments. EDITA has been found to be remarkably effective using VLA system temperature tables. All four tasks have the advantage of being very specific in displaying the bad data. Multiple executions should not be required. However, they may require you to look at each IF, Stokes, channel (or baseline) separately (unless you make certain broad assumptions); EDITA and EDITR do allow you to look at all polarizations and/or IFs at once if you want. They all require you to develop special skills since they offer so many options and operations with the TV cursor (mouse these days). A couple of general statements can be made

- For data corrupted by RFI, try FTFLG but be careful. It displays frequency versus time for the selected sources and polarization, but averages all baselines together. A few bad baselines can make it look like all are bad and cause you to flag too much. Flagging channels seen to be bad in all POSSM plots is a good thing and checking channels that appear bad in FTFLG more carefully elsewhere is also good.
- The ultimate tool for EVLA flagging is SPFLG but that shows a frequency-time display for each baseline individually. That me be reasonable for the ATCA or VLBA, but 351 baselines in the VLA is a lot of work. Nonetheless, SPFLG may help you identify channels you may wish to flag more generally.
- For highly corrupted data (say with considerable RF interference, significant cross-talk between antennas, or erratic antennas) TVFLG may be useful. It gives an overall view of the data which is far superior to that given by LISTR. RFI and similar problems are more troublesome at lower frequencies, so TVFLG is probably preferred for L, P, and "4" bands.
- Most VLA data at higher frequencies are of good quality and the flexibility of TVFLG is not needed. In such cases, LISTR with OPCODE = 'MATX' can find scans with erroneous points efficiently.
- The displays given by TVFLG and, to a lesser extent, LISTR in its MATX mode are less useful when there are only a few baselines. Thus, for arrays smaller than the VLA, users may wish to use SPFLG on spectral-line data sets and EDITR on continuum data sets.
- A reasonable strategy to use is to run LISTR first. If there are only a few questionable points, use LISTR and UVFLG, otherwise switch to an interactive task, such as EDITA followed by TVFLG.
- Task FLAGR is a somewhat experimental task to measure the rms in the data on either a baseline or an antenna basis and then delete seriously discrepant points and times when many antennas/correlators are questionable. It also clips amplitudes and weights which are outside specified normal ranges. Task FINDR reports the rmses and excessive values to assist in running FLAGR.
- Task RFLAG flags data on the belief that RFI is highly variable in time and/or frequency. It can do plots showing the time and/or frequency statistics as a function of spectral channel and recommends flagging levels. These, or user-chosen values, may be applied to the data to produce large flag tables; see § 4.3.13. RFLAG is best used after data have some initial calibration to remove delay and bandpass issues and equalize the flux scale.
- Task CLIP makes entries in a flag table, applying calibration and then testing amplitudes for reasonableness on a source-by-source basis. In 31DEC23, it even has an option to flag on the difference of parallel-hand correlations. Use that option only after you have good calibration. CLIP can be very

useful for large data sets, but does not show you the bad data to evaluate yourself. ACLIP does the same operation on auto-correlation data.

- Task FGSPW performs a matrix scalar average over all spectral channels on a per-IF (aka "spectral window") per polarization per baseline per SOLINT or scan basis. It then flags those IFs with excessive amplitude. This operation is designed to catch those windows where amplitudes have overflowed the hardware due to RFI within the spectral window and may catch other problems as well.
- Task WIPER makes entries in a flag table for all data samples wiped from a UVPLT-like display of any *uv* data set parameter versus any other parameter. The source, Stokes, IF, time, etc. of the points are not known during the interactive editing phase, but some baseline information is displayed. It can plot and edit any choice of STOKES in one execution and can flag/unflag by baseline.
- Task DEFLG makes entries in a flag table whenever the phases are too variable as measured by too low a ratio of vector-averaged to scalar-averaged amplitudes. This may be useful when applied to the calibrator source in phase-referencing observations and for other data at the highest and lowest frequencies which are affected by atmospheric and ionospheric phase variability.
- Task SNFLG makes entries in a flag table whenever the phase solutions in an SN or CL table change excessively between samples on a baseline basis. It can also flag data if the amplitude solutions differ from their mean excessively. It can instead flag data when the amplitude and/or phase solutions or solution weights are outside user-specified ranges.
- Task REWAY examines spectral rms to try to determine better weights for the data. It is better to use it after delay, bandpass, and complex gain issues have been addressed at some level. Run it with no flagging of the output for bad values of the spectral rms. Then plot the weights with VPLOT or ANBPL to look for weights that are seriously abnormal (high or low). Those data may need to be flagged. High weights mean that the data are of abnormally low amplitude, whilst low weights mean that the data are very noisy. REWAY uses robust methods to find the rms and so a few channels of RFI may not cause very low weights, but lots of RFI or receiver failures will make the weights abnormally low. REWAY now displays statistical information to help you assess what weights are "high" and "low".
- Task WETHR makes entries in a flag table whenever various weather parameters exceed specified limits. WETHR also plots the weather (WX) table contents.
- Task VPFLG flags all correlators in a sample whenever one is flagged. Observations of sources with circular polarization (Stokes V) require this operation to correct the flagging done on-line (which flags only known bad correlators).
- Task FGPLT plots the times of selected flag-table entries to provide you information on what these powerful tasks have done.
- Task FGPRT prints a matrix of the number of flags applying to each baseline in a user specified range of time, spectral channel, IF, source, etc.

Many of the above tasks produce large flag tables. The task REFLG attempts to compress such tables, sometimes quite remarkably and sometimes incorrectly. Task FGDIF may be used to confirm that the flag tables before and after flag the same things. FGCNT may also help confirm this.

4.3.6 Correcting delay errors with FRING

We have had difficulty setting all of the delays in the EVLA to values which are sufficiently accurate. If the delay is not set correctly, the interferometer phase will vary linearly with frequency, potentially wrapping through several turns of phase within a single spectral window ("IF band"). Your POSSM displays probably show such an effect. Delay errors are a problem familiar to VLBI users and *AIPS* has a well-tested method to correct the problem. Using your LISTR output, select a time range of about one minute *toward the end of a scan* on a strong point-source calibrator, usually your bandpass calibrator. Then

> DEFAULT FRING ; INP C$_R$ — to initialize the FRING inputs and review them.

> INDI n; GETN m C$_R$ — to select the data set on disk n and catalog number m.

> SOLINT 1.05 * x C$_R$ — to set the averaging interval in minutes slightly longer than the data interval (x) selected.

> TIMERANG *db* , *hb* , *mb* , *sb* , *de*, *he* , *me* , *se* C$_R$ — to specify the beginning day, hour, minute, and second and ending day, hour, minute, and second (wrt REFDATE) of the data to be included. Too much data will cause trouble.

> DPARM(9) = 1 C$_R$ — to fit only delay, not rate. *This is very important, but for EVLA only, 0 is treated as 1.*

> DPARM(4) = t C$_R$ — to help the task out by telling it the integration time t in seconds. Oddities in data sample times may cause FRING to get a very wrong integration time otherwise.

> INP C$_R$ — to check the voluminous inputs.

> GO — to run the task, writing SN table 1 with delays for each antenna, IF, and polarization.

The different IFs in current EVLA data sets may come from different basebands and therefore have different residual delays. The option APARM(5) = 3 to force the first $N_{if}/2$ IFs to have one delay solution while the second half of the IFs has another is strongly recommended, but only when the first half all come from one of the "AC" or "BD" basebands (hardware IFs) and the second half come from the other. The 3-bit data path of the EVLA actually has four hardware IFs, so APARM(5) = 4 produces four delay solutions, dividing the IFs in quarters. The Widar correlator may now separate the hardware basebands into unequal numbers of spectral windows. APARM(5) = -1 will cause FRING to use BPARM to specify the IFs used in each delay solution. Note that, at low frequencies, the phases may also be affected by dispersion (phase differences proportional to wavelength). FRING now offers APARM(10) to enable solving for a single delay plus dispersion from the fitted single-IF delays. If you have a large number of narrow spectral channels, CHINC may be used to reduce the memory and time requirements of FRING. This SN table will need to be applied to the main CL table created by INDXR or OBIT.

> TASK 'CLCAL' ; INP C$_R$ — to look at the necessary inputs.

> TIMERANG 0 C$_R$ — to reset the time range.

> GAINUSE 0 ; GAINVER 0 C$_R$ — to select the highest CL table as input and write one higher as output (probably version 2 and 3, resp. in this case).

> SNVER 1 ; INVER 1 C$_R$ — to use only the SN table just created.

> INP C$_R$ — to review the inputs.

> GO C$_R$ — to make an updated calibration table.

Be sure to apply this (or higher) CL table with DOCALIB 1 in all later steps.

4.3.7 Primary flux density calibrators

Careful measurements made with the D array of the VLA have shown that the Baars *et al.*[1] formulæ for "standard" calibration sources are in error slightly, based on the assumption that the Baars' expression for 3C295 is correct. Revised values of the coefficients have been derived by Rick Perley and Brian Butler. Task SETJY has these formulae built into it, giving you the option (OPTYPE 'CALC') of letting it calculate the fluxes for primary calibrator sources 3C48, 3C123, 3C138, 3C147, 3C196, 3C286, 3C295, and 1934-638. The default setting of APARM(2) = 0) will calculate the flux densities based on Perley-Butler 2017 values which cover the range 50 MHz to 50 GHz for the primary calibrators:3C48, 3C138, 3C147, 3C286, and 3C295 plus 3C123 and 3C196, all with a number of synonyms. Other sources which will be computed, but which may not be good calibrator sources, are J0444-2809, PictorA, 3C144 (Taurus A, Crab Nebula), 3C218 (Hydra A), 3C274 (Virgo A), 3C348 (Hercules A), 3C353, 3C380, 3C405 (Cygnus A), 3C444, and 3C461 (Cassiopeia A). multiple names are also allowed for these sources; see EXPLAIN SETJY. Most of the sources in the extra list have limits on the frequency range over which the function is valid; SETJY will tell you if the frequencies are out of range. Higher values of APARM(2) select older systems of coefficients if you need them to match previous data reductions. SETJY will recognize both the 3C and IAU designations (B1950 and J2000) for the standard sources. See EXPLAIN SETJY for details of all models. You may also insert your own favorite values for these sources instead (OPTYPE = ' ') and you will have to insert values for any other gain calibrators you intend to use (or use GETJY, see § 4.3.10). Adverbs SPECINDX and SPECURVE allow you to enter spectral index information to help set the calibrator fluxes.

Unfortunately, since all the primary flux calibrators are resolved by the VLA in most configurations and at most frequencies, they cannot be used directly to determine the amplitude calibration of the antennas without a detailed model of the source structure, see Figure 4.1 as an example. Beginning in April 2004, model images for the calibrators at some frequencies are included with *AIPS*. Models of 3C286, 3C48, 3C138, and 3C147 are available for all 6 traditional bands of the VLA *except* 90 cm and even for S band of the EVLA. Type CALDIR ↵ to see a list of the currently available calibrator models. Sources which are small enough to be substantially unresolved by the VLA have variable flux densities which must be determined in each observing session. A common method used to determine the flux densities of the secondary calibrators from the primary calibrator(s) is to compare the amplitudes of the gain solutions from the procedure described below.

Use SETJY to enter/calculate the flux density of each primary flux density calibrator. The ultimate reference for the VLA is 3C295, but 3C286 (1328+307), which is slightly resolved in most configurations at most frequencies, is the most useful primary calibrator. CALIB has an option that will allow you to make use of Clean component models for calibrator sources. You are strongly encouraged to use the existing models. If you follow past practice at the VLA, you may have to restrict the *uv* range over which you compute antenna gain solutions for 3C286, and may therefore insert a "phony" flux density appropriate only for that *uv* range at this point. In both cases, the following step should be done. CALIB will scale the total flux of the model to match the total flux of the source recorded by SETJY in the source table. This corrects for the model being taken at a somewhat different frequency than your observations and for the model containing most, but not all, of the total flux. An example of the inputs for SETJY, where you let it calculate the flux, would be:

> TASK 'SETJY' ; INP ↵	
> SOURCES '3C286' , ' ' ↵	if you used 3C286 as the source name; see the LISTR scan output.
> BIF 1 ; EIF 2 ↵	will calculate for both "AC" and "BD" IFs.
> OPTYPE 'CALC' ↵	perform the calculation.
> APARM(2) = 0 ↵	to use the VLA "2013" coefficients.
> INP ↵	to review inputs.
> GO ↵	when inputs okay.

[1] 1977, *Astr. & Ap.*, 61, 99

Or you can set the flux manually as shown below:

> TASK 'SETJY' ; INP ↵	
> SOURCES '3C286' , ' ' ↵	if you used 3C286 as the source name.
> ZEROSP 7.41 , 0 ↵	I flux 7.41 Jy, Q, U, V fluxes 0.
> BIF 1 ; EIF 1 ↵	selects first IF IF.
> INP ↵	to review inputs.
> OPTYPE ' '	use values given in ZEROSP.
> GO ↵	when inputs okay.
> BIF 2 ; EIF 2 ↵	selects second IF IF.
> ZEROSP 7.46, 0 ↵	I flux 7.46 Jy at the $2^{\rm nd}$ IF, Q, U, V fluxes 0.
> GO ↵	

Note that, although SOURCES can accept a source list, ZEROSP has room for only one set of I, Q, U, V flux densities. To set the flux densities for several different sources or IFs, you must therefore rerun SETJY for each source and each IF, changing the SOURCES, BIF, EIF, and ZEROSP inputs each time. Alternatively, set ZEROSP to the flux at 1 GHz and enter SPECINDX and SPECURVE adverbs to describe the dependence with frequency.

CALIB will use the V polarization flux in the source table if one has been entered. The RR polarization will be calibrated to I+V and the LL to I-V. While this has little practical use with circular polarizations because V is almost always negligible, it can be used for linearly polarized data from the WSRT. That telescope has equatorially mounted dishes, so the VV polarization is I-Q and the HH is I+Q independent of parallactic angle. For WSRT data, you should relabel the polarizations to RR/LL and enter `I, 0, 0, -Q` for ZEROSP, since Q is not negligible in standard calibrators.

In 31DEC22, more extensive calibration of linear polarization became available. CALIB will use the source table Q and U fluxes and the parallactic angles to compute proper gains for the V and H polarizations of point sources. A new procedure VHCALIB and task VHCAL have been written to apply full Clean-component polarization models to determine the parallel-hand gains of linear systems such a Meerkat.

If the LISTR scan listing shows calibration sources with a blank CALCODE, you should also use SETJY to correct this. Set OPTYPE = ' ', set the desired SOURCE name, and set CODETYPE to the desired CALCODE. Expert users sometimes prefer TABPUT for this function; run PRTAB to see what values must be set in PIXXY.

4.3.8 Calibrating the bandpass shape with BPASS

Bandpass calibration is for the modern VLA *required* rather than merely *recommended*. Having chosen those channels which may be reliably used to normalize the bandpass functions,

> DEFAULT BPASS ; INP	to reset all adverbs and choose the task.
> INDI n; GETN m ↵	to select the data set on disk n and catalog number m.
> DOCAL 1 ↵	to apply the delay calibration — *very important*.
> CALSOUR '*bandpass_cal*' ↵	to select the strong bandpass calibrator.
> SOLINT 0 ↵	to find a bandpass solution for each scan on the BP calibrator.
> ICHANSEL $c11, c12, 1, if1, c21, c22, 1, if2, c31, c32, 1, if3, \ldots$ ↵	to select the range(s) of channels which are reliable for averaging in each IF. Use the central 30% of the channels if your calibrators are all very strong or more like 90% if they are not. Remember these values — you will use them again.

> BPASSPRM(5) 1 ; BPASSPRM(10) 3 ↵ — to normalize the results only after the solution is found using the channels selected by ICHANSEL. Use BPASSP(5)=-1 if your phases are not stable within each scan. Note that there are numerous options for BPASSPRM(10) including medians as well as means.

> GO ↵ — to make a bandpass (BP) table.

Do not use spectral smoothing at this point unless you want to use the same smoothing forever after. Apply the flag table. A model for the calibrator may be used; see § 4.3.9.1.

Note: if your have a strong bandpass calibrator, observed frequently, you may use BPASS instead of CALIB to calibrate the gains. You must, if course, turn off the bandpass normaization entirely to use this option.

BPASS now contains the adverbs SPECINDX and SPECURVE through which the spectral index and its curvature (to higher order than is known for any source) may be entered. For the standard amplitude calibrators 3C286, 3C48, 3C147, and 3C138, these parameters are known and will be provided for you by BPASS. For other sources, you may provide these parameters, but BPASS will fit the fluxes in the SU table for a spectral index (including curvature optionally) if you do not. Note that, if no spectral index correction is applied, the spectral index of the calibration source will be frozen into the target source. Bandwidths on the EVLA are wide enough that this is a serious problem. If you do not know the spectral index of your calibration source, BPASS itself or the new task SOUSP may be used to determine the spectral indices from the SU table. Of course, that means that GETJY must already have been run. Since BPASS must usually be run before CALIB and hence GETJY, this suggests that one may have to iterate this whole process at least once. SOUSP now offers the option of correcting one or more SN tables after it adjusts the source fluxes for the spectral index it determined. This may reduce the need for further iterations.

Note that the bandpass parameters shown above assume that the phases are essentially constant through each scan of the bandpass calibrator. This may not be true, particularly at higher frequencies. In this case, you have two choices. One is to set BPASSPRM(5) to 0 which will determine the vector average of the channels selected by ICHANSEL at every integration and divide that into the data of that integration. This will remove all continuum phase fluctuations, but runs a risk of introducing a bias in the amplitudes since they do not have Gaussian statistics. BPASSPRM(5) = -1 now applies a phase-only correction on a record-by-record basis. A better procedure, which is rather more complicated, is as follows. Use SPLIT to separate the bandpass calibrator scans into a separate single-source file applying any flags and delay calibration and the like. Then run CALIB on this data set with a short SOLINT to determine and apply a phase-only self-calibration. On the uv data set written out by CALIB, run BPASS using the parameters described in the previous paragraph. Finally, use TACOP to copy the BP table back to the initial data set.

The spectral quality of the final images has been found to be determined in part by the quality of the bandpass solutions. In particular, for reasons which are not yet known, the bandpasses are not exactly antenna dependent especially in the edge channels. This "closure error" may be measured in individual and statistical ways by BPASS and reported to you. To check on this problem for your data set, set

> MINAMPER a ↵ — to count and, if BPASSPRM(2)>1, to report amplitude closure failures $> a$ per cent. Note that closure errors are accumulated as logarithms so that 0.5 and 2.0 are both errors of 100%.

> MINPHSER p ↵ — to count and, if BPASSPRM(2)>1, to report phase closure failures $> p$ degrees.

> BPASSPRM(2) 1 ↵ — to report statistics of amplitude and phase closure failures without reporting individual failures.

> BPASSPRM(6) a ↵ — to report all channels in which the average amplitude closure error $> a$ per cent.

> BPASSPRM(7) p ↵ — to report all channels in which the average phase closure error $> p$ degrees.

> SOLTYPE 'R' ↵ — to select robust solutions which discard data with serious closure problems. Try other types if there are solution failures.

It is probably a good idea to set MINAMPER and MINPHSER fairly high (*i.e.*, 20 and 12) to make a big deal only about major excursions, but to set BPASSPRM(6) and (7) fairly low (*i.e.*, 0.5 and 0.5) to view the spectrum of closure errors (which will look a lot like the spectrum of noise on your final Clean images). There is even a task called BPERR which will summarize and plot the error reports generated by BPASS and written to text files by PRTMSG.

The bandpass solutions are calculated at each bandpass calibrator scan. As a consequence, they are likely to be unevenly spaced in time and may even have times (due to on-line or later editing) at which there are solutions for some IFs and polarizations but not all. When the latter happens, program source data will be lost unless the missing solutions are filled in. The task BPSMO may be used for this purpose or to create a new BP table at regular time intervals using one of a number of time-smoothing functions. Set APARM(4) = -1 for the "repair" mode or set APARM(4) to the desired BP interval.

After the bandpasses have been generated, you can examine them using tasks BPLOT and POSSM. You can obtain an average from all antennas with

> TASK 'POSSM' ↵
> INDI *i* ; GETN *j* ↵ — to specify the line data set.
> SOURCES '*cal1*' , '*cal2*' , . . . ↵ — to specify the bandpass calibrators.
> ANTENNAS 0 ↵ — to include all antennas.
> TIMER 0 ↵ — to average over all times.
> BCHAN 1 ; ECHAN 0 ↵ — to display all channels.
> BPVER 1 ↵ — to select the BP table.
> FREQID 1 ↵ — to set the FQ value to use.
> APARM = -1, 0 ↵ — to do a scalar average and have the plot self-scaled and labeled in channels.
> APARM(8) 2 ↵ — to plot BP table data.
> NPLOTS 0 ↵ — to make one plot only, averaging all included data.
> INP ↵ — to review the inputs — check closely.
> GO ↵ — to run the program when inputs set correctly.
> GO LWPLA ↵ — to send the plot to the (PostScript) printer/plotter.

To view each antenna individually, using the TV to save paper

> DOTV TRUE ↵ — to use the TV.
> NPLOTS 1 ↵ — to plot one antenna per page/screen.
> GO ↵ — to display the bandpasses, averaged over time, on the TV with one antenna per screen.

POSSM shows each screen for 30 seconds before going ahead. You can cause it wait indefinitely by hitting button A, speed it up by hitting TV buttons B or C, or tell it to quit by hitting button D. If DOTV = -1, then POSSM makes multiple plot extension files, which can be sent to the printer (individually or collectively) by LWPLA. You might want to use a larger value of NPLOTS to reduce the number of pieces of paper.

BPLOT is used to create one or more plots (on the TV or in plot files) of the selected bandpass table. The plots will be a set of profiles separated on the vertical axis by an increment in time or antenna number (depending on the sort selected). More than one plot for more than one antenna or more than one time may be generated. Multiple IFs and polarizations will be plotted along the horizontal axis if they are present in the BP table and selected by the adverbs. Thus, BPLOT is useful for plotting the change in bandpass shape as a function either of time or of antenna.

Task BPEDT offers a TV-interactive graphical way to check your BP table. It displays the amplitude and phase of one antenna to be edited and up to 10 more antennas for comparison. The task is like EDITA except

that the horizontal axis is spectral channel rather than time. You may walk through the entire bandpass table selecting the editable antenna and time in sequence. If some of the solutions show residual effects from RFI, then you may generate flags in a flag table to delete certain spectral channels for certain antennas and times. If you do this, of course, you should re-run BPASS applying the new flags. In 31DEC21, BPEDT offers the option of flagging the BP values instead and even interpolating over the flagged values. Task BPFLG may be used with large BP tables after you have determined appropriate clip levels. Task BPEPL may be used to make plots similar to those of BPEDT without the interactivity. It will be faster if you have a lot of times in the BP table. If you do have lots of times in your BP table, BPPLT is another way to view a portion of the table as a function of time.

The BP tables are applied to the data by setting the adverb DOBAND > 0 and selecting the relevant BP table with the adverb BPVER. There are three modes of bandpass application. The first (DOBAND 1) will average all bandpasses for each antenna within the time range requested, generating a global solution for each antenna. The second mode (DOBAND 2) will use the antenna bandpasses nearest in time to the data point being calibrated. The third mode (DOBAND 3) interpolates in time between the antenna bandpasses and generates the correction from the interpolated data. This mode has been found to be required for VLA data. If BPSMO was used to make a fairly finely sampled BP table, then DOBAND 2 may be used. Modes DOBAND 4 and DOBAND 5 are the same as modes 2 and 3, respectively, except that data weights are ignored.

It is often not possible to observe a strong bandpass calibrator many times during a run. In this case, one can run BPASS on the single scan on the strong calibrator and then remove the main bandpass shape with DOBAND 1 in task SPLAT. Corrections to this basic bandpass shape as a function of time may then be determined with adequate signal-to-noise using task CPASS. This task can be used to fit the residual bandpass with a small number of parameters ($<<$ the number of spectral channels) at each calibrator scan. The results may then be applied with DOBAND 2. Check the output of CPASS carefully — it is capable of making bandpass shapes with large ripples that are not present in the data. CPASS also knows about the flux coefficients of all standard sources and can fit the spectral index of other sources from the flux values in the SU table.

4.3.9 Calibrating the complex antenna gains with VLACALIB

4.3.9.1 Using calibrator models

It is now considered standard practice to use flux calibrator models and you are strongly encouraged to do so. As mentioned above, all the primary flux calibrators are resolved at *most* frequencies and configurations. Figure 4.1 shows the visibilities and image of the commonly used calibrator 3C48 at X-band, it is obvious this source is far from being point like. Since April 2004, source models have been shipped with *AIPS* as FITS files. Currently, models for 3C48, 3C286 and 3C138 are available for all bands except 90 cm and 3C147 at all bands except, X, C and P. Rick Perley has many models on two different web sites, one for 2014 and one for 2019 observations. These may be reached in 31DEC20 by setting DATAIN appropriately. Additional models are in the works, so you should always check to see what is available:

> CALDIR ⏎ to list the available models by source name and band code.

You can load a model with the procedure VLACALIB (see below) or you can load the model in manually using CALRD:

> TASK 'CALRD' ⏎ to select the calibrator source reading task.
> OBJECT '3C286' ⏎ to load a model of 3C286.
> VLOBS 'K' ⏎ to select the available model at K band.
> OUTDISK n ⏎ to write the model image and Clean components to disk n.
> GO ⏎ to run the task and load the model.

Set DATAIN = '2019' to select models from Perley's 2019 website. Soon the task MOD3D will enable muti-facet models at P and 4 band to be distibuted with a "3D" model attached to a single facet.

4.3.9.2 Calibrating complex gains

The primary calibration task in *AIPS* is CALIB. Most of the complexity of CALIB can be hidden using the procedure VLACALIB. Before attempting to use this procedure, you must first load it by typing:

> RUN VLAPROCS ↵ to compile the procedures.

Type HELP VLAPROCS for a full list of the procedures available in VLAPROCS.

The procedure VLACALIB *automatically* downloads and uses calibrator models, if one is available and the inputs are set correctly. After you have loaded VLACALIB you may invoke the calibrator model usage by setting:

> INDI n ; GETN m ↵ to select the data set, $n = 3$ and $m = 1$ above.
> CALSOUR = '*Cala*' ↵ to name *one* primary flux calibrator to invoke automatic calibrator model usage.
> UVRANGE 0 ↵ set to zero to invoke automatic calibrator model usage.
> ANTENNAS 0 ↵ set to zero to invoke automatic calibrator model usage.
> REFANT n ↵ reference antenna number — use a reliable antenna located near the center of the array.
> MINAMPER 10 ↵ display warning if baseline disagrees in amplitude by more than *10%* from the model.
> MINPHSER 10 ↵ display warning if baseline disagrees by more than 10° of phase from the model.
> FREQID 1 ↵ use FQ number 1.
> DOPRINT 1 ; OUTPRINT ' ' ↵ to generate significant printed output on the line printer.
> INP VLACALIB ↵ to review inputs.
> VLACALIB ↵ to make the solution and print results.

This procedure will load the calibrator model (using CALRD) and use it when it runs CALIB, then print any messages from CALIB about closure errors on the line printer, and finally run LISTR to print the amplitudes and phases of the derived solutions. Plots of these values may be obtained using task SNPLT.

Then you may select the model image with GET2N for use directly in CALIB. Example inputs for CALIB are:

> TASK 'CALIB'; INP ↵ to select task and review inputs.
> INDI n ; GETN m ↵ to select the data set, $n = 3$ and $m = 1$ above.
> CALSOUR = '*Cala*' , ' ' ↵ flux calibrator for which you have a model.
> UVRANGE 0 ↵ no *uv* limits needed.
> ANTENNAS 0 ↵ antenna selection not needed.
> REFANT n ↵ reference antenna number — use a reliable antenna located near the center of the array.
> WEIGHTIT 1 ↵ to select $1/\sigma$ weights which may be more stable.
> IN2DI o ; GET2N p ↵ to select the model.
> NCOMP 0 ↵ to use all components.
> SOLMODE 'A&P' ↵ to do amplitude and phase solutions.
> APARM(6) 2 ↵ to print closure failures.
> MINAMPER 10 ↵ to display warning if baseline disagrees in amplitude by more than *10%* from the model.

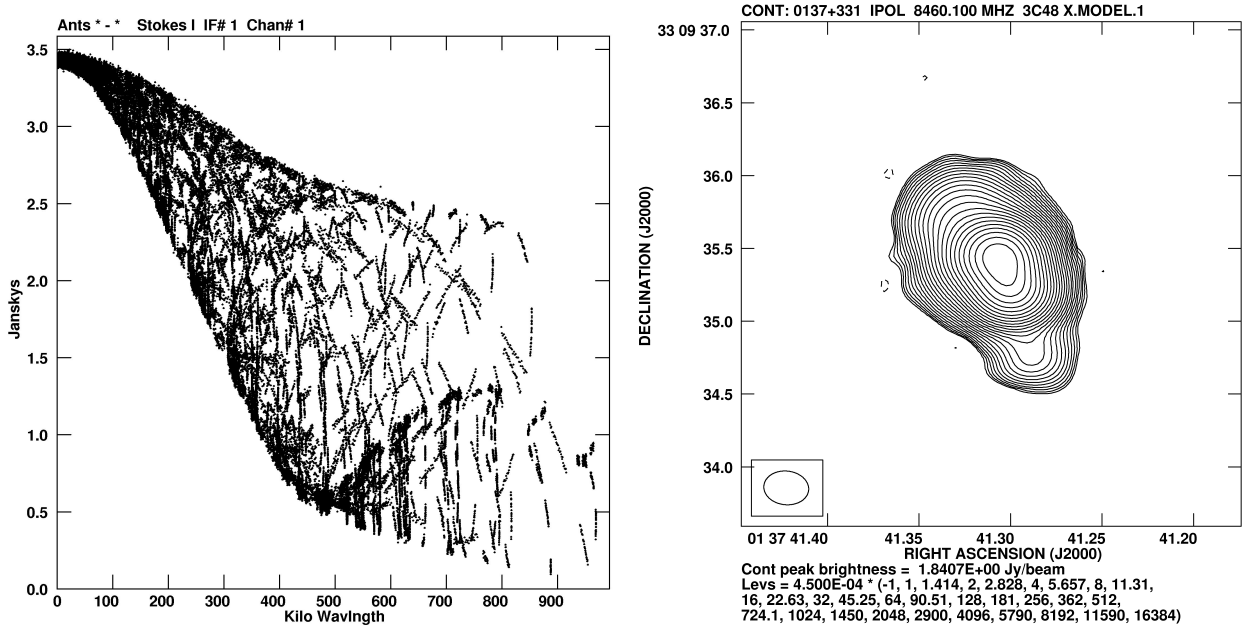


Figure 4.1: Displays of the visibilities (left) and image (right) for the fundamental calibration source 3C48. The plots were made using UVPLT, KNTR, and LWPLA; see § 6.3.1 and § 6.3.2.1. Data from all VLA configurations including the VLBA antenna in Pie Town were used. A point source would have visibilities that have a constant amplitude at all baselines and an image matching the beam plotted in the lower-left corner.

> MINPHSER 10 C_R	to display warning if baseline disagrees by more than 10° of phase from the model.
> CPARM(5) 1 C_R	to vector average amplitudes over spetral channels and then scalar average them over time before determining solution.
> FREQID 1 C_R	to use FQ number 1.
> INP CALIB C_R	to review inputs.
> GO C_R	to make the solution.

CALIB will use the clean components table attached to the model to find antenna gain solutions. It will sum the clean components within a certain radius of the center of the map (so that confusing sources that are part of the model do not influence the gain) and scale them to the flux in the SU table. Therefore, you must still run SETJY before running CALIB.

After running CALIB check the solutions for all antennas with SNPLT or LISTR (OPTYPE='GAIN'). If you have multiple primary or secondary calibrators you will have to run CALIB separately for each, using models where they are available and restricting the UVRANGE and ANTENNAS where they are not. You can either write into the same SN table by setting SNVER to a table number or to different SN tables by setting SNVER = 0. Then you you can proceed as normal flagging and editing your data and proceed to final calibration as described in § 4.3.10.

4.3.9.3 Flux calibration without calibrator models

Normally, the primary flux calibration sources are located too far from the target sources to be used to track time variations in phase and amplitude. Therefore, secondary calibration sources whose fluxes and spectral

indices are not known in advance.

Once you have read in procedure VLACALIB (see § 4.3.9.1), you may use it to invoke CALIB. You will have to do this once for each calibrator, unless you can use the same UVRANGE for more than one of them. Thus,

> INDI n ; GETN m ↵	to select the data set, $n = 3$ and $m = 1$ above.
> CALSOUR = '*Cala*' , '*Calx*' ↵	to name two calibrators using the same UVRANGE and other adverb values.
> UVRANGE *uvmin uvmax* ↵	uv limits, if any, in kiloλ.
> ANTENNAS *list of antennas* ↵	antennas to use for the solutions, see discussion above.
> REFANT n ↵	reference antenna number — use a reliable antenna located near the center of the array.
> MINAMPER 10 ↵	display warning if baseline disagrees in amplitude by more than *10%* from the model.
> MINPHSER 10 ↵	display warning if baseline disagrees by more than 10° of phase from the model.
> DOPRINT 1 ; OUTPRINT ' ' ↵	to generate significant printed output on the line printer.
> FREQID 1 ↵	use FQ number 1.
> INP VLACALIB ↵	to review inputs.
> VLACALIB ↵	to make the solution and print results.

This procedure will first run CALIB, then print any messages from CALIB about closure errors on the line printer, and finally run LISTR to print the amplitudes and phases of the derived solutions. Plots of these values may be obtained using task SNPLT.

If the secondary calibrators require different values of UVRANGE, then CALIB must be run until it has run for all calibration sources. Attached to your input data set is a solution SN table. Each run of CALIB writes in this table (if SNVER = 1), for the times of the included calibration scans, the solutions for all IFs using the flux densities you set for your calibrators with SETJY or GETJY. (CALIB assumes a flux density of 1 Jy if no flux density is given in the SU table.) If a solution fails, however, the whole SN table can be compromised, forcing you to start over. It is possible to write multiple SN tables with SNVER = 0. Later programs such as GETJY and CLCAL will merge all SN tables which they find (if told to do so). Tables with failed solutions must be deleted.

4.3.10 Bootstrapping secondary flux-density calibrators

Task GETJY can be used to determine the flux density of the secondary flux calibrators from the primary flux calibrator based on the flux densities set in the SU table and the antenna gain solutions in the SN tables. The SU and SN tables will be updated by GETJY to reflect the calculated values of the secondary calibrators' flux densities. This procedure should also work if (incorrect) values of the secondary calibrators' flux densities were present in the SU table when CALIB was run. Bad or redundant SN tables should be deleted using EXTDEST before running GETJY, or avoided by selecting tables one at a time with adverb SNVER.

To use GETJY:

> TASK 'GETJY' ; INP ↵	
> SOURCES '*cal1*' , '*cal2*' , '*cal3*' . . . ↵	to select secondary flux calibrators.
> CALSOU '3C286' , ' ' ↵	to specify primary flux calibrator(s).
> CALCODE ' ' ↵	to use all calibrator codes.
> BIF 1 ; EIF 2 ↵	to do both IFs.
> FREQID 1 ↵	to use FQ number 1.

> ANTENNAS 0 C_R	to include solutions for all antennas.
> TIMERANG 0 C_R	to include all times.
> SNVER 0 C_R	to use all SN tables.
> INP C_R	to review inputs.
> GO C_R	to run the task when the inputs are okay.

GETJY will give a list of the derived flux densities and estimates of their uncertainties. These are now found by "robust" methods and additional information about numbers of aberrant solutions are given. If any of the uncertainties are large, then reexamine the SN tables as described above and re-run CALIB and/or GETJY as necessary. Multiple executions of GETJY will not cause problems as previous solutions for the unknown flux densities are simply overwritten. You may wish to run the task SOUSP to determine the spectral indices of your calibrators from their fluxes in the SU table. You can even replace the values in the SU table with the curve fit by SOUSP and correct the gains in one or more SN tables with the newly determined fluxes. These spectral index parameters may be useful in running BPASS and PCAL. However, BPASS knows the flux coefficients for all standard calibration sources and can fit the spectral index of other calibration sources from the SU table.

4.3.11 Editing visibility data with EDITA

The task EDITA uses the graphics planes on the *AIPS* TV display to plot data from tables and to offer options for editing (deleting, flagging) the associated *uv* data. Only the TY (VLB and historic VLA system temperature), SY (modern VLA SysPower table), SN (solution), and CL (calibration) tables may be used. We recommend using EDITA with the SN table updated by GETJY as a quick and easy way to examine the quality of the solutions. If a small amount is disturbed, use EDITA to flag the relevant data. If the solutions for most antennas are disturbed, make a note of the time and source and use other flagging tools to find and flag the cause of the disturbance. If you flag more than a little, you should delete the SN table(s) and re-run VLACALIB and GETJY. Try:

> TASK 'EDITA ; INP C_R	to review the inputs needed.
> INDI n ; GETN m C_R	to select the data set, $n = 3$ and $m = 1$ above.
> INEXT 'SN' C_R	to use the solution table.
> INVERS 0 C_R	to use the highest numbered table, usually 1.
> TIMER 0 C_R	to select all times.
> BIF 1 ; EIF 0 C_R	to specify all IFs; you can then toggle between them interactively and even display all at once.
> ANTENNAS 0 C_R	to display data for all antennas.
> FLAGVER 1 C_R	to use flag (FG) table 1 on input.
> OUTFGVER 0 C_R	to create a new flag table with the flags from FG table 1 plus the new flags.
> SOLINT 0 C_R	to avoid averaging any samples.
> DOHIST FALSE C_R	to omit recording the flagging in the history file.
> DOTWO TRUE C_R	to view a 2^{nd} observable for comparison
> CROWDED 2 ; DO3COL TRUE C_R	to allow plots with all polarizations and/or IFs simultaneously, using color to differentiate the polarizations and IFs. The 2 speeds up the display.
> INP C_R	to review the inputs.
> GO C_R	to run the program when inputs set correctly.

If you make multiple runs of EDITA, it is important to make sure that the flagging table entries are all in one version of the FG table. The easiest way to ensure this is to should set FLAGVER and OUTFGVER to 0 and keep

it that way for all runs of EDITA. This may create an excessive number of flag tables, but unwanted ones may be deleted with EXTDEST. If you make a mistake two flag tables may be merged with the task TAPPE. A sample display from EDITA is shown on the next page.

The following discussion assumes that you have read § 2.3.2 and are familiar with using the *AIPS* TV display. An item in a menu such as that shown in the figure is selected by moving the TV cursor to the item (holding down or pressing the left mouse button). At this point, the menu item will change color. To obtain information about the item, press *AIPS* TV "button D" (usually the D key and also the F6 key on your keyboard). To tell the program to execute the menu item, press any of *AIPS* TV buttons A, B, or C. Status lines around the display indicate what is plotted and which data will be flagged by the next flagging command. In the figure below, only the displayed antenna (2), and time range will be flagged. You must display at least a few lines of the message window and your main AIPS window since the former will be used for instructions and reports and the latter will be needed for data entry (*e.g.*, antenna selection).

The first thing to do with EDITA is to look at all of the polarizations, IFs, and antennæ, in order to make notes about disturbances in the solutions. Use SWITCH POLARIZATION to switch between polarizations and ENTER IF to select the IF to edit. Alternatively, NEXT POL/IF will cycle through all polarizations and IFs. If CROWDED was set to 1 or 2, SWITCH POLARIZATION will cycle through displaying both polarizations as well as each separately, and ENTER IF will accept 0 as indicating all. These options appear only if there is more than one polarization and/or more than one IF in the loaded data. Use ENTER ANTENNA to select the antenna to be flagged and ENTER OTHER ANT to select secondary antennæ to be displayed around the editing area. If the secondary antennæ have no obvious problems, then they do not have to be selected for editing. EDITA will plot all of the times in the available area, potentially making a very crowded display. You may select interactively a smaller time range or "frame" in order to see the samples more clearly. It is necessary to select each frame in order to edit the data in that frame so it helps to make the TV screen as big as possible with the F2 button or your window manager. Note that the vertical scales used by EDITA are linear, but that the horizontal scale is irregular and potentially discontinuous. Integer hours are indicated by tick marks and the time range of the frame is indicated. Use FLAG TIME or FLAG TIME RANGE to delete data following instructions which will appear on the message window. While you are editing, the source name, sample time and sample value currently selected will be displayed in the upper left corner of the TV screen.

Having noted all obviously bad points, you may now selectively flag visibility data. Select SWITCH ALL IF, SWITCH ALL TIME, SWITCH ALL ANT, SWITCH ALL POL, and SWITCH ALL SOURC so that the next flag command(s) apply to the portion of the data that should be flagged. The current setting of these toggles is displayed at the lower left corner. The various FLAG options allow you to flag single times, time ranges, all samples above or below a selected level, and single points either carefully or quickly. Finally, apply your flagging to your *uv* data set by selecting EXIT or throw it all away by selecting ABORT..

Note that it may also be profitable to run EDITA using the SY table. Unlike complex gains which reflect the values of all antennas, the SY values are strictly dependent only on the specific antenna. They are however affected by RFI in ways that may not affect the visibilities, so flagging *uv* data based on the SY values should be done with caution. EDITA, when run on SY or TY tables, has the option to display the difference of the current parameter with a running mean of that parameter. Such displays are a powerful editing tool.

4.3.12 Applying solutions to the calibration table with CLCAL

At this point you should have gain and phase solutions for the times of all calibration scans, including the correct flux densities for the secondary calibrators. The next step is to interpolate the solutions derived from the calibrators into the CL table for all the sources. CLCAL may be run multiple times if subsets of the sources are to be calibrated by corresponding subsets of the calibrators, unless you limit it to one or more tables with SNVER and INVERS, CLCAL assumes that all SN tables contain only valid solutions and concatenates all of the SN tables with the highest numbered one. Therefore, any bad SN tables should be removed before

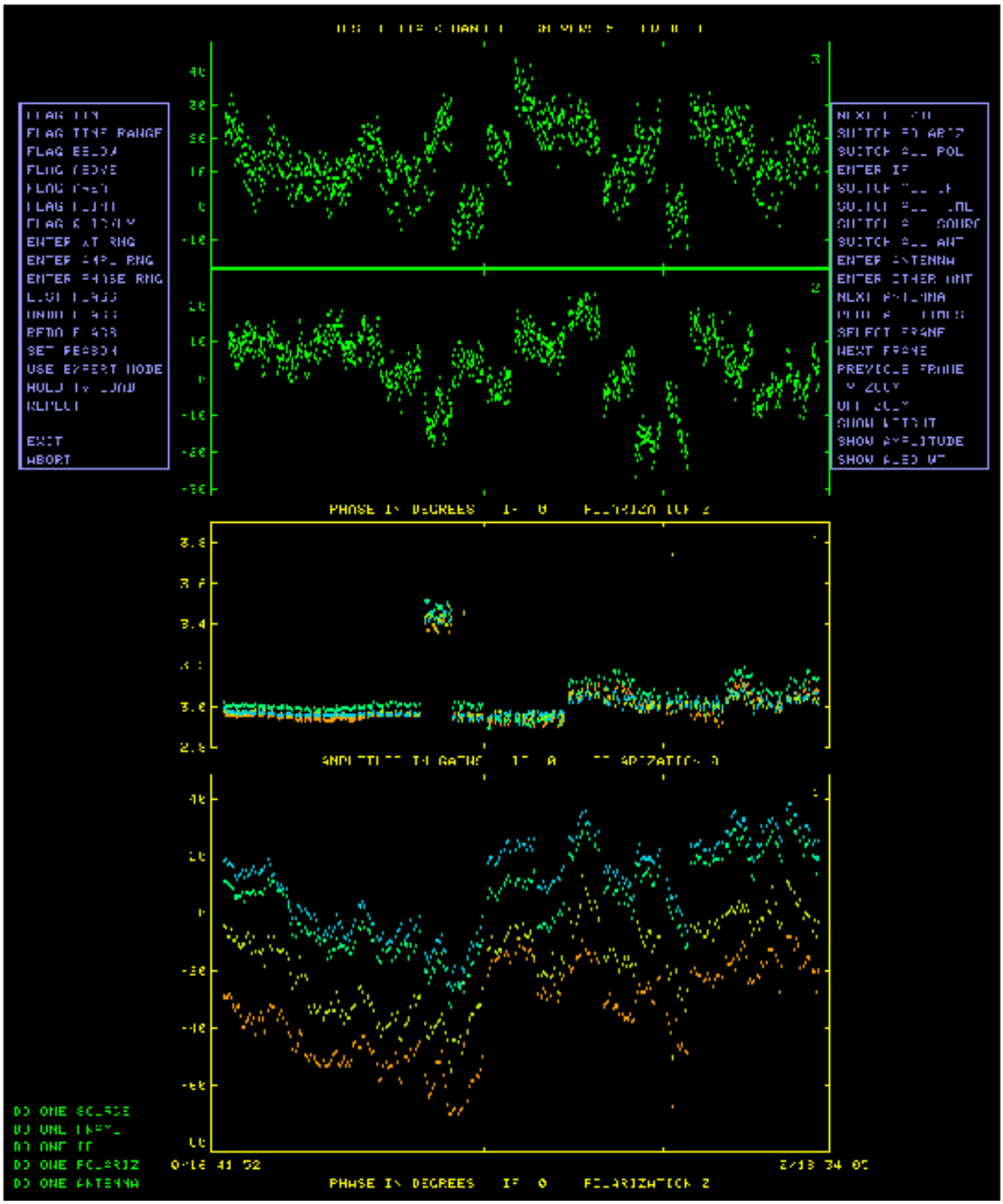


Figure 4.2: Sample TV screen from EDITA. Solution phases are being used to edit the uv data with amplitude shown for comparison. The EDITA menu (in the boxes), the status lines (at the bottom left), the editing area (bottom) of the data from the selected antenna (1), and the subsidiary plots of data from selected secondary antennæ (2, 3) are shown. When flagging, the edit tool (bar or box), and the edit location values are also displayed in different graphics planes which normally appear in different colors. In this example, with CROWDED=2, 2 polarizations and 2 IFs are displayed and may be edited simultaneously. Color is used to differentiate polarizations and IFs.

using CLCAL. For polarization calibration, it is essential that you calibrate the primary flux calibrator (3C48 or 3C286) also so that you can solve for the left minus right phase offsets and apply PCAL.

CLCAL has caused considerable confusion and user error because it implements two somewhat contrary views of its process. The older view, represented by previous versions of this *CookBook*, had the user gradually building a final CL table from multiple runs of CLCAL, each with a selected set of calibration sources, target sources, antennas, time ranges, and so forth. In this scheme, the user had to take great care that the final CL table actually contained information for all antennas, sources, and times for which it would be needed. It was easy to get this wrong! The second and now prevailing view is that every execution of CLCAL should write a new CL table containing all sources, antennas, and times, but with a selected subset modified by the current execution. This leads to there being a potentially large number of CL tables, but no data will be flagged due to the absence of data in the CL table. The user will still have to be careful to insure that all CL records have received the needed calibration information.

To use CLCAL:

> TASK 'CLCAL' ; INP ↵	to review the inputs.
> SOURCES '*sou1*' , '*sou2*' , '*sou3*' , . . . ↵	sources to calibrate, ' ' means all.
> CALSOUR '*cal1*' , '*cal2*' , '*cal3*' , . . . ↵	calibrators to use for SOURCES.
> FREQID *n* ↵	use FQ number *n*.
> OPCODE 'CALP' ↵	to combine SN tables into a CL table, passing any records not altered this time.
> GAINVER 0 ↵	to select the latest CL table as input.
> GAINUSE 0 ↵	to select a new output CL table.
> REFANT *m* ↵	to select the reference antenna; needed only if REFANT reset since CALIB was run.
> INTERP '2PT' ↵	to use linear interpolation of the possibly smoothed calibrations..
> SAMPTYPE ' ' ↵	to do no time-smoothing before the interpolation.
> SAMPTYPE 'BOX' ↵	to use boxcar smoothing, followed by interpolation.
> BPARM *n* , *n* ↵	to smooth, if BOX selected, with an *n*-hr long boxcar in amplitude and phase.
> DOBLANK 1 ↵	to replace failed solutions with smoothed ones but to use all previously good solutions without smoothing.
> INP ↵	to check inputs.
> GO ↵	to run CLCAL.

Calibrator sources may also be selected with the QUAL and CALCODE adverbs; QUAL also applies to the sources to be calibrated. Note that REFANT appears in the inputs because *AIPS* references all phases to those of the reference antenna. If none is given, it defaults to the one used in the most solutions.

The smoothing and interpolation functions in CLCAL have been separated into two adverbs and the smoothing parameters are now conveyed with BPARM and ICUT. In smoothing, the DOBLANK adverb is particularly important; it controls whether good solutions are replaced with smoothed ones and whether previously failed solutions are replaced with smoothed ones. One can select either or both.

Note that CLCAL uses both the GAINUSE and GAINVER adverbs. This is to specify the input and output CL table versions, which should be different. If you are building a single CL table piece by piece, then these must be set carefully and normally held fixed. In the more modern view, they are set to zero and the task takes the latest CL table as input and makes a new one. CL table version 1 is intended to be a "virgin" table, free of all injury from any calibration you do using the *AIPS* package. It may not always be devoid of information, as "on-line" corrections may be made and recorded here by some telescope systems, *e.g.*, the VLBA. The VLA, through tasks FILLM, BDF2AIPS, or INDXR, can now put opacity and antenna gain information in this

file. CLCAL and most other *AIPS* tasks are forbidden to over-write version 1 of the CL table. This protects it from modification, and keeps it around so that you may *reset* your calibration to the raw state by using EXTDEST to destroy all CL table extensions with versions higher than 1. Be careful doing this, since you rarely want to delete CL version 1. Should you destroy CL table version 1 accidentally, you may generate a *new* CL table version 1 with the task INDXR. This new CL table may contain the calibration generated from the weather and antenna gain files.

If you have any reason to suspect that the calibration has gone wrong — or if you are calibrating data for the first time — you should examine the contents of the output CL table. The task EVASN may help you determine the degree of phase and amplitude coherence in your calibration table. A lack of coherence suggests that the calibration is rather uncertain. Task SNPLT will provide you with a graphical display:

> DEFAULT SNPLT ; INP	to reset all adverbs and choose the task.
> INDI n; GETN m C_R	to select the data set on disk n and catalog number m.
> INEXT 'CL' C_R	to examine the latest calibration table.
> OPTYPE 'AMP' C_R	to examine amplitudes.
> OPTYPE 'PHAS' C_R	to examine phases (do both).
> NPLOTS 4 ; DOTV 1 C_R	to have 4 plots per page on the TV display.
> OPCODE 'ALIF' ; DO3COL 1 C_R	to display all selected IFs in each plot using colors from red to blue.
> INP C_R	to check the inputs.
> GO C_R	to examine your CL table.

Note that SNPLT may be used to examine SN, SY, and TY tables as well. OPCODE controls whether 1 IF or all IFs and 1 polarization or all polarizations are displayed. Task SNBPL will plot the calibration values on a baseline basis which can help to isolate the source of incoherence. SNFLG can be used to flag incoherent baselines rather than deleting whole antennas.

To test that the calibrators are actually properly calibrated by the new CL table, use UVPLT Try

> DEFAULT UVPLT ; INP	to reset all adverbs and choose the task.
> INDI n; GETN m C_R	to select the data set on disk n and catalog number m.
> SOURCES *'cala'* ' C_R	to select calibrator sources one at a time.
> DOCAL 1 ; DOBAND 1 C_R	to apply calibration.
> BCHAN $N/8$; ECHAN=$7N/8$ NCHAV=ECHAN-BCHAN+1 C_R	;to average the central channels of each IF.
> BPARM 11 , 2 C_R	to examine phase.
> BPARM 11 , 1 C_R	to examine amplitude.
> DOTV 1 C_R	to use the TV display.
> GO C_R	

You may need to limit which IFs are displayed or color the IFs with DO3COL 1. The phases should be near zero and the amplitudes should be nearly constant with baseline length. A decrease in the amplitude at the longest spacings indicates that the calibrator is not strictly a point source.

ANBPL converts baseline-based data before or after calibration into antenna-based quantities. In particular, the calibrated weights are very sensitive to problems with amplitude calibration.

> DEFAULT ANBPL C_R	to select task and initialize all its parameters.
> IND m ; GETN n C_R	to specify the multi-source data set.
> STOKES 'HALF' ; TIMERANG 0 C_R	to DISPLAY both parallel-hand polarizations.
> FREQID 1 C_R	to select FQ value to image.
> BIF 1 ; EIF 0 C_R	to select all IFs.
> BCHAN n ; ECHAN m C_R	to combine a range of channels.

> DOCALIB 1 C_R	to apply calibration.
> GAINUSE 0 C_R	to use highest numbered CL table.
> FLAGVER 1 C_R	to edit data.
> DOBAND 3 ; BPVER 1 C_R	to correct bandpass with time smoothing using table 1.
> BPARM 2, 17 C_R	to plot weight versus time.
> NPLOTS 3 ; DOTV 1 C_R	To plot 4 antennas per page on the TV.
> DOCRT 0 C_R	to suppress printed versions of the antenna-based values.
> INP C_R	to review the inputs.
> GO C_R	to run ANBPL.

If the previous steps indicate serious problems and/or you are seriously confused about what you have done and you want to start the calibration again, you can use the procedure VLARESET from the RUN file VLAPROCS to reset the SN and CL tables.

> INP VLARESET C_R	to verify the data set to be reset.
> VLARESET C_R	to reset SN and CL tables.

4.3.13 Flagging RFI with RFLAG

A very promising relatively new tool flags RFI on the assumption that it is either quite variable in time or in frequency. This task, called RFLAG, computes the rms over short time intervals in each spectral channel and IF individually and flags the interval whenever the rms exceeds a user-controlled threshold. Optionally, it will also use a sliding median window of user-specified width over the spectral channels to the real and imaginary parts of the visibility separately. Any channel deviating from the median in either part by more than a user-specified amount will also be flagged. If DOPLOT> 0, RFLAG will make plots of normal and cumulative histograms and of the mean and rms of the time and spectral computations as a function of channel. It will also make a flag table only if requested (DOFLAG> 0). These plots will suggest threshold parameters and allow you to choose values to use. A flag table is made for any value of DOFLAG if no plots are requested (DOPLOT≤ 0).

In detail, RFLAG is run using

> DEFAULT RFLAG ; INP C_R	to clear and review the adverbs.
> INDI n; GETN m C_R	to select the data set on disk n and catalog number m.
> SOURCES '*source_1*', '*source_2*', ... C_R	to select sources of similar flux level.
> DOCALIB 1 ; DOBAND 1 C_R	to apply continuum and bandpass calibration.
> STOKES 'FULL' C_R	to examine all polarizations.
> DOPLOT 15 ; DOTV 1 C_R	to examine all kinds of plots on the TV.
> FPARM 3 , x , -1, -1 C_R	to examine spectral rms over 3 time intervals each a bit longer than x seconds. The -1's cause the program to use other adverbs for the cutoffs and to do a spectral solution as well as the time one.
> FPARM(9) = 4.0 ; FPARM(10) = 4. C_R	to set the cutoff values as 4 times the median rms plus deviation found in the spectral plots as a function of IF. The default is 5.
> FPARM(13) x ; FPARM(17) y C_R	to flag all samples above x Jy and flag all of any channel with more than $0 \leq y \leq 1$ of the time samples flagged.
> FUNCTYPE 'LG' C_R	to plot the histograms on a log scale.
> NBOXES 1000 C_R	to use 1000 boxes in the histograms.
> INP C_R	to re-examine the inputs. VPARM will let you control aspects of the plotting.

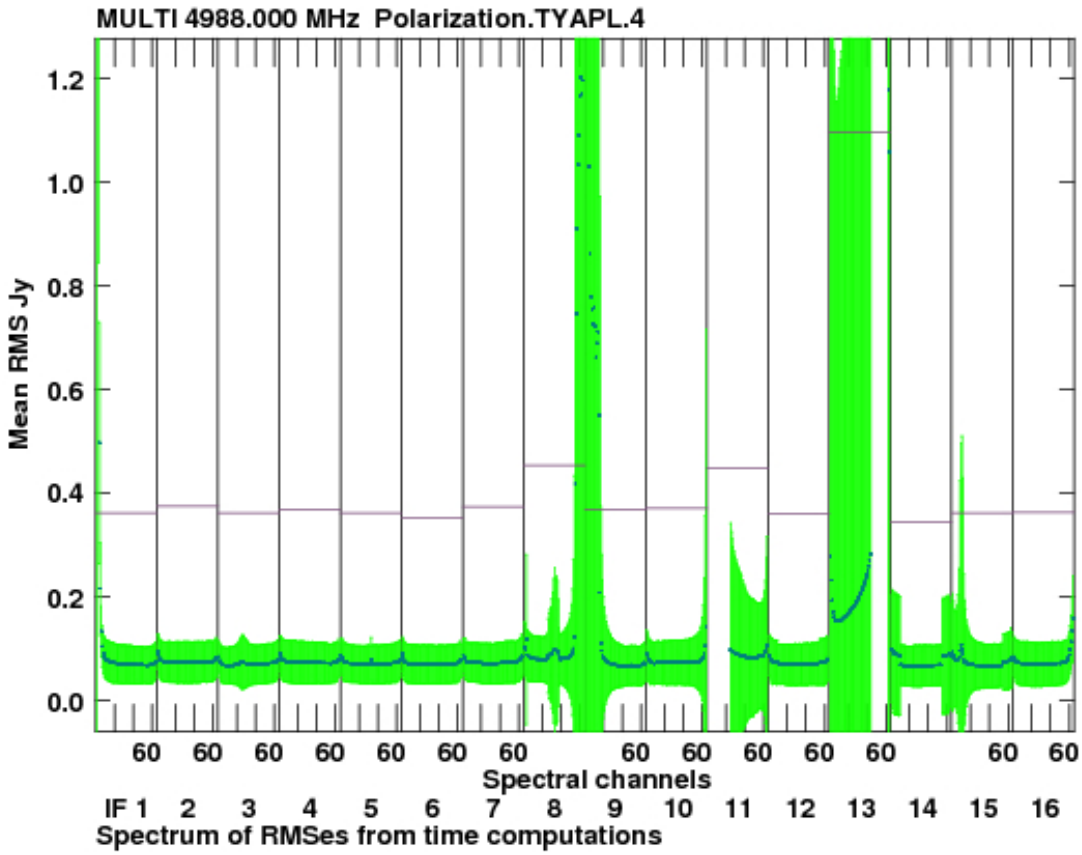


Figure 4.3: Plot produced by RFLAG showing the mean rms of short time intervals (dark line in middle of green) and the rms of the rmses (green vertical bars) as a function of spectral channel. The recommended flagging level, returned in adverb NOISE is shown by the horizontal bars in each spectral window.

> GO to run the program.

This will produce plots and set cutoff levels in adverbs NOISE and SCUTOFF. An example of the spectral plot is shown in Figure 4.3. Another run, with DOPLOT = 0 will apply these cutoffs and create a new flag table. Note that the flux cutoff levels may depend on the source flux, calling for different levels for strong calibrators, weak calibrators, and very weak target sources. Different cutoff levels for STOKES='RRLL' and STOKES = 'RLLR' may also be needed. A strong, resolved target source may require different levels for different UVRANGEs. UVRANGE is now applied on a per spectral window basis so you no longer have to break up wide bandwidth data into multiple files when UVRANGE is required. RFLAG is a new task, so experiment a bit. Note that, if you set DOFLAG=1, the creation of a new flag table will happen after the plots in the same execution of RFLAG. If a channel is found bad at a time in any one polarization, all polarizations are flagged. If you have a significant spectral line signal in your data, use DCHANSEL to have the affected channels ignored throughout RFLAG or set FPARM(4) = 0 so that the spectral function is not performed.

There are a lot of adverbs to RFLAG. FPARM(5) allows you to speed up the spectral part of the flagging by testing more than just the central channel in the sliding median filter. FPARM(6) allows you to expand all flags to adjacent channels. FPARM(7), (8), (11), and (12) control the extending of flags to additional channels, baselines, or antennas if too large a fraction of channels, baselines, or baselines to an antenna are flagged in the basic time and spectral operations. Similar adverbs also occur in the new task REFLG whose job it is to compress the enormous flag tables generated generated by RFLAG. REFLG does not handle flags generated by CLIP, TVFLG, and SPFLG since they vary with polarization. REFLG can extend a flag to all times if too large a fraction of time is flagged for a given channel, baseline, etc. REFLG may not correctly compress all flags, but its other options are useful. The application of 10 million flag entries to a data set repetitively is rather expensive. Copying the data, applying the flags once and for all, is the best solution. RFLAG may now do this; set DOOUTPUT to true in 31DEC22 and later, along with output name adverbs OUTNAME, *et al.* UVCOP has been the traditional method to do this. However, TYAPL which needs to be run next and must make a new copy of the data has been given the option of applying a large flag table to avoid having to copy the data set twice. Task FGCNT lets you see how much of your data is flagged by any particular flag table. Task FGPRT will plot a matrix by baseline of the number of flags applying in a user specified range of time, spectral channel and IF, source, etc.

4.3.14 Restart the calibration

Having done a more careful job with your editing, it is now time to discard with EXTDEST the bandpass (BP) tables and all CL tables after the one written by VLANT. Discard all SN tables, but keep the highest numbered flag (FG) table.

> INEXT = 'BP' ; INVERS = -1 ; EXTDEST ↵ to remove all bandpass tables.

> INEXT = 'SN' ; INVERS = -1 ; EXTDEST ↵ to remove all solution tables.

> INEXT = 'CL' ; FOR INVERS=3:100; EXTDEST;to remove all calibration tables after VLANT.

END ↵

Be very careful with the flag tables to save the last one - use IMHEADER to show you how many there are.

4.3.15 Calibration with the SysPower table

Because of its wide dynamic range, the EVLA does not normalize its output visibilities. To calibrate gains it records the total power when the switched noise tube is on and when it is off. These data, taken in synchronism with the visibilities, are recorded in the SysPower table of the ASDM. The OBIT program BDFIn, available to *AIPS* users in the verb BDF2AIPS, reads this table and creates an *AIPS* SY table. The columns of this table contain POWER DIF ($Gain \times (P_{on} - P_{off})$), POWER SUM ($Gain \times (P_{on} + P_{off})$), and POST GAIN ($Gain$) columns for right and left polarizations with values for each IF.

This table is accessible to *AIPS* users with a number of tasks. Begin with task PRTSY to view the table statistically over time on a per IF, per antenna basis or to view scan or source median averages of one or more of the SY table parameters. A more detailed look is available from SYPRT. Then to examine its contents graphically in various ways, use SNPLT with OPTYPEs 'PDIF','PSUM','PGN','PON','POFF','PSYS','PDGN', or 'PSGN'. You could use OPTYPE = 'MULT' to examine more than one of these at one time, comparing any oddities in *e.g.*, Psum and Pdif. Note that PSYS is especially interesting since $T_{cal} * P_{sum}/(2 * P_{dif}) = T_{sys}$, the system temperature. It should reflect changes in elevation and strength of the observed source, but should be immune to adjustments to the gain of the telescope. It determines data weights in TYAPL while $\sqrt{P_{dif_i} P_{dif_j}}$ divides into the visibilities. You may use EDITA (§ 4.3.11) to edit your uv data on the basis of the contents of the SY table. Editing may be based on Psum, Pdif, Pgain, Tsys, and on the differences between these parameters and a running median of these parameters. One may also edit the SY table itself with SNEDT; the same parameters are available. LISTR can even display the SY table Psum, Pdif, system temperatures, and gain factors with OPTYPE 'GAIN' and DPARM(1) set to 17, 18, 15, or 16, respectively.

In 31DEC25, a new experimental task SYHIS has become available. Unlike TYSMO, this task works on a per IF, per antenna, per polarization basis. It can plot histograms of the SY input table parameters. It can then clip the SY values, median-window filter the clipped data, plot histograms of the results, clip the median-window values, smooth the remaining values, and then replace the clipped samples (or all samples) with the smoothed values. These actionas are similar to those of TYSMO except that the task sets the clip levels separately for each antenna, IF, and polarization based on the values found in the histograms.

More importantly, the SY table can be used to do an initial calibration of the visibility data. Use the display programs to decide if your SY table is fine as is or needs editing. SNPLT may be the best way to examine The SY table. In 31DEC23, SNRMS may be used to determine the average values in the table and their rms. The following is somewhat controversial; there are users who believe that the SY table is never adequately reliable for all antennas because of RFI and other issues. Since the value of Tcal are not perfectly known, nor are the antenna efficiencies, the corrections made by TYAPL can never be perfect. Nonetheless, in many cases, they are a great help. CALIB will still be needed to finish the amplitude calibration.

The tasks TYSMO and TYAPL may be used with EVLA data having an SY table. TYSMO flags SY samples on the basis of Pdif, Psum, Pgain, and Tsys and then smooths Psum, Pdif and Pgain to replace the flagged samples and/or reduce the noise. You *will* want to do this to remove outlying bad points and to reduce the jitter in these measurements. TYSMO even applies a flag table to the SY table before its clipping and smoothing operations. Be sure to plot the results to make sure that the task did what you wanted. It is both powerful and very helpful. Then use TYAPL to remove a previously applied SY table (if any) and to apply the SY table you have prepared. The result should be data scaled nearly correctly in Jy and weights in $1/\mathrm{Jy}^2$ in all IFs. The CUTOFF option allows you to use obviously good values from the SY table while passing the data from antennas with poor SY values along unchanged. If the SY values from some polarizations or IFs are bad due to RFI while others are good, you may copy the good values to replace the bad values using task TYCOP. The wide-band, 3-bit mode of the EVLA has serious non-linearities in the Pdif measurements which are not yet fully understood. For such data, use OPTYPE='PGN' to apply only the post-detection gains to the data. This will remove abrupt jumps due to changes in those gains (which will be more common with 3-bit data). Note that TYSMO and TYAPL also require a table of the Tcal values which OBIT provides in an *AIPS* CD table. Amplitude calibrations are not applied to EVLA data weights until they have been made meaningful by TYAPL or REWAY. Set FLAGVER in TYAPL if you want to apply your flag table once and for all. TYAPL can handle flag tables much larger than those that many other tasks can handle. Note that TYAPL must copy the data set since it makes corrections to the antennas on very short intervals (not suitable to a CL table).

If you have observations of the Sun, do not use TYAPL; use SYSOL instead. This task does the usual TYAPL operation on non-Solar scans, but, for Solar scans, it must determine the average gain and weight factors on those antennas having solar Tcals and then apply those averages to all antennas not equipped with Solar Tcals. Use a well-edited SY table if possible. This Solar capability was made available in OBIT (version 567) and *AIPS* in mid-June 2017. The revised format of the SY table includes a column to tell tasks whether to use normal or solar Tcals for that row of the table.

4.3.16 Calibrating polarization

You may skip this section unless you have cross-hand polarization data and wish to make use of them. Although there have been major improvements in *AIPS* polarization routines, they still do not correct parallel hand visibilities for polarization leakage. Thus you need to calibrate polarization only if you wish to make images of target source Q and U Stokes parameters. Note that we discuss here the calibration of both linear and circular polarization with tasks TECOR, RLDLY, and PCAL. At the phase difference stage, however, the process for linear feeds is different than for circular feeds.

The calibration of wide-band visibility data sensitive to polarization involves three distinct operations: (1) determining and correcting the offset in delay between the two parallel-hand polarizations, (2) determining and correcting the data for the effects of imperfect telescope feeds, and (3) removing any systematic phase offsets between the two systems of orthogonal polarization. These three components of polarization calibration will be considered separately.

Faraday rotation in the Earth's ionosphere can add a phase difference between the two parallel-hand polarizations which is both time and direction dependent. These are particularly important at lower frequencies (< 4GHz), but may rarely affect higher frequency-data as well. One way to remove at least some of the ionospheric phase offsets is by applying a global ionospheric model derived from GPS measurements. The *AIPS* task TECOR processes such ionospheric models that are in standard format known as the IONEX format. These models are available from the Crustal Dynamics Data Information System (CDDIS) archive. There is a procedure which is part of VLAPROCS, called VLATECR that automatically downloads the needed IONEX files from CDDIS and runs TECOR. It will examine the header and the NX table and figure out which dates need to be downloaded, so the observation date in the header must be correct and an NX table must exist. See EXPLAIN VLATECR for other requirements. The default TECRTYPE is jplg which seems to work well for the VLA.

> RUN VLAPROCS ↵	to acquire the procedures; this should be done only once since they will be remembered.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> INP VLATECR ↵	to review the inputs.
> APARM 1,0 ↵	to apply dispersive delay corrections which may be useful for wide bandwidths. The remaining APARM values appear to work well for the VLA when left at 0.
> VLATECR ↵	to run the procedure.

TECOR applies its corrections to the highest numbered CL table and writes a new one. A significant bug fix was made in January 2023 and applied retroactively to the 31DEC21 and 31DEC22 releases. TECOR now writes a "TE" table which may be plotted with TEPLT to examine all details of this correction. The new task MFIMG may also be used to make an image of the TEC data or the model used for the Earth's magnetic field in TECOR. All standard image displays may then be used to view the results.

Extensive studies have revealed that TECOR usually over-corrects VLA data. It removes the time-dependent IFRM but leaves behind an extra rotation measure that changes some with obsering date and very slightly with source. A package of software called ALBUS has been found to give somewhat better solutions for the IFRM. In 31DEC24 a task also named ALBUS was written to invoke this package to correct uv data including dispersive delay which is important to VLBI data. Unfortunately, the complications of this package require the Linux operating system and an outside program called apptainer. See EVLA Memo 235 for details.[2]

Frequently, the delay difference between right- and left-hand, or V and H, polarizations must be determined even if FRING was not required for the parallel-hand data. Use POSSM to plot the RL and LR (or VH and HV) spectra to see if there are significant slopes in phase. If so, use a calibration source with significant polarization, although the EVLA D terms are often large enough to provide a usable signal in the absence

[2] "Testing Ionospheric Faraday Rotation Measure Models", Perley R., Greisen, E., Tremou, L., Butler, B., Willis, T., October 2024

of a real polarized signal. Note that 3C286 is significantly polarized and is likely to be the best source to use for this purpose. Then

> TASK 'RLDLY' ; INP ↵ — to look at the necessary inputs.

> BCHAN c_1 ; ECHAN c_2 ↵ — to select channels free of edge effects.

> DOCAL 1 ; GAINUSE 0 ↵ — to apply the FRING results and all other current calibrations.

> REFANT n_r ↵ — to select a reference antenna - only baselines to this antenna are used so select carefully. Alternatively, REFANT 0 will loop over all possible (not necessarily good) reference antennas, averaging the result.

> DOIFS j ↵ — to set the adverb to the value of APARM(5) used in FRING (§4.3.6. The IFs are done independently (≤ 0), all together ($= 1$), in halves ($= 2$), or more generally in N groups ($= N$). A value of -1 causes RLDLY to use BPARM to define the IF groupings, as in FRING.

> TIMERANG *db* , *hb* , *mb* , *sb* , *de*, *he* , *me* , *se* ↵ — to specify the beginning day, hour, minute, and second and ending day, hour, minute, and second (wrt REFDATE) of the data to be included. Use an interval not unlike the one you used in FRING.

> INP ↵ — to check the inputs.

> GO ↵ — to produce a new SN table with a suitable left polarization delay.

Note that RLDLY now always creates an SN table and may be run with multiple calibrator scans. If there is only one calibrator scan and APARM(2)≤ 0, it will also copy the CL table which was applied to the input data through GAINUSE to a new CL table applying the correction to the L polarization delay. For all other cases, you must apply the added L polarization delays with CLCAL.

To begin with, it is probably better to determine a continuum solution for source polarization and antenna D terms before doing the lower signal-to-noise spectral solutions. To find an average solution for each IF:

> DEFAULT 'PCAL' ; INP — to reset all adverbs and choose the task.

> INDI n; GETN m ↵ — to select the data set on disk n and catalog number m.

> DOCAL 1 ; DOBAND 3 ↵ — to apply the delay, complex gain, and bandpass calibration.

> CALSOUR '*pol_cal1*', '*pol_cal2*' ↵ — to select the polarization calibrator(s) by whatever form of their names appears in your LISTR output. These sources must have I polarization fluxes in the source table.

> ICHANSEL $c11, c12, 1, if1, c21, c22, 1, if2, c31, c32, 1, if3, \ldots$ ↵ — to select the range(s) of channels which are reliable for averaging in each IF. These probably should be the same values that you used in BPASS.

> DOMODEL -1; SPECTRAL -1 ↵ — to solve for source polarization in a continuum manner.

> PRTLEV 1 ↵ — to see the answers and uncertainties on an antenna and IF basis.

> CPARM 0,1 ↵ — to update the source table with the calibrator source Q and U found.

> INP ↵ — to review the inputs.

> GO ↵ — to find the antenna leakage terms and the source Q and U values on an IF-dependent basis.

PCAL will write the antenna leakage terms in the antenna file and the source Q and U terms in the source table (if CPARM(2) > 0). DOMODEL may be set to true only if the model has Q $=$ U $= 0$ since PCAL cannot solve for the right minus left phase difference. If SOLINT=0, PCAL will break up a single scan into multiple

intervals, attempting to get a solution even without a wide range of parallactic angles.

Use TASAV to copy all your table files to a dummy *uv* data set, saving in particular the CL table with the results of the amplitude and phase calibration. This step is not essential, but it reduces the magnitude of the disaster if the the next step fails in some way. (Note - this may be a good idea at several stages of the calibration process!)

> TASK 'TASAV' ↵
> CLRO ↵ Use default output file file name.
> INP ↵ to review the (few) inputs.
> GO ↵ to run the program.

The task TACOP may be used to recover any tables that get trashed during later steps. Note that the next step writes a new CL table, but changes the AN and SU tables in place.

Having prepared a continuum solution for Q and U, you must also correct it for the difference in phase between R and L (or V and H) polarizations which normally varies considerably between IFs. For circularly polarized feeds only, the task RLDIF will correct the antenna, source, and calibration tables for this difference using observations of a source with known ratio of Q to U. 3C286 is by far the best calibrator for this purpose.

> DEFAULT RLDIF ; INP to reset all adverbs and choose the task.
> INDI n; GETN m ↵ to select the data set on disk n and catalog number m.
> DOCAL 1 ; DOBAND 3 ↵ to apply the delay, complex gain, and bandpass calibration.
> DOPOL 1 ↵ to apply the polarization calibration.
> BCHAN c_1; ECHAN c_2 ↵ to average data from channels c_1 through c_2 only.
> SOURCES '*pol_cal1*', '*pol_cal2*' ↵ to select the polarization calibrator(s) by whatever form of their names appears in your LISTR output. These sources must have known polarization angles.
> SPECTRAL 0 ↵ to do the correction in continuum mode.
> DOAPPLY 1 ↵ to apply the solutions to a CL table (making a new modified one) and to the AN and SU tables, updating them in place.
> DOPRINT 0 ↵ to omit all the possible printing.
> INP ↵ to review the inputs.
> GO ↵ to determine and apply the corrections.

The EVLA polarizers appear to be very stable in time, but to have significant variation with frequency. See Figure 4.4. Serious polarimetry with the EVLA will *require* solving for the antenna polarization leakage as a function of frequency. To compute a spectral solution, assuming you already did the process in the preceding paragraph:

> TGET PCAL ↵ to retrieve the PCAL adverbs.
> SPECTRAL 1 ↵ to do the channel-dependent mode.
> DOMODEL 0 ↵ to solve for Q and U as a function of frequency. Because PCAL does not solve for a right-left phase difference and that difference is a function of spectral channel, you must solve for a source polarization.
> SPECPARM 0 ↵ to determine the calibration source I, Q, and U spectral indices from fluxes in the source table. If you use PMODEL you must provide spectral indices for the model that apply in the frequency range of the data (curvature cannot be specified).
> INTPARM p_1, p_2, p_3 ↵ to smooth the data after all calibration has been done while honoring ICHANSEL.
> INP ↵ to review the inputs, the task will take a while to run.

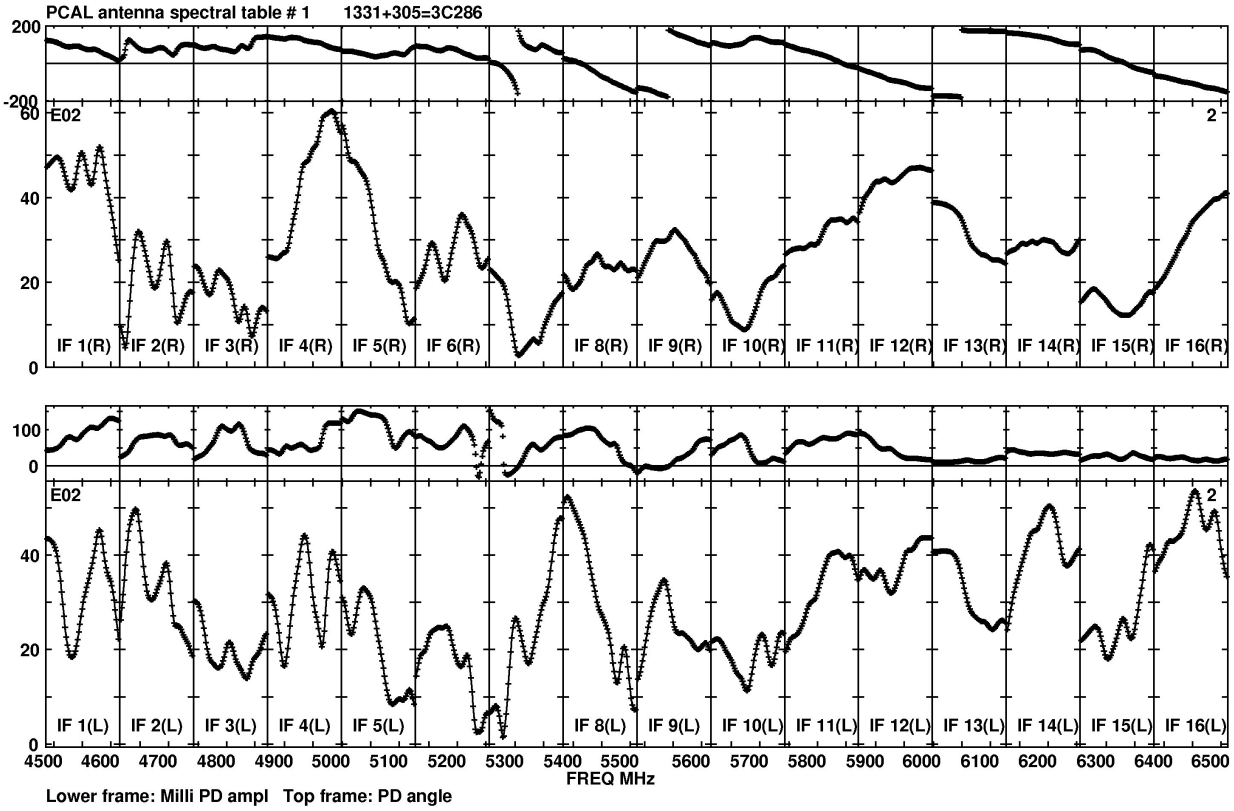


Figure 4.4: Example spectrum showing D term solutions for one antenna in right and left polarizations covering about 2 GHz at C band

> GO ↵ to run the task writing a PD table of spectral leakages ("D terms") and, if DOMODEL ≤ 0, a CP table of source Q and U spectra.

If the combination of flagging, ICHANSEL, and INTPARM results in no solutions for some channels, the solutions from nearby channels will be interpolated or extrapolated so that all channels get solutions. If the calibration source is known to have no polarization, then you may set DOMODEL 1 and set PMODEL = *flux* , 0. Note that PCAL does not solve correctly for polarization when run on linearly-polarized data. You must use an unpolarized calibration source on such data sets. See below for a new way to get out of this problem.

After running PCAL in spectral mode, you may examine the resulting PD (polarization D terms) table with POSSM using APARM(8)=6 and BPLOT using INEXT = 'PD'. To compare two or more PD tables, use task PDPLT in 31DEC20. If a CP table (calibrator polarization) was written, you may also use POSSM with APARM(8) = 7 or 8 and BPLOT with INEXT = 'CP' to examine the results.

You are almost, but not quite done. The combination of CALIB and BPASS has produced a good calibration for everything except the phase difference between right and left polarizations. This is now a function of spectral channel and needs to be corrected. The task RLDIF has been modified to determine a continuum or spectral right minus left phase difference and to modify the CL or BP table, respectively, to apply a phase change to the left polarization on an IF or channel, respectively, basis. Thus

> DEFAULT RLDIF ; INP to reset all adverbs and choose the task.

> INDI n; GETN m ↵ to select the data set on disk n and catalog number m.

> DOCAL 1 ; DOBAND 3 ↵ to apply the delay, complex gain, and bandpass calibration.

> DOPOL 1 ↵	to apply the polarization calibration, spectral if present.
> INTPARM p_1, p_2, p_3 ↵	to smooth the data after all calibration has been done.
> BCHAN c_1; ECHAN c_2 ↵	to use solutions from channels c_1 through c_2 only, extrapolating solutions to channels outside this range.
> SOURCES '*pol_cal1*', '*pol_cal2*' ↵	to select the polarization calibrator(s) by whatever form of their names appears in your LISTR output. These sources must have known polarization angles.
> POLANGLE p_1, p_2 ↵	to provide the task with the source polarization angle(s) in degrees in source number order. The phase correction will be twice this value minus the observed RL phase. Do not provide values for 3C286, 3C147, 3C48, and 3C138. These are known to RLDIF including rotation measures and other spectral dependence.
> ROTMEAS r_1 ↵	to provide in 31DEC23, the rotation measure in radians per meter squared for the first calibration source.
> SPECTRAL 1 ↵	to do the correction in spectral mode.
> DOAPPLY 1 ↵	to apply the solutions to a BP table (making a new modified one) and to the PD and CP tables.
> DOPRINT -1 ; OUTPRI '*file_name*' ↵	to write the phase corrections applied to a text file suitable for plotting by PLOTR.
> INP ↵	to review the inputs.
> GO ↵	to determine and apply the corrections.

The right minus left phase difference may be a modest (we hope) function of time. To have RLDIF solve for this, set adverbs DOKEEP = 1 to invoke this option, SOLINT to an averaging time in minutes, and ICHANSEL as needed. The applied PP table will still be the global average solution, but individual time interval solutions will be written to the PP table and may be plotted with POSSM. An SN table is written so that the channel average right minus left phase may be applied as a function of time and IF to the CL table if desired.

Note that RLDIF applies to circularly polarized feeds only and requires you to know the linear polarization angle of at least one calibration source. The methods for calibrating linearly-polarized feeds, which are only at P- and 4-bands on the VLA, are still being developed. It appears that one requires one or more calibration sources known to have polarization in Stokes U and to have no emission in Stokes V (circularly-polarized emission). If you have such data, then the new task VHDIF performs the functions of RLDIF for linear feeds. 3C345 is such a calibration source, but at P-band its rotation measure makes the U polarization very weak at some frequencies while it is strong enough at others. Task PPAPP may be used to apply SPECTRAL > 0 tables to the BP tables of additional data sets.

In 31DEC24, a new task VH2RL may help in calibrating linear polarization in *AIPS*. It translates linear polarization data into circular. In order to do this, there are a number of requirements. There must be all 4 correlations in the data which must be calibrated by at least BPASS in its unnormalized mode (or use CALIB also), RLDLY, and VHDIF. Use an unpolarized source for BPASS, CALIB, and RLDLY; there are many at P-band. After applying VH2RL, you may then calibrate with additional, linearly-polarized sources and run PCAL and RLDIF in the normal way.

4.3.17 Additional calibrations for EVLA data

Task SNP2D will help calibrate data sets which contain a strong, *e.g.*, maser, line in one or a few channels with interesting but weaker signals in other channels and/or the continuum. After applying the best standard

calibration described above, you can split out the maser channel(s) with UVCOP. Make images and self-calibrate following the processes described in Chapter 5 making an SN table containing the full calibration needed to apply to the maser. Then use task SNP2D to convert this into delays and phases to apply to the full, multi-IF data set. This will work so long as the residual phases found in the self-cal are small and is actually required to do the best calibration possible over very wide bandwidths.

For the historic VLA, FILLM was able to compute meaningful data weights which were then improved with the amplitude calibration. For the EVLA, however, the data weights begin as solely the integration time. For 8-bit data, TYAPL (§ 4.3.15) is able to convert these into meaningful weights. However, if TYAPL was used only with OPTYPE = 'PGN', or not used at all, then it behooves you to rectify the situation. There is a task called REWAY which computes a robust rms over spectral channels within each IF and polarization. It can simply base the weights on these on a record-by-record, baseline-by-baseline basis. Alternatively, it can use a scrolling buffer in time so that the robust rms includes data for a user-specified number of records surrounding the current one. A third choice is to average the single-time rmses over a time range and then convert them to antenna-based rmses. In all three modes, the task can then smooth the rmses over time applying clipping based on user adverbs and the mean and variance found in the rmses. A flag table (extension file) may be written to the input data file removing those data found to have rmses that are either too high or too low. For these weights to be meaningful, the bandpass and spectral polarization calibration must be applied and it helps to omit any RFI or other real spectral-line signal channels from the rms computation. For the weights to be correctly calibrated, all amplitude calibration must also be applied. For these reasons, REWAY might well be used instead of SPLIT — when TYAPL was not used — running it one source at a time. Thus,

> DEFAULT REWAY ; INP	to reset all adverbs and choose the task.
> INDI n; GETN m ↵	to select the data set on disk n and catalog number m.
> DOCAL 1 ; DOBAND 3 ↵	to apply the delay, complex gain, and bandpass calibration.
> SOURCES '$target_1$' , ' ' ↵	to do one target source.
> APARM 11, 30, 12, 0, 10, 4 ↵	to use a rolling buffer of 11 times separated by no more than 30 seconds and then smoothed further with a Gaussian 12 seconds in FWHM. Data are flagged if the rms is more than 4 times the variance away from the mean averaged over all baselines, IFs, and polarizations. Flagging on the variance of the rms from the mean on a baseline basis is essentially turned off by the 10.
> GO ↵	to write out a calibrated, weighted data set for the first target source.

Then, when that finishes

> SOURCES '$target_2$' , ' ' ; GO ↵	to do another target source.

It is not clear that this algorithm is optimal, but it certainly should be better than using all weights 1.0 throughout. It will be interesting to compare data weights found with TYAPL to those found with REWAY. The task now displays statistical information to assist the user in determining which weights are excessively high or low.

With the historic VLA, spectral-line observations were normally done with frequencies which changed with time in order to track a chosen source velocity at a specific spectral channel. The narrow bandwidths of the historic VLA kept this operation out of non-linear regimes. The EVLA, however, has very wide bandwidths, making the appropriate frequency changes different for each spectral window. For spectral imaging, however, this frequency shift must be done, at least for narrow bandwidths, long observing sessions, or combinations of multiple observing sessions. It is probably best to separate out the spectral window (IF) containing the desired spectral line signals and do the following on that window alone.

At present, the EVLA observing setup allows you to select the initial frequency of observation based on a desired LSRK velocity in the central channel. From there, however, the observations are conducted

at a fixed frequency. Furthermore, the information about rest frequencies, source velocities, and even more fundamental parameters such as reference frame (LSRK or barycentric) and type of velocity (radio or optical) are lost. SETJY allows you to correct this. First use SETJY to set the desired rest frequencies (note that they are allowed to be a function of IF) and the VELTYP and VELDEF. Then use OPTYEP='VCAL' over all sources and IFs. This will compute the velocities at which you observed for the first time you observed each source and enter the values in the source table.

Having done this, the task CVEL may be used to shift the visibility data to correct for the rotation of the Earth about its axis as well as the motion of the Earth about the Solar System barycenter and the motion of the barycenter with respect to the Local kinetic Standard of Rest. CVEL works on multi-source as well as single-source data sets. It applies any flagging and bandpass calibration to the data before shifting the velocity (which it does by a carefully correct Fourier transform method). Note, the use of Fourier-transforms means that one *must not* use CVEL on data with channel separations comparable to the widths of some of the spectral features. Furthermore, narrow EVLA bands apparently have sharp cutoffs at the edges which cause any continuum signal to generate sine waves in amplitude after the FFT. Therefore, UVLSF *must be run before* CVEL. The velocity information used by CVEL must be correct. Use LISTR and SETJY to insure this before using SPLIT and CVEL. A special version of CVEL has been written to correct not only for the Earth's motion but also for planetary motion to observe a line at rest with respect to a planet; see PCVEL.

4.3.18 Making images from multi-source data with IMAGR

IMAGR can be used to make images from multi-source data files. It is probably a good idea to make a couple of quick images to make sure that the calibration is okay. An example set of inputs to IMAGR for a continuum image is:

> DEFAULT IMAGR ; INP C$_R$	to select task and initialize all its parameters. This selects the usual convolution and weighting functions among other things.
> IND m ; GETN n C$_R$	to specify the multi-source data set.
> SRCNAME '*sou1*' , ' ' C$_R$	to choose one source to image.
> STOKES 'I' ; TIMERANG 0 C$_R$	to image total intensity from all times.
> FREQID 1 C$_R$	to select FQ value to image.
> BIF 1 ; EIF 0 C$_R$	to image all IFs — multi-channel mode images only one IF.
> BCHAN n ; ECHAN m C$_R$	to combine a range of channels.
> NCHAV $(m-n+1)$ C$_R$	to include all spectral channels plus IFs *bif* through *eif* in each image. Note that each channel and IF included in the "average" image is handled individually at its correct frequency.
> DOCALIB 1 C$_R$	to apply calibration. Use DOCAL 100 C$_R$ if the weights should *not* be calibrated.
> GAINUSE 0 C$_R$	to use highest numbered CL table.
> FLAGVER 1 C$_R$	to edit data.
> DOPOL TRUE C$_R$	to correct for feed polarization if you solved for it.
> DOBAND 3 C$_R$	to correct bandpass with time smoothing.
> BPVER 1 C$_R$	to select BP table to apply.
> OUTNAME '*sou1*' C$_R$	to set the output file name to the source name.
> OUTDISK 0 C$_R$	to use any output disk with enough space.
> IMSIZE 512 512 C$_R$	to set the size in cells of image.
> CELLSIZE 0.25 , 0.25 C$_R$	to set the size of each image cell in arc-seconds.

> RASHIFT 0 ; DECSHIFT 0 ↵ to (not) shift image center.
> NFIELD 1 ; NGAUSS 0 ↵ to make only one image at high resolution.
> UVWTFN ' ' ↵ to use uniform weighting.
> ZEROSP 0 ↵ to introduce no zero-spacing flux.
> NITER 0 ↵ to do no Cleaning.
> INP ↵ to review the inputs.
> GO ↵ to run IMAGR when the inputs are set correctly.

Note that some of the above values are set by DEFAULT but we wish to emphasize the parameters here. The CELLSIZE and IMSIZE depend on your data; use SETFC to make recommendations.

4.3.19 Back up your data with FITTP or FITAB

After you have edited your target sources with RFLAG and checked the calibration with IMAGR, your calibration is complete — at least until you do self-calibration. It is probably time to back up your data. You may wish to apply the final large flag table from RFLAG before backing up the data. Use UVCOP to apply especially large flag tables, but be sure to copy all of your data otherwise. Then

> DEFAULT FITTP ; INP ↵ to initialize the adverbs.
> IND *m* ; GETN *n* ↵ to specify the multi-source data set.
> DATAOUT '*logical:filename* ↵ to specify the output disk file name in the usual manner; see § 3.10.3
> DOPLOT 1 ↵ to include plot files.
> INP ↵ to review the inputs.
> GO ↵ to write a "uvfits" file on disk.

Task FITAB writes a variant of this format in which the visibility data are also written in a FITS table format. The FITAB form is preferred esthetically by some, but most other software packages cannot read it properly. The FITS format used for plot files in 31DEC24 will not be useful to any sotware other than *AIPS*. The FITS format used by *AIPS* is extensively documented in *AIPS* Memo 117.[3]

4.3.20 Creating single-source data files with SPLIT

When you are happy with the and editing represented by the current set of calibration and flag tables, you can convert the multi-source file into single-source files, applying your calibration and editing tables. Remember that only one FREQID can be SPLIT at a time.

> TASK 'SPLIT' ↵
> SOURCES '*sou1*' , '*sou2*' , . . . ↵ to select sources, ' ' means all.
> TIMERANG 0 ↵ to keep all times.
> BIF 1 ; EIF 2 ↵ to keep both IFs
> FREQID 1 ↵ to set the one FQ value to use.
> DOCALIB 1 ↵ to apply calibration to the data and the weights.
> GAINUSE 0 ↵ to use the highest numbered CL table.
> DOPOL TRUE ↵ to correct for feed polarization, but only if you solved for the polarization in § 4.3.16.
> DOBAND 3 ↵ to correct bandpass with time smoothing.
> BPVER 1 ↵ to select BP table to apply.

[3] Eric W. Greisen, 2020, "*AIPS* FITS File Format," *AIPS* Memo 117, NRAO, with numerous updates

> STOKES '' C_R to write the input Stokes type.
> DOUVCOMP FALSE C_R to write visibilities in uncompressed format.
> APARM 0 C_R to avoid channel averaging and autocorrelation data.
> INP C_R to review the inputs.
> GO C_R to run the program when inputs set correctly.

The files produced by this process should be completely calibrated and edited and ready to be imaged or further processed as described in later chapters. Note that one may wish to defer the DOPOL 1 part of this until after self-calibration of the parallel-hand visibilities. When finally applying the polarization calibration, the option of applying a rotation measure to the cross-hand data is offered. This will make the cross-hands not represent what was measured, but instead represent a value at frequency 0. Removing a known rotation measure, however, then allows averaging over frequency in Q and U for better signal-to-noise.

4.4 Concluding remarks, early science

AIPS itself, and particularly this chapter, do not begin to cover all of the issues that will arise with EVLA data. The increased sensitivity of the EVLA will mean that imaging will no longer be able to ignore effects that are difficult to correct such as pointing errors, beam squint, variable antenna polarization across the field, leakage of polarized signal into the parallel-hand visibilities, etc., etc. These are research topics which may have solutions in *AIPS* or other software packages such as OBIT and CASA eventually.

4.5 Additional recipes

4.5.1 Chewy banana split dessert

1. Prepare and bake one package (19.8 0z) chewy fudge (or other favorite) **brownie mix**. Allow to cool thoroughly, four hours or more.
2. Peel 2 large ripe **bananas** and place very thin slices on top of brownie.
3. Cover bananas evenly with one 12-oz. container of **whipped topping** (thawed) and drizzle 1/2 cup **chocolate syrup** over that.
4. Refrigerate to chill completely. Cut into squares to serve.

4.5.2 Mexican bananas

1. Mix together 1 cup **sugar**, 1 teaspoon **cinnamon**, 1/8 teaspoon **nutmeg**, and 1/8 teaspoon **ginger**.
2. Peel 6 firm **bananas**, cut in half lengthwise, and brush with 1/4 cup **lemon juice**.
3. Place a banana half at end of each of 12 **tortillas** and sprinkle with sugar mixture.
4. Roll tortillas, brush top and sides with 1/4 cup **evaporated milk**, and then sprinkle with remaining sugar mixture.

4.5.3 Sopa de Plátano

1. Cook 10 whole red-skinned under-ripe **bananas** in one quart of water over low heat.
2. Peel and mash bananas with 1/4 teaspoon **cloves**, 1/4 teaspoon **orégano**, and 1 teaspoon **powdered cinnamon**.
3. Knead the mixture, add a pinch of **salt**, and fry in 4 tablespoon **shortening** until slightly browned.
4. Chop 4 medium-sized **tomatoes**, 2 **green peppers**, and 1 medium-sized **onion**.
5. Fry vegetables in 1/4 cup **olive oil** about 5 minutes and then add 1 teaspoon **salt**.
6. Place banana mixture on serving dish and garnish with the hot vegetables.

Thanks to Ruth Mulvey and Luisa Alvarez *Good Food from Mexico*.

4.5.4 Almond fudge banana cake

1. Mash 3 extra-ripe **bananas** to make 1 1/2 cups.
2. Beat 1 1/2 cups **sugar**, and 1/2 cup softened **margarine** until light and fluffy. Beat in 3 **eggs**, 3 tablespoons **amaretto liqueur** (or 1/2—1 teaspoon **almond extract**), and 1 teaspoon **vanilla extract**.
3. Combine 1 1/3 cups **all-purpose flour**, 1/3 cup unsweetened **cocoa powder**, 1 teaspoon **baking soda**, 1/2 teaspoon **salt**, and 1/2 cup toasted **chopped almonds**.
4. Add dry mixture and bananas alternately to beaten mixture. Beat well.
5. Turn batter into greased 10-inch bundt pan. Bake in 350° F oven 45 to 50 minutes or until toothpick inserted in center comes out nearly clean and cake pulls away from sides of pan. Cool 10 minutes. Remove cake from pan to wire rack to cool completely.
6. Puree 1 small **banana** and beat into 1 ounce (1 square) melted **semisweet chocolate**. Drizzle this glaze over top and down sides of cooled cake.

4.5.5 Delightful banana cheesecake

1. Preheat oven to 350° F.
2. Combine 1.5 cups crushed **cereal** (3 cups un-crushed Multi-Bran Chex suggested), 1/3 cup melted **margarine** or butter, and 1/4 cup packed **brown sugar**; mix well.
3. Press firmly onto bottom and sides of greased 9-inch pie plate. Bake 8–10 minutes, then cool completely.
4. Arrange 1.5 cups sliced **bananas** onto sides and bottom of cooled crust.
5. Combine 16 oz. softened light or regular **cream cheese**, 1.5 cups **powdered sugar**, and 3/4 teaspoon **vanilla extract**.
6. Mix well, then fold in 2 cups light or regular **whipped topping**. Pour over sliced bananas.
7. Cover and refrigerate for 4 hours or until set.
8. Garnish with 1/2 cup sliced **bananas**.

Thanks to Ralston Purina Company.

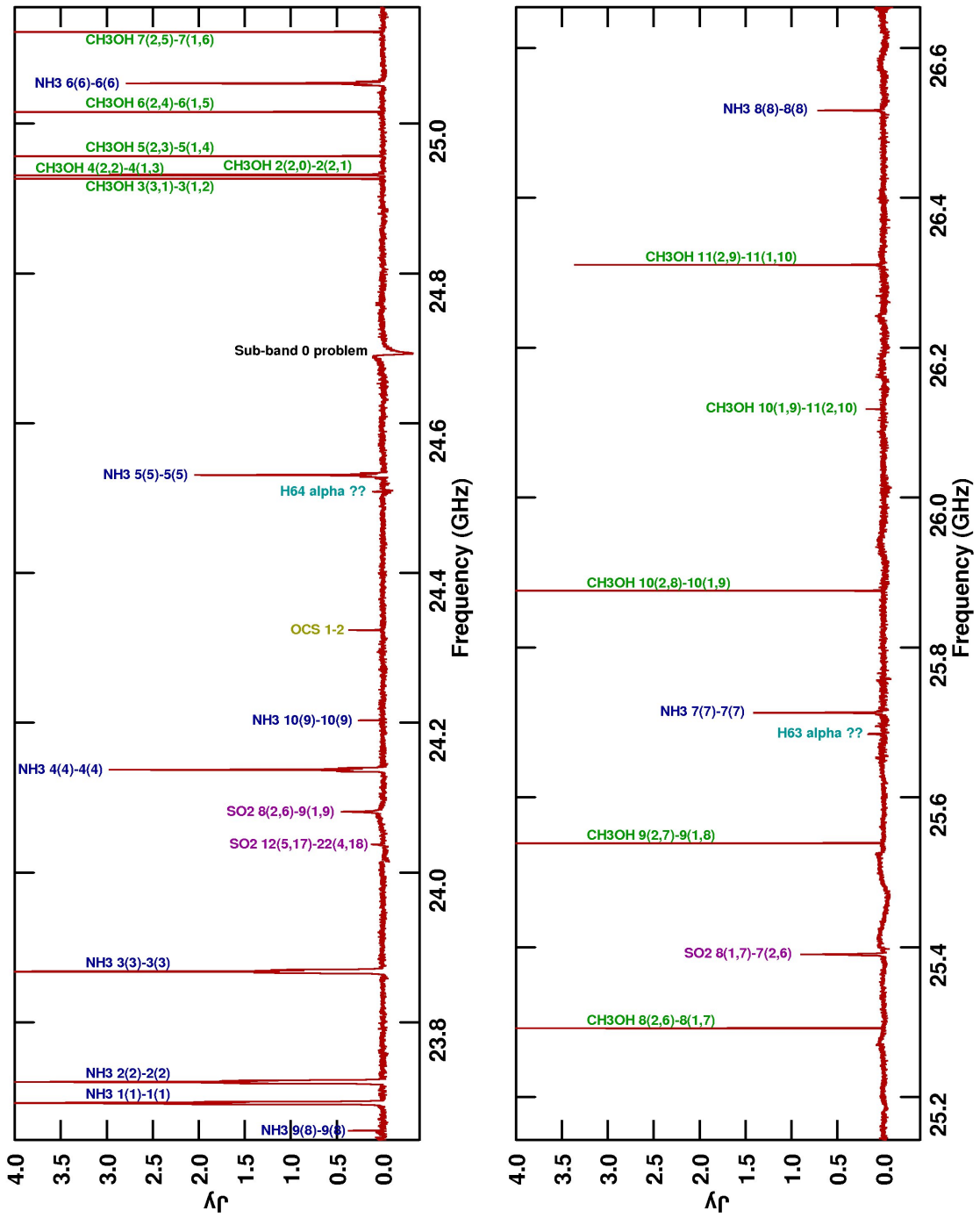


Figure 4.5: **Early EVLA science:** The spectrum of the hot core of Orion A at K band. Three separate observations of 8192 channels each 0.125 MHz wide were made using 12 antennas in the D array. Two hours total telescope time went into each of the two lower thirds of the spectrum and 1 hour was used for the highest third. The plot was made using ISPEC over a 54 by 60 arc second area. Line identifications provided by Karl Menten.

5 Making Images from Interferometer Data

This chapter is devoted to the use of *AIPS* to make and improve images from interferometer visibility data. It begins with a brief description of the routes by which such data arrive in *AIPS*. The basics of weighting, gridding, and Fourier transforming the data to make the so-called "dirty" image are described, followed by a discussion of deconvolution, particularly Clean. The output of Clean is a model of the sky which, in cases of good signal-to-noise, can be fed back to improve the calibration of the interferometer data, a process called "self-calibration." How this is done in *AIPS* is described. This entire process often isolates bad data samples, not previously removed from the data set. An interactive, baseline-based data editor called EDITR is described at the end of the chapter. You may find it more useful than TVFLG (§ O.1.6) for removing data at this stage in the processing. Task IMAGR now does "3-dimensional" imaging, SCMAP contains an editing option at each self-calibration cycle, and EDITR has replaced IBLED as the baseline-based editor of choice. A *UV*-plane based interactive editor called UFLAG is now available.

Lists of *AIPS* software appropriate to this chapter can be obtained at your terminal by typing ABOUT UV ↵, ABOUT CALIBRATION ↵, ABOUT EDITING ↵, and ABOUT IMAGING ↵. Relatively recent versions of these lists are also given in Chapter 13 below. Basic data calibration is discussed in Chapter 4, editing is discussed in § 4.3.5 and § 8.1, and imaging and self-calibration are also discussed in § 8.4 for spectral-line data and in § 9.7 for VLBI data.

5.1 Preparing *uv* data for imaging

AIPS requires visibility data to be calibrated before imaging. If your data are not yet calibrated, return now to Chapter 4, read in your data, and carry out the steps necessary to determine calibration corrections for your data. Note that the main imaging task, IMAGR, does not require you to run SPLIT to apply the calibration in advance. IMAGR can do that for you. Nonetheless, for simplicity and speed — if you are running IMAGR multiple times — it may be best to SPLIT and perhaps even UVSRT the data in advance of running IMAGR. When used for self-calibration, tasks CALIB and SCMAP normally work on data that have been SPLIT in advance.

If your calibrated data are not already on disk in *AIPS* cataloged files, then you will need to import them. These data will normally arrive in *AIPS* from FITS format disk files, although FITS format tapes may also be used. FITS is the internationally recognized standard for moving astronomical data between different types of computers and different software packages. Pre-1990 VLA data may also be stored on EXPORT format tapes. This format was written by the now-deceased VLA DEC-10 and as an option by old versions of *AIPS*.

5.1.1 Indexing the data — PRTTP

Copy your data file(s) to a disk visible to your computer and set an environment variable, *e.g.*, MYAREA, to point at the disk directory containing the files; see § 3.10.1 and § 3.10.3. Then start AIPS, and enter

> DATAIN = 'MYAREA:*filename* ↵ to point at the data file named *filename*.

> TPHEAD ↵ to see a summary of the file header to make sure it is the file you want.

Should you still have magnetic tape, bring your data tape to the *AIPS* processor and follow the tape mounting instructions in § 3.9. The program PRTTP reads a full tape and prints out a summary of all the *uv* and image data on tapes written in any of the supported formats. It is helpful to route the output to the line printer by setting DOCRT=-1 ; OUTPRINT = ''. The file at which the tape is currently positioned can also be "indexed" by the AIPS verb TPHEAD.

5.1.2 Loading the data — FITLD and UVLOD

FITLD copies FITS-format images and *uv* data from tape (or from an external FITS-format disk file) into your *AIPS* catalog on disk. The following shows inputs to FITLD for reading data from a FITS-disk file:

> TASK 'FITLD' ; INP ↵	to set the task name and review the required inputs.
> DATAIN 'FITS:*filename* ↵	to read the FITS-disk file in the public area known as $FITS of name *filename*.
> CLRONAME ↵	to use the file names in the file.
> OUTDISK 3 ↵	to specify writing to disk 3, *e.g.*, to select a disk with sufficient free space. (See § 3.6 for help in monitoring free disk space).
> DOUVC 1 ↵	to use compressed *uv* disk format to save space.
> GO ↵	to run FITLD..

Multiple FITS-disk files may be read in one run of FITLD; set NFILES and name the files with sequential post-pended numbers beginning with 1 (*e.g.*, FITS-file_1, FITS-file_2, . . ., FITS-file_*n*). See § 3.10.3 for a discussion of FITS disk files.

If you have the data on tape, mount the tape and use the commands described in § 3.9.4 to position the tape to the desired data file. Check the positioning with TPHEAD, set adverb NFILES=0 to leave the tape position fixed, set NFILES and OPTYPE if you wish to load more than one file, and then issue the GO to FITLD. If your data are in the old EXPORT format, you must use UVLOD instead. This task is restricted to *uv* files, but can read both FITS and EXPORT formats. Since the latter may have multiple sources, frequencies, and the like in each file, UVLOD has extra adverbs to let you specify source name, frequency band, source qualifier number, and, if all others fail, position within the file. See HELP UVLOD ↵ for details.

Once FITLD has finished, check that your disk catalog now contains the *uv* data you have just tried to load by:

> INDI OUTDISK ; UCAT ↵

which will list all *uv* data sets in your disk catalog. This list should look something like:

```
CATALOG ON DISK  3
CAT USID MAPNAME      CLASS   SEQ  PT     LAST ACCESS      STAT
  1   76 3C138 A C   .UVDATA .   1 UV 22-MAR-1995 12:33:34
```

Alternatively, get terminal *and* hard-copy listing of your catalog by:

> CLRNAME ; INTY 'UV' ↵	to list all disks, *uv* files only.
> CATALOG ↵	to put the catalog listing in the message file.
> PRTMSG ↵	to print the message file.

This sequence takes a little longer to execute, but the hard-copy list (sent to the appropriate printer) may be useful if your catalog is a long one. Note that the catalog has assigned an ordinal number to the data set (76) in the first (CAT) column of the listing. This number and the disk number (3) should be noted for future reference as they are useful when selecting this data set for further processing. Don't forget that the 2nd, 3rd, 4th, and 5th input and the output data name sets also have corresponding MCAT, UCAT, and CATALOG verbs. See § 3.3 and § 3.3.1.

5.1.3 Sorting the data — UVSRT

Some of the *AIPS* imaging tasks, such as UVMAP, require the *uv* data to be in "XY" sort order (decreasing $|u|$). The recommended IMAGR is able to sort the data for you and will do so only if it has to. If you are planning to run IMAGR a number of times, you can help things along by sorting the data in advance. Note, however, that self-calibration requires data in TB (time-baseline) order. Thus, if you are planning to use self-calibration, you should probably sort the data to — or leave them in — TB order. To sort a data set:

> TASK 'UVSRT' ; INP ↵	to set the task name and list the input parameters.
> INDI *n* ; GETN *ctn* ↵	to select the input file, where *n* is the disk number with the *uv* data and *ctn* is its catalog number on that disk. (*n* = 3 and *ctn* = 1 from our UCAT example).
> OUTN INNA ; OUTCL 'UVSRT' ↵	to set the output file name to the same as the input file name and the output file class to UVSRT; these are actually the defaults.
> SORT 'XY' ↵	to select the "XY" sort type required for image making.
> INP ↵	to review the inputs you have selected. *N.B.*, check them carefully since the sort can be time consuming for large data sets.
> GO ↵	to run the task UVSRT.

The task MSORT may be faster for data sets with large numbers of spectral channels and for data sets that are nearly in the desired order. Task OOSRT is yet another option.

Once UVSRT has finished, check that a *uv* database with the "class" .UVSRT has appeared in your disk catalog by:

> INDI 0 ; UCAT ↵

The catalog listing might now look like:

```
CATALOG ON DISK  3
CAT USID MAPNAME      CLASS    SEQ  PT     LAST ACCESS      STAT
  1   76 3C138 A C   .UVDATA .    1 UV 22-MAR-1995 12:33:34
  2   76 3C138 A C   .UVSRT  .    1 UV 22-MAR-1995 12:56:50
```

Note that the catalog number of the sorted file need not be contiguous with that of the unsorted file. All *AIPS* installations now have "private" catalog files containing only your data. Your *uv* files will have contiguous catalog numbers starting from 1 when you first write *uv* data to disk. See also § 3.3.3.

Deep integrations often involve multi-day observations of the same source position in the same antenna configuration. After calibration (and usually at least one round of self-cal), such data may be combined and compressed by the RUN file STUFFR. This compiles a procedure that will convert times to hour angles (TI2HA), sort, and concatenate the data from all days, and then do a baseline-length dependent time averaging (UBAVG). This produces a data set which is more manageable in size and which can still be self-calibrated at some level, although the days are now fully merged. Task HA2TI reverses the process, although the separation of merged days will be lost.

5.1.4 Combining data sets for imaging — DBCON and VBGLU

Multiple data sets may be combined — with some serious restrictions — to make a single image. DBCON only works properly if the two data sets are rather similar. Multi-source files must have been observed with the same antenna locations and names; DBCON then produces a multi-source output file with one subarray. For single-source files (after SPLIT) however, DBCON can create an output with multiple "subarrays" represented by multiple AN (antenna) tables. Thus data from multiple array configurations having the same polarization,

spectral, and IF structure can be combined and imaged together. DBAPP does a similar operation for data sets that match more closely. DBCON allows you to scale the weights of the two data sets to control how they contribute to the final image. Task AVGWT will provide data on the average and rms of weights in selected portions of a data set.

IMAGR can loop over subarray, allowing the desired imaging. However, IMAGR cannot loop over frequency ID numbers. To combine data sets at different frequencies one must use VBGLU. This task was initially written to combine multiple passes of the correlator over the same data set since, now obsolete correlators were limited in the number of IFs in a single pass through the input data. Because of the foibles of those correlators, VBGLU can combine data samples from one data set that at times do not occur in the others. It simply flags the data from the IFs that were not observed at that time. This capability was extended in 31DEC21 to allow for data sets from different days. So long as the polarizations of the data sets and the number of spectral channels match, up to 4 data sets may be combined from disparate observing sessions in one run of VBGLU. This does make for a large data file but enables imaging, e.g. data from both ends of C band together, or even S and C band, if the spectral index is close to zero.

5.2 Basic image making — IMAGR

AIPS had several imaging tasks, each with distinctive capabilities, BUT they have all been superseded by IMAGR. The abilities of IMAGR include:

1. data calibration application for multi-source or self-calibrated single-source data sets.
2. data sorting if needed to fit the weighting, gridding, or Cleaning.
3. time averaging of data in a baseline-dependent fashion based on field of view.
4. data weighting options far more general than those in any other task and including all those used in previous tasks.
5. data imaging in up to 4096 simultaneous fields, each up to 16384x16384 in size.
6. Cleaning of all fields simultaneously with subtraction of the Clean components from the data at each major cycle followed by re-computation of the residual images — avoiding aliasing of sidelobes and allowing components almost to the edges of each field.
7. re-projection of the (u, v, w) baseline coordinates to make each field tangent to the Celestial sphere at its center thereby making a larger area of each field free of projection defects.
8. correction of Clean components for various wide-field and wide-bandwidth effects.
9. truly interactive TV display of residual images allowing you to alter the areas over which Clean components are sought.
10. sensible Cleaning strategies for, and restoration to, overlapped image fields.
11. choice of Clark or Steer-Dewdney-Ito methods of component selection.
12. filtering of weak, isolated Clean components to reduce the Clean bias.
13. simultaneous Cleaning with multiple component widths.
14. automatic selection of the areas to be Cleaned based on image statistics and peak values.

This section will concentrate on how to use IMAGR to weight, grid, and Fourier transform the visibility data, making a "dirty beam" and a "dirty map." We will begin with a simple example and then discuss a number of matters of image-making strategy to help make better images. Deconvolution will be discussed in the next section. This separation reflects our belief that you should first use IMAGR to explore your data to make sure that there are no gross surprises — emission from unexpected locations, "stripes" from bad calibration or interference, and the like. If you begin Cleaning immediately, you may find that you are using Clean to convert noise and sidelobes into sources while failing to image the real sources, if any. It is a good idea to make the first images of your field at the lowest resolution (heaviest taper) justified by your data. This will allow you to choose input parameters to combine imaging and Cleaning steps optimally.

We do not discuss imaging theory and strategy in much detail here because it is discussed fully in numerous lectures in *Synthesis Imaging in Radio Astronomy*[1].

5.2.1 Making a simple image

The most basic use of IMAGR is to make an image of a single field from either a single-source data set or, applying the calibration, from a multi-source data set. Do not be discouraged by the length of the INPUTS list for IMAGR. They boil down to separate sets for calibration (with which you are familiar from Chapter 4), for basic imaging, for multi-field imaging, and for Cleaning. We will consider the second set here, the third in the next sub-section, and the last in § 5.3. A standard procedure MAPPR provides a simplified access to IMAGR when calibration, polarization, multiple fields, and other more complicated options are not needed.

A typical use of IMAGR at this stage is to construct an unpolarized (Stokes I) image at low resolution and wide field to search for regions of emission or at full resolution for deconvolution by image-plane techniques discussed in § 5.3.7. The following example assumes the use of an already calibrated, single-source data set:

> DEFAULT IMAGR ; TASK 'MAPPR' C$_R$	to set the "task" name and set all IMAGR's adverbs to initial values.
> INP C$_R$	to see what parameters should be set.
> INDI *m*; GETN *n* C$_R$	to select the desired *uv* database.
> IMSIZE 1024 C$_R$	to make a square image 1024 pixels on each side.
> CELLSIZ 1 C$_R$	for 1 arc-second cells.
> UVTAP *utap vtap* C$_R$	to specify the widths to 30% of the Gaussian taper in u and v in kλ (kilo-wavelengths).

Other inputs are defaulted sensibly, which is why we started with a DEFAULT and are using the MAPPR procedure. In particular, Clean is turned off with NITER = 0, other calibrations are turned off, and all of the data (all IFs, channels, sub-arrays) will be used. Data weighting will be somewhere between pure "uniform" and pure "natural" (see § 5.2.3). Note that task SETFC can be requested to examine your data file and make recommendations on the best combination of CELLSIZE and IMSIZE. Consider also both:

> DOCRT = -1 ; EXPLAIN IMAGR C$_R$ to print the long explain file, and

> HELP *xxx* C$_R$

where *xxx* is a parameter name, *e.g.*, IMSIZE, UVWTFN, etc., to get useful information on the specific parameter. The default *uv* convolution function is a spheroidal function (XTYPE, YTYPE = 5) that suppresses aliasing well. Check that you are satisfied with the inputs by:

> INP C$_R$

then:

> MAPPR C$_R$ to run IMAGR.

[1] *Synthesis Imaging in Radio Astronomy*, A Collection of Lectures from the Third NRAO Synthesis Imaging Summer School, eds. R. A. Perley, F. R. Schwab and A. H. Bridle, Astronomical Society of the Pacific Conference Series Volume 6 (1989)

In IMAGR, you may limit the data used to an annulus in the *uv* plane with UVRANGE, given in kilo-wavelengths. This is a useful option in some cases, but, since it introduces a sharp edge into the data sampling and otherwise discards data that could be improving the signal-to-noise, it should be used with caution and is not available in MAPPR. Taper and other data weighting options may accomplish much the same things, but do not introduce sharp edges and do not entirely discard the data.

In the example above, we chose to make the image and each cell square. This is not required. Images can be any power of two from 64 to 16384, *e.g.*, 2048 by 512 or 128 by 8192, if you want, and the cells may also be rectangular in arc-seconds. There may be good reasons for such choices, such as to avoid imaging blank sky (saving disk, time ...) and to make the synthesized beam be roughly round when measured in pixels. Rectangularity may complicate rotating the image later with *e.g.*, LGEOM, but the problem can be handled with the more complex HGEOM. IMAGR has the ROTATE adverb to allow you to rotate your image with respect to the usual right ascension and declination axes to align elongated source structure with the larger axis of your image.

IMAGR will create both "dirty" beam and map images. The *AIPS* monitor provides some important messages while IMAGR is running. When you see IMAGR*n*: APPEARS TO END SUCCESSFULLY on this monitor, you should find the requested images in your catalog using:

> INDI 0 ; MCAT ↵

This would produce a listing such as:

```
CATALOG ON DISK  2
CAT USID MAPNAME       CLASS   SEQ  PT      LAST ACCESS      STAT
 42   76 3C138 A C    .IIM001.    1 MA 22-MAR-1999 13:50:10
 43   76 3C138 A C    .IBM001.    1 MA 22-MAR-1999 13:59:58
```

Note that the default beam class is IBM001; the default image class will be IIM001. The images produced with NITER = 0 by IMAGR can be deconvolved by various image-plane methods (§ 5.3.7).

5.2.2 Imaging multiple fields and image coordinates

There is little real need for the multi-field capability of IMAGR unless you are Cleaning. In that case, the ability to remove components found in each field from the *uv* data and, thereby, to remove their sidelobes from every field, is practically a necessity. Nonetheless, it may be more efficient to make multiple fields in one GO and a good idea to check the field size and shift parameters while looking for emission sources before investing significant resources in a lengthy Clean. Task SETFC can recommend cell size, image size, and field locations to cover the central portion of the single-dish beam. In 31DEC25, adverb DOINVERS may be used to obtain an alternate set of field locations. Tasks CHKFC and FLATN may then be used to choose whichever is better for your imaging.

You specify the multiple-field information with:

> NFIELD n ↵	to make images of n fields.
> IMSIZE i, j ↵	to set the *minimum* image size in x and y to i and j, where i and j must be integer powers of two from 64 to 8192.
> FLDSIZ $i_1, j_1, i_2, j_2, i_3, j_3, \ldots$ ↵	to set the area of interest in x and y for each field in turn. Each i_n and j_n is rounded up to the greater of the next power of 2 and the corresponding IMSIZE. FLDSIZE controls the actual size of each image and sets an initial guess for the area over which Clean searches for components. (That area is then modified by the various box options discussed later.)
> RASHIFT $x_1, x_2, x_3, \ldots$ ↵	to specify the x shift of each field center from the tangent point; $x_n > 0$ shifts the field center to the East (left).

> DECSHIFT $y_1, y_2, y_3, \ldots$ ↵

to specify the y shift of each field center from the tangent point; $y_n > 0$ shifts the field center to the North (up).

> DO3DIMAG TRUE ↵

to specify that the (u, v, w) coordinates are re-projected to the center of each field. DO3DIMAG FALSE also re-projects (u, v, w) to correct for the sky curvature while keeping all fields on the same tangent plane. The two choices now produce very similar results.

If ROTATE is not zero, the shifts are actually with respect to the rotated coordinates, not right ascension and declination. There may be good reasons to have the fields overlap, but this can cause some problems which will be discussed in § 5.3. IMAGR has an optional BOXFILE text file which may be used to specify some or all of the FLDSIZE, RASHIFT, and DECSHIFT values. To simplify the coordinate computations, the shift parameters may also be given as right ascension and declination of the field center, leaving IMAGR to compute the correct shifts, including any rotation. BOXFILE may also be used to specify initial Clean boxes for some or all fields, values for BCOMP, spectral-channel-dependent weights, and even a list of facets to ignore totally.

The OUTCLASS of the fields is controlled by IMAGR with no user assistance. For dirty images it is IIM001 for the first field, IIM002 for the second, and so forth. The I is replaced by Q, U, *et al.* for polarized images and the IM is replaced by CL when Cleaning. When ONEBEAM is true, one beam of class IBM001 is used for all fields. When ONEBEAM is false, a beam of class IBM*nnn* is used for fields of class IIM*nnn* or ICL*nnn*, with similar substitutions for other Stokes parameters.

Users are often confused by the fact that radio synthesis images are made in a rectangular coordinate system of direction cosines that represents a *projection* of angular coordinates onto a tangent plane. Over wide fields of view, the image coordinates are not simple scalings of right ascension and declination. For details of all coordinate systems supported by *AIPS*, please consult *AIPS* Memos No. 27, "Nonlinear Coordinate Systems in *AIPS*," and No. 46, "Additional Non-linear Coordinates," by E. W. Greisen (available via the World-Wide Web § 3.8 and § 3.10.3).[2] The coordinate system for VLA images is the SIN projection, for which the image coordinates x and y relate to right ascension α and declination δ as

$$\begin{aligned} x &= \cos\delta \sin\Delta\alpha \\ y &= \cos\delta_0 \sin\delta - \sin\delta_0 cos\delta \cos\Delta\alpha \end{aligned}$$

where $\Delta\alpha = \alpha - \alpha_0$ and the coordinates with subscript "0" are those of the tangent point that serves as the origin of the image coordinate system. When DO3DIMAG is false, all fields have a single coordinate origin, but, when DO3DIMAG is true, each field has a different coordinate origin (at its center). RASHIFT and DECSHIFT are now "simple" shifts, specified with respect to the reference coordinate of the input *uv* data set, rather than SIN projection shifts. Thus

$$\begin{aligned} \texttt{RASHIFT} &= (\alpha - \alpha_0)\ \cos\delta_0 \\ \texttt{DECSHIFT} &= \delta - \delta_0 \end{aligned}$$

For many practical purposes, it is sufficiently accurate to suppose that imaging parameters do correspond to simple angular shifts of the image on the sky. *AIPS input terminology* reflects this simplification, although *actual coordinate shifts and transformations* in all *AIPS* tasks and verbs *are accomplished rigorously using the full non-linear expressions*. If you want to relate shifts in pixels (image cells) to shifts in sky coordinates (α, δ) *manually*, you must understand, and take account of, the non-linear coordinate system yourself. The verb IMVAL can help by displaying the non-linear coordinates for the specified input pixel. This is rarely necessary, however.

[2] See also *AIPS* Memo No. 113, "Faceted Imaging in *AIPS*" by Kogan and Greisen for details of the DO3DIMAG = false geometry now used.

5.2.3 Data weighting

The minimum noise in an image is produced by weighting each sample by the inverse square of its uncertainty (thermal noise). AIPS assumes that the input weights are of this form, namely $W \propto 1/\sigma^2$. FILLM offers the option, for VLA data, of weighting data in this fashion using recorded system temperatures (actually "nominal sensitivities"). EVLA data may be weighted using SysPower data with TYAPL or simply with the variances found in the data with REWAY. Channel-dependent weights may also be determined by BPWAY which finds a normalized weighting "bandpass." The resulting weights are reasonably accurate for this use *after calibration*. In some cases, weights are simply based on integration times and the assumption that each antenna in the array had the same system temperature. If the weights are not of the correct form, run REWAY or, if necessary,FIXWT on the *uv* data set to calculate weights based on the variances in the data themselves. Then to get the minimum-noise image, specify

> UVWTFN 'NA' ⏎ to get "natural" weighting.

to have all samples simply weighted by their input weights. Unfortunately, most interferometers do not sample the *uv* plane at all uniformly. Typically, they produce large numbers of samples at short spacings with clumps of samples and of holes at longer spacings. Thus, the beam pattern produced by natural weighting tends to have a central beam resembling a core-halo source with the broad halo (or plateau) produced by all the short spacing data and also to have rather large sidelobes due to the clumps and holes. In some VLBI arrays, data from some baselines have weights much much greater than from other baselines due to differences in antenna size and receiver temperature. Only the high-weight baselines contribute significantly to a natural-weighted image in this case.

To reduce the effects of non-uniformity in data sampling, the concept of "uniform" weighting was devised. In its purest form, uniform weighting attempts to give each cell in the *uv*-plane grid the same weight. Thus, the weight given each sample, is its weight divided by the sum of weights of all samples in the cell in which it occurs. In this case, in some cells a sample will count at full weight while in another, possibly adjacent, cell a sample will count at only a small fraction of its weight. To obtain this classic weighting in IMAGR enter:

> UVSIZE 0 ; UVWTFN ' ' ⏎ to specify a weight grid the size of the image grid and the default weighting scheme.

> ROBUST = -7 ⏎ to turn off all weight tempering.

IMAGR actually implements a far more flexible (and therefore more complicated) scheme to give you a wide range of weighting choices. The intent of uniform weighting is to weight a sample inversely with respect to the local density of data weights in a wider sense than the default cell boundaries. IMAGR allows you to choose the size of cells in the *uv* plane with UVSIZE, the radius in units of these cells over which each sample is counted with UVBOX, and the way in which each sample is counted over this radius with UVBXFN. The weighting grid can be smaller or larger than the image grid. You can even make the *uv* cells be very small by specifying a very large UVSIZE; you are limited only by the available memory in your computer and the time you wish to spend weighting the data. Note, of course, that uniform and natural weighting are the same if the cells are small enough unless you specify a significant radius over which to count the samples. IMAGR does not stop here, however. It also allows you to alter the weights before they are used, to count samples rather than weights, and to temper the uniform weights with Dan Briggs' "robustness" parameter. Thus

$$W_{out} = \frac{T W_{in}^{p}}{\sum_{(i)} W_{in}^{pq} + R \overline{\sum_{(i)} W_{in}^{pq}}}$$

where W_{in} is the input weight, W_{out} is the weight used in imaging, T is any tapering factor, p is an input weight modification exponent, q separates uniform weights ($q = 1$) and uniform counts ($q = 0$), the sum is actually

$$\sum_{(i)} W_{in}^{pq} \equiv \sum_{j}^{N} W_{in}^{pq}(j)\, \mathrm{fun}(\sqrt{(u_i - u_j)^2 + (v_i - v_j)^2})$$

with fun being some function of the separation between sample i and all samples j, the over-line represents the average over all samples, and

$$R \equiv \frac{10^{\texttt{ROBUST}}}{5}.$$

The exponents are set by UVWTFN as: $q = 1$ except $q = 0$ when the first character of UVWTFN is 'C' and $p = 1$ except $p = 0.5, p = 0.25$ and $p = 0$ when the second character of UVWTFN is 'S', 'V', and 'O' (the letter), respectively.

At this point you should be totally confused. To some extent, we are. While IMAGR has been around for a while, the impact of all of these parameters on imaging is not well understood. You may wish to experiment since it *is* known — see figures on next page — that weighting can make a significant difference in the signal-to-noise on images, can alter the synthesized beam width and sidelobe pattern, and can produce bad striping in the data when mildly wrong samples get substantially large weights. The default values do seem to produce desirable results, fortunately. The beam width is nearly as narrow as that of pure uniform weighting, but the near-in sidelobes are neither the positive "shelf" of pure natural weighting nor the deep negative sidelobes of pure uniform weighting. The expected noise in the image is usually rather better than for pure uniform weighting and sometimes approaches that of natural weighting. Deconvolution should be improved with reduction of erroneous stripes, noise, and sidelobe levels. You should explore a range of UVTAPER and ROBUST (at least) in a systematic way in order to make an informed choice of parameters.

The option to average the input data in time over a baseline-length dependent interval adds further complications. It has the desirable effect of reducing the size of your work file, thereby speeding the imaging process. But it changes the distribution of data samples ahead of uniform weighting since it averages the short-spacing samples a great deal more than the long-spacing samples. This makes natural weighting less undesirable and causes uniform weighting to be less sensitive to extended source structures. Set IM2PARM(11) and (12) to request this option. IM2PARM(13) adds frequency averaging to the time averaging option. It may be desirable to keep narrow spectral channels in the data set for editing purposes, but the bandwidth synthesis may be speeded up significant while causing no degradation by a judicious choice of channel averaging.

If your source has complicated fine structure and has been observed with the VLA at declinations south of about +50°, there may be important visibility structure in the outer regions of the *uv* plane that is sampled sparsely, even by "full synthesis" imaging. In such cases, Clean may give images of higher dynamic range if you are not too greedy for resolution at the imaging stage. Use UVTAPER to down-weight the poorly sampled outer segments of the *uv* plane in such cases. (UVRANGE could be used to exclude these data, but that introduces a sharp discontinuity in the data sampling with a consequent increase in sidelobe levels.) Tapering is, to some extent, a smooth inverse of uniform weighting; it down-weights longer spacings while uniform weighting down-weights shorter spacings in most arrays. The combination can produce an approximation to natural weighting that is smooth spatially.

IMAGR does all weighting, including tapering, in one place and reports the loss in signal-to-noise ratio from natural weighting due to the combination of weighting functions. This reported number does *not* include the loss due to discarding data via UVRANGE, GUARD, the finite size of the *uv*-plane grid, data editing, and the like.

5.2.4 Cell and image size, shifting

Other things being equal, the accuracy of beam deconvolution algorithms (§ 5.3) generally improves when the shape of the dirty beam is well sampled. When imaging complicated fields, it may be necessary to compromise between cell size and field of view, however. If you are going to Clean an image, you should set your imaging parameters so that there will be at least three or four cells across the main lobe of the dirty beam.

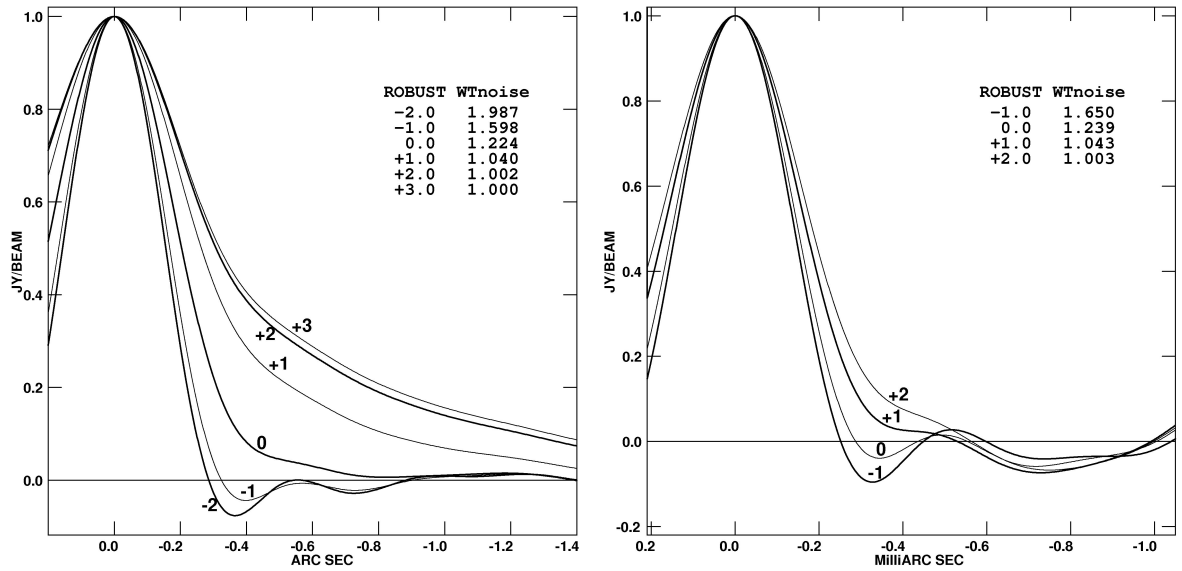


Figure 5.1: Slices taken through the centers of synthesized beams for various values of the ROBUST parameter. Plot at left for a VLA A- and B-array data set, while the plot at right is for a VLBA data set. Do not assume that these plots apply to your data sets, however. Tables give noise increase over natural weighting (ROBUST large).

Actually, this is not the full story. If you have a large number of samples toward the outer portions of the *uv*-data grid, then the width of the main lobe of the dirty beam will not be correctly measured. Making the cell size smaller — raising the size of the *uv*-data grid (in wavelengths) — will change the apparent beam width even if no additional data samples are included. Even when you have a cell size small enough to accurately represent the dirty beam, the presence of samples in the outer portion of the *uv*-data grid can confuse high dynamic-range deconvolution. The high-resolution information contained in these outer samples cannot be represented with point sources separated by integer numbers of too-large cells. The result is a sine wave of plus and minus intensities, usually in the x or y direction, radiating away from bright point objects and a Clean that always finds a component of opposite sign at a virtually adjacent pixel whenever a component is taken at the bright point sources. This is often a subtle effect lost in the welter of long Cleans, but has led to the concept of a "guard" band in the *uv*-data grid. The adverb GUARD in IMAGR and friends, controls the portion of the outer *uv*-data grid which is kept empty forcibly by omitting any data that would appear there. The default is the outer 30% of the radius (or less if there is taper), which is a compromise between the 50% that it probably should be and the epsilon that some vocal individuals believe is correct. All imaging tasks will tell you if they omit data because they fall off the grid or outside the guard band and will warn you of possible Cleaning problems if data lie inside the guard band but outside a more conservative guard band.

Because Clean attempts to represent the brightness distribution of your source as an array of δ-functions, the deconvolution will have higher dynamic range if the brightest point-like features in your images have their maxima exactly at pixel locations. In this case, the brightest features can be well represented by δ-functions located at image grid points. If you are pursuing high dynamic range, it may therefore be worth adjusting the image shift and cell-size parameters so that the peaks of the two brightest point-like features in your image lie exactly on pixels.

If you are going to use image-plane deconvolutions such as APCLN, SDCLN, or VTESS, you must image a large enough field that no strong sources whose sidelobes will affect your image have been aliased by the FFT and so that all real emission is contained within the central quarter of the image area. With IMAGR, you should make a small image field around each confusing source (or use Clean boxes within larger fields).

5.2.5 Zero-spacing issues

You help Clean to guess what may have happened in the unsampled "hole" at the center of the *uv* plane by including a zero-spacing (usually single-dish) flux density when you make the image. This gives Clean a datum to "aim at" in the center of the *uv* plane. Extended structure can often be reconstructed by deep Cleaning when the zero-spacing flux density is between 100% and ∼125% of the average visibility amplitude at the shortest spacings (run UVPLT to estimate this average for your data set). If your data do not meet this criterion, there may be no reliable way for you to image the extended structure of your source without adding further information to your observations (*e.g.*, by adding *uv* data from a more compact array, by Fourier transforming a suitably tapered and deconvolved single dish image of the VLA primary beam, or by using such an image as the default image for a maximum entropy deconvolution as in § 5.3.7). See § 10.5 for further discussion. IMAGR treats the zero spacing differently from previous tasks. The adverb ZEROSP gives five values, the I, Q, U, V fluxes, and a weight. This weight should be in the same units as for your other data, since the ZEROSP sample is simply appended to your data set and re-weighted and gridded just like any other data sample. To have the zero spacing be used, both ZEROSP(1) and ZEROSP(5) must be greater than zero, even when you are imaging some other polarization. Previous "wisdom" held that the weight should be "the number of cells that are empty in the center of the *uv* plane," but this does not appear to be correct with IMAGR.

If UVPLT shows a rapid increase in visibility amplitudes on the few shortest baselines in your data, but not to a value near the integrated flux density in your field, you may get better images of the *fine* structure in your source by excluding these short baselines with the UVRANGE parameter. There is no way to reconstruct the large-scale structure of your source if you did not observe it, and the few remnants of that structure in your data set may just confuse the deconvolution. Be aware that, in this circumstance, you cannot require your image of total intensity to be everywhere positive. The fine-scale structure can consist of both positive and negative variations on the underlying large-scale structure.

5.3 Deconvolving images

The most widely used deconvolution method is Clean, originally described by Högbom. All *AIPS* Clean tasks implement a Clean deconvolution of the type devised for array processors by Barry Clark (*Astron. & Astrophys.* **89**, 377 (1980)). (Your computer does not need not to have an array processor or other special vector hardware to run them, however.) The recommended task IMAGR implements Clark's algorithm with enhancements designed by Bill Cotton and Fred Schwab. These enhancements involve going back to the original *uv* data at each "major cycle" to subtract the current Clean-component model and re-make the images. This allows for more accurate subtraction of the components, for Cleaning simultaneously multiple (perhaps widely spaced) smaller images of portions of the field of view, for Cleaning of nearly the full image area, for more accurate removal of sidelobes, and for corrections for various wide-field and wide-bandwidth effects. Of course, all these extras do come at a price. For large data sets with fairly simple imaging requirements, image-based Cleans, particularly APCLN, may be significantly faster.

The next section describes the basic parameters of Cleaning with IMAGR. The second section describes the use and limitations of multiple fields in IMAGR; the third section describes the setting of Clean "boxes" and the TV option in IMAGR; the fourth section describes some experimental extensions to standard Clean; and the fifth section describes various wide-field and wide-bandwidth correction options. Clean component files are tables which can be manipulated, edited, and plotted both by general-purpose table tasks and by tasks designed especially for CC files. Some aspects of this are discussed in the fifth section. Images may also be deconvolved by other methods in *AIPS*. § 5.3.7 mentions several of these and describes the most popular alternatives, image-based Clean with APCLN and SDCLN and a Maximum Entropy method embodied in the task VTESS.

5.3.1 Basic Cleaning with IMAGR

IMAGR implements a Clean deconvolution of the type devised by Barry Clark and enhanced by Bill Cotton and Fred Schwab. Clean components — point sources at the centers of cells — are found during "minor" iteration cycles by Cleaning the brightest parts of the residual image with a "beam patch" of limited size. More precise Cleaning is achieved at the ends of "major" iteration cycles when the Fourier transform of the Clean components is computed, subtracted from the visibility data, and a new residual dirty image computed. The rule for deciding when a major iteration should end in order to achieve a desired accuracy is complicated (see the Clark paper). IMAGR lets you vary the major iteration rule somewhat to suit the requirements of your image. Type DOCRT FALSE; EXPLAIN IMAGR ↵, if you haven't already, to print out advice on imaging and Cleaning.

IMAGR both makes and Cleans images. See § 5.2 for the inputs needed to make the images. The inputs for basic Cleaning are:

> OUTS 0 ↵	to create a new output file. If OUTSEQ $\neq 0$, the specified value is used. OUTSEQ must be set to restart a Clean (see below).
> GAIN 0.1 ↵	to set the loop gain parameter, defaults to 0.1. Values of 0.2 or more may be suitable for simple, point-like sources, while even smaller values may be required for complex sources with smooth structure.
> FLUX f ↵	to stop Cleaning when the peak of the residual image falls to f Jy/beam.
> NITER n ↵	to stop Cleaning when n components have been subtracted. There is no default; zero means no Cleaning.
> BCOMP 0 ↵	to begin a new Clean — see below for restarting one.
> NBOXES 0 ; BOXFIL ' ' ↵	to specify no Clean search areas in advance; see § 5.3.3.
> CMETHOD ' ' ↵	to allow IMAGR to use DFT or gridded-FFT component subtraction at each major cycle, depending on which is faster. 'DFT' forces DFT and 'GRID' forces gridded subtraction at all iterations. Use the default. DFT is more accurate, but usually much slower; see the explain file for details.
> FACTOR 0 ↵	to use the "normal" criteria for deciding when to do a major cycle; see below.
> BMAJ 0 ↵	to have IMAGR use a Clean beam which is a fit to the central lobe of the dirty beam.
> DOTV 1 ↵	to have dirty and residual images displayed on the TV; see § 5.3.3.
> LTYPE 3 ↵	to have axis labels plotted on the TV along with each image; see § 5.3.3. LTYPE ≤ 2 causes the labeling to be omitted.
> INP ↵	to review the inputs — read carefully.
> GO ↵	to start IMAGR.

The procedure MAPPR may be used for single-field Cleaning.

The FACTOR parameter in IMAGR can be used to speed up or to slow down the Cleaning process by increasing or decreasing the number of minor cycles in the major cycles. The default FACTOR 0 causes major cycles to be ended using Barry Clark's original criterion. Setting FACTOR in the range 0 to $+1.0$ will speed up the Clean, by up to 20% for FACTOR 1.0, at the risk of poorer representation of extended structure. Setting FACTOR in the range 0 to -1.0 will slow it down, but gives better representation of extended structure.

Two other subtle parameters which help to control the Clean may need to be changed from their defaults.

MINPATCH controls the minimum radius in the dirty beam (in pixels) used during the minor cycles to subtract sidelobes of one component from other nearby pixels. If your dirty beam is complicated, with significant near-in sidelobes and your source extended, then the default 51 cells may be too small. IMAGR uses a larger patch during the first few major cycles, but will be reduced eventually to a MINPATCH patch. IMAGR normally creates a dirty beam twice the size of the largest field (or 4096 pixels whichever is smaller). This allows for a very large beam patch in the early cycles, letting widely spaced bright spots be Cleaned more accurately. If your image does not have widely spaced bright spots, you can save some compute time by reducing this beam size with IMAGRPRM(10); see the help file. MAXPIXEL controls the maximum number of image pixels searched for components during any major cycle. If MAXPIXEL were very large, IMAGR would spend all of its time examining and subtracting from pixels it is never going to use for components. If it is too small, however, then pixels that should be used during a major cycle will not be used and major cycles may end up using only a few components before doing another (expensive) component subtraction and re-imaging. Again, we do not know what to recommend in detail. The default (20050) seems good for normal 1024x1024 images, smaller values are better for smaller images of compact objects, and rather larger values may be good for extended objects or large numbers of fields. If the first Clean component of a major cycle is significantly larger than the last component of the previous cycle (and the messages let you tell this), then too few cells are being used.

If you do not specify the parameters of the Clean beam, a Gaussian Clean beam will be fitted to the central portion of the dirty beam. The results may not be desirable since the central portions of many dirty beams are not well represented by a single Gaussian and since the present fitting algorithm is not very elaborate. If you use the default, check that the fitted Clean beam represents the central part of the dirty beam to your satisfaction. Use task PRTIM on the central part of the dirty beam to check the results — another reason to make an un-Cleaned image and beam first. To set the Clean beam parameters:

> BMAJ *bmaj* ↵ — to set the FWHM of the major axis of the restoring beam to *bmaj* arc-sec. BMAJ = 0 specifies that the beam is to be fitted.

> BMIN *bmin* ↵ — to set the FWHM of the minor-axis of the restoring beam to *bmin* arc-sec; used if BMAJ > 0.

> BPA *bpa* ↵ — to set the position angle of beam axis to *bpa* degrees measured counter-clockwise from North (*i.e.*, East from North); used if BMAJ > 0.

Note that specification of the Clean beam makes the spatial resolution and units of the restored components different fromthose of the residual image. IMAGR will smooth and scale the residual image before adding the components to the output so that all portions of the image have the same spatial resolution and brightness units. Use BMAJ < 0 if you want the *residual* image, rather than the Clean one, to be stored in the output file.

Note that the number of Clean iterations, and many of the other Cleaning parameters, may be changed interactively while IMAGR is running by use of the *AIPS* SHOW and TELL utilities. Type SHOW IMAGR ↵ while the task is running to see what parameters can be reset, and their current values. Then reset the parameters as appropriate and TELL IMAGR ↵ to change its parameters as it is running. (The changes are written to a disk file that IMAGR checks at appropriate stages of execution, so they may not be passed on to the program immediately — watch your *AIPS* monitor for an acknowledgment that the changes have been received, perhaps some minutes later if the iteration cycles are long or your machine is heavily loaded. AIPS verb STQUEUE will show all queued TELLs.) Of particular interest is the ability to turn the TV display back on and to extend the Clean by increasing NITER. There are two ways to tell IMAGR that it has done enough Cleaning: by selecting the appropriate menu item in the TV display or by sending a OPTELL = 'QUIT' with TELL. The former can only be done at the end of a major cycle and only if the TV display option is currently selected, while the latter can be done at any time (although it will only be carried out when the current major cycle finishes).

IMAGR makes a *uv* "workfile" which is used in its Clean step to hold the residual fringe visibilities. Its name is controlled with the IN2NAME, IN2CLASS, IN2SEQ, and IN2DISK parameters. If the first three are left

blank and 0, the workfile will be deleted when IMAGR terminates. Even if the workfile already exists, IMAGR assumes that its contents must be initialized from the main *uv* file unless the ALLOKAY adverb is set ≥ 2.. This file is useful if you suspect that there are bad samples in your data. Use LISTR (§ O.1.5) UVFND (§ 6.2.1), PRTUV (§ 6.2.1), UVPLT (§ 6.3.1) or even TVFLG (§ O.1.6) to examine the file. If you find data which you think are corrupt, remove them from the *input uv* data set with UVFLG. These workfiles may eventually use an annoying amount of disk if IN2SEQ is left 0. Be sure to delete old ones with ZAP in this case.

IMAGR may be restarted to continue a Clean begun in a previous execution. To do this, you must set the OUTSEQ to the sequence number of file you are restarting. A good way to do this is

> OUTDISK d ; GETONAME ctn ↵ — to set the output name parameters to the name parameters of catalog entry ctn on disk d.

The other parameters that must be set to restart a Clean are OUTVER, the output Clean Components version number, and BCOMP, the number of Clean components to take from the previous Cleans. A restart saves you much of the time it took IMAGR to do the previous Clean, although it will make new beam images and a new file of residual visibilities unless you specify that it should not using ALLOKAY. An image can be re-convolved by setting NITER = the sum of the BCOMPs and specifying the desired (new) Clean beam. Images can be switched between residual and Clean (restored) form in the same way, setting BMAJ = -1 to get a residual image. (For single fields, tasks RSTOR and CCRES may be used for this purpose.) IMAGR writes over the Clean image file(s) as it proceeds to Clean deeper. You can preserve intermediate Clean images, however, either by copying them to another disk file with SUBIM or by writing them to tape with FITAB or FITTP.

5.3.2 Multiple fields in IMAGR

IMAGR can also deconvolve components from up to 4096 fields of view simultaneously, taking correct account of the w term at each field center (DO3DIMAG false) or even re-projecting the (u, v, w) coordinates as well as the phases to each field center (DO3DIMAG true). This is a vital advantage if there are many localized bright emission regions throughout your primary beam; only the regions containing significant emission need to be imaged and cleaned, rather than the entire (mainly empty) area of sky encompassing them all. It may even be necessary to image regions well outside the primary beam, not because you will believe the resulting images, but to remove the sidelobes of sources in those distant fields from the primary fields. To take advantage of this option, you must have prior knowledge of the location and size of the regions of emission that are important — yet another reason to make a low resolution image of your data first. Task SETFC helps you prepare multi-field input to IMAGR using the NRAO VLA Sky Survey (NVSS) source catalog and even the current coordinate of the Sun. It can also recommend cell and image sizes. After Cleaning, multiple fields (and even multiple pointings of a mosaic) from IMAGR may be put into a single large image on a single geometry by FLATN. Task CHKFC may be used with FLATN to check that a given BOXFILE covers the desired portion of sky with fields and Clean boxes. The BOXFILE may be edited by task BOXES to put Clean boxes around sources from a source list such as the NVSS or WENSS. The task FIXBX may be used to convert the Clean boxes from the facets and cell sizes of one box file to those of another.

The use of IMAGR to make images of multiple fields was described in § 5.2.2. To repeat some of the description, you specify the multiple-field information with:

> NFIELD n ↵ — to make images of n fields.

> IMSIZE i, j ↵ — to set the minimum image size in x and y to i and j, where i and j must be integer powers of two up to 8192.

> RASHIFT $x_1, x_2, x_3, \ldots$ ↵ — to specify the x shift of each field center from the tangent point; $x_n > 0$ shifts the map center to the East (left).

> DECSHIFT $y_1, y_2, y_3, \ldots$ ↵ — to specify the y shift of each field center from the tangent point; $y_n > 0$ shifts the map center to the North (up).

> FLDSIZ $i_1, j_1, i_2, j_2, i_3, j_3, \ldots$ ↵R

to set the area of interest in x and y for each field in turn. Each i_n and j_n is rounded up to the greater of the next power of 2 and the corresponding IMSIZE. FLDSIZE controls the actual size of each image and sets an initial guess for the area over which Clean searches for components. (That area is then modified by the various box options discussed in § 5.3.3.)

If ROTATE is not zero, the shifts are actually with respect to the rotated coordinates, not right ascension and declination. The actual x shift will be RASHIFT which is $(\alpha - \alpha_0)\cos(\delta)$. IMAGR has an optional BOXFILE text file which may be used to specify some or all of the FLDSIZE, RASHIFT, and DECSHIFT values. It is the only way to specify these parameters for fields > 64. To simplify the coordinate computations, the shift parameters may also be given as right ascension and declination of the field center, leaving IMAGR to compute the correct shifts, including any rotation. BOXFILE may also be used to specify initial Clean (and "UNClean") boxes for some or all fields, values for BCOMP, spectral-channel-dependent weights, a list of "star" positions to be added to plots, and even a list of facets to ignore totally.

The manner in which the multi-field Clean is conducted requires some discussion. When ONEBEAM is true, there is a single dirty beam for all fields. For either value of DO3DIMAG, the dirty beams for each facet are, at least subtly, different. Using a single dirty beam allows the task to run faster at some compromise to accuracy. Since the Clean component models are subtracted from the uv data at each major cycle, Clean will correct much of this compromised accuracy. In OVERLAP < 2 mode with one beam, all pixels within Clean windows above the current threshold from all fields are selected for the Clark Clean at the same time. The component flux at which the major cycle terminates is adjusted by the number of iterations before and during that major cycle. All components found from all fields in the major cycle are subtracted at once from the residual data and a new set of residual images is constructed. When ONEBEAM is false, there is a different dirty beam for each field. Thresholds are set by reviewing the data in all fields as above. However, a major cycle is then conducted for each field individually in order of decreasing peak residuals (within the Clean boxes). The first field alone determines the flux at which the major cycle terminates for all fields. Components are subtracted from the residual data one field at a time.

There is a third arrangement, selected by specifying OVERLAP ≥ 2, which is useful if the multiple fields overlap. All fields are imaged at the beginning to allow the user to set the initial Clean boxes. Then, at each cycle, the one field thought to have the highest residual with its Clean boxes is imaged, a major cycle of Clean performed, and the components found subtracted from the residual uv data. The process is repeated using the previous estimates of the maxima (with a revised value for the field just Cleaned). This arrangement requires some extra imaging at the beginning (and occasionally during Cleaning), has some uncertainties about the setting of thresholds and major cycle flux limits, and will invoke the DOTV option for every field individually except at the beginning. It has the benefit of removing the strongest sources (if there is overlap) and their sidelobes from the later fields before they are imaged. This arrangement removes the instabilities that arise if the same spot is Cleaned from 2 fields. IMAGR carefully checks the Clean boxes in OVERLAP < 2 mode and eliminates one of any two overlapping boxes.

It appears that the most reasonable approach would be to use ONEBEAM FALSE ; OVERLAP 2 at the beginning of a deep Cleaning in order to deal carefully with the brightest source components, avoiding putting erroneous components on their sidelobes. But when the dynamic range of the residual image is reduced, ONEBEAM TRUE ; OVERLAP 1 will be accurate enough and much faster. IMAGR has an OVRSWTCH option to control switching from the former to the latter without having to restart the Clean.

There are a number of aspects of multi-field Clean that can trip up the unwary. The first is that the sidelobes of an object found in one field are *not* subtracted from the other fields in the minor Clean cycle. In fact, they are not even subtracted from pixels more than the beam patch size away in the same field. This can cause sidelobes of the strongest sources to be taken to be real sources during the current major cycle. (The OVERLAP ≥ 2 sequence reduces this effect significantly.) At the end of the major cycle, all components from all fields are subtracted from the uv data. At this point, all sidelobes of the components are gone from all fields, but the erroneously chosen "objects" with their sidelobes will appear (in negative usually). This is normally not

a problem. During the next cycle, Clean will put components of the opposite sign on the erroneous spots and they will eventually be corrected. Nonetheless, it is a good idea to restrict the Cleaning to the obvious sources to begin with, saving Clean the trouble of having to correct itself, and to open up the search areas later in the Clean. The TV options make this easy to do in IMAGR; see § 5.3.3.

The situation is more complex if the multiple fields overlap. If a sidelobe in the overlap area is taken as a source in one major cycle, it will appear as a negative source in both fields at the start of the next major cycle (*only when* OVERLAP < 2). Clean will then find negative components in both fields and correct its original error twice, producing a positive "source" at the next major cycle. Such errors never get fully corrected. A simple rule of thumb is never to allow the search areas of one field to overlap with the search areas of another field — or use OVERLAP ≥ 2.. Even then, there is one other "gotcha." In the restore step, Clean only restores components to the fields in which they were found (again, unless OVERLAP > 0). Thus, a real source visible in two fields will be found in only one after Clean; your two images of the same celestial coordinate will be in substantial disagreement. Therefore, you must be careful about which parts of which images you believe to represent the sky. Instead, use OVERLAP ≥ 1 to have Clean components from all fields restored to all fields as needed. If OVERLAP ≥ 2 , the Clean and imaging are done in a fashion which greatly reduces the instabilities arising from Cleaning the same source (or sidelobe) from more than one field.

5.3.3 Clean boxes and the TV in IMAGR

Clean works better if it is told which pixels in an image are allowed to have components. The initial information on this is provided by the FLDSIZE adverb which gives the pixel dimensions of a rectangular window centered in each field in which Clean looks for components. This window can be nearly the full size of the image because the components are subtracted from the ungridded uv data. Cleaning windows or "boxes" can be specified with the adverbs:

> NBOXES n ↵ — to set the number of boxes in which to search for Clean components. Must be ≤ 50; if 0, one Clean box given via FLDSIZE is used and CLBOX is ignored.

> CLBOX $lx1, by1, rx1, ty1, lx2, by2, rx2, ty2, \ldots$ ↵ — to specify the pixel coordinates of the Clean windows as leftmost x, bottom-most y, rightmost x, topmost y for boxes 1 through NBOXES. Circular boxes may also be specified as -1, radius, center x, center y interspersed in any order with the rectangular boxes. Default is given by FLDSIZE(1).

> BOXFILE '$area : infilename$' ↵ — to specify the name of a text file listing the Cleaning windows. Blank means no file.

> OBOXFILE '$area : outfilename$' ↵ — to specify the name of an output text file to list the Cleaning windows after any modifications made while running IMAGR. Blank means a temporary file in $HOME. OBOXFILE can be the same as BOXFILE. Can also be simply an environment variable followed by a colon to be assigned a unique name in that area corresponding to this execution of IMAGR.

The BOXFILE text file is an optional means by which Clean windows may be entered at the start of a run of IMAGR for all fields, not just the first. It is also the only way to enter more than 50 boxes for the first field; the limit is min (2048, 131072/NFIELD) (!) boxes per field with this option. The format of the file is one box per line beginning with the field number followed by the four numbers describing the box as in CLBOX above. Any line in which the first non-blank character is not a number is taken as a comment, a field definition (see § 5.2.2), a BCOMP value, a channel weight, or an "UNClean" box definition. This last begins with a U or u, followed by a field number and a circular or rectangular box which defines a region in which Clean components are not allowed to be found. This is used to force time-dependent components into their

own field in the TDEPEND set of procedures. There may well be other reasons, such as centering a point component on a pixel in its own field, for forbidding the Cleaning of a source in a particular field. NBOXES and CLBOX are overridden if any boxes for the first field are given in the file.

You can use the TV cursor in advance of running IMAGR to set the Cleaning boxes. First, load the TV display with either the dirty image or a previous version of the Clean image of the first field; see § 6.4.1. Then type:

> TVBOX ↵ to begin an interactive, graphical setting of up to 50 boxes, or

> REBOX ↵ to do a similar setting of the boxes, beginning with the NBOXES boxes already in CLBOX.

Position the TV cursor at the bottom left corner of the first Cleaning box and press a trackball or mouse button. Then position the cursor at the top right corner of the box and press Button B. Repeat for all desired boxes. This will fill the CLBOX array and set NBOXES for the first field. Note that the terminal will display some additional instructions. These will tell you how to switch to a circular box and how to reset any of the previously set corners or radii/centers should you need to do so. HELP REBOX ↵ will provide rather more details. The verb DELBOX allows you to delete boxes from CLBOX interactively.

You can also use the TV cursor in a very similar way to build and modify the BOXFILE text file. (You can also use your favorite text editor of course; see § 3.10.1 for general information about specifying and using external text files.) The verb FILEBOX reads the text file (if any) given by BOXFILE selecting those boxes (if any) already specified for the specified field number which fit fully on the current image on the TV. Which field number you want is given with the NFIELD adverb, or, if that is zero, deduced from the Class name of the image on the TV. (Be careful to load the TV with the desired image before running FILEBOX!) You then carry out a graphical setting or resetting of boxes in exactly the same manner as with REBOX. The new and changed boxes are then added to the end of the text file. Different portions of the current field and other fields may be done and redone as often as needed. Verb DFILEBOX may be used like DELBOX to delete boxes interactively from a BOXFILE. The BOXFILE may be edited by task BOXES to put Clean boxes around sources from a source list such as the NVSS or WENSS. The task FIXBX may be used to convert the Clean boxes from the fields and cell sizes of one box file to those of another.

Task FILIT does something like FILEBOX but it does the TV load for you, using a roam mode if the image is larger than the TV display area. From a TV menu, you may then add boxes, change boxes, and delete boxes and even use an auto-boxing feature to add boxes automatically based on the peak values and noise in the image. FILIT will work on a set of facet images and should become the task of choice for examining multi-facet images and their Clean boxes. The non-interactive task SABOX will find boxes to use for a set of facet images, prepared, for example, by a shallow un-boxed Clean. If you need to take your boxes to CASA, MASKS will make a mask image for you.

IMAGR will also create Clean boxes for you automatically. The adverbs IM2PARM(1) through IM2PARM(6) control the process, selecting the maximum number of boxes to be found at any one time, the cutoff levels as factors times the residual image robust rms, and more. IM2PARM(7) controls the starting list of Clean boxes for the next spectral channel — do they start with those input to IMAGR or also include those found in the current channel. See HELP IM2PARM ↵ for information and *AIPS* Memo No. 115 "Auto-boxing for Clean in *AIPS*" by Greisen for a detailed explanation of this new option.

The real power of IMAGR becomes apparent if you set DOTV = n, where NFIELD $\geq n > 0$ is the field number first displayed on the TV. Before each major cycle, the current residual image is displayed on the tv with a labeled step wedge (in 31DEC25) and a menu of options is offered to you. (Note that the residual image before the first major cycle is the un-Cleaned dirty image.) The image displayed is interpolated up or decremented down (by taking every $n^{\rm th}$ pixel in each direction) to make it fit on the display and the current Clean boxes are shown. If you do not select a menu option, IMAGR proceeds after 30 seconds, except initially where the default timeout is 10 minutes. IM2PARM(8) and (9) may be used to alter these timeouts.

The interactive options appear in two columns. To select an option, move the TV cursor to the option (remember the left mouse button — see § 2.3.2) and press buttons A, B, or C. Button D will get you some

on-line help about the menu option. The basic options, in the order in which they are displayed, are:

ABORT TASK	to stop the task abruptly, destroying the output images and exiting as quickly as possible.
TURN OFF DOTV	to resume the Cleaning now and stop using the TV to display the residual images. To turn the TV display back on, if needed, use the TELL IMAGR verb with suitable adverbs, including DOTV TRUE.
STOP CLEANING	to stop the Clean at this point, restore the components to the residual images, write them on disk, and exit.
OFFZOOM	to turn off any zoom magnification
OFFTRANS	to turn off any black & white enhancement
OFFCOLOR	to turn off any pseudo-coloring
TVFIDDLE	to interactively zoom and enhance the display in black & and white or pseudo-color contours as in AIPS
TVTRAN	to enhance in black & white as in AIPS
TVPSEUDO	to select many pseudo-colorings as in AIPS
TVFLAME	to enhance with flame-like pseudo-colorings as in AIPS
TVZOOM	to set the zoom interactively as in AIPS
CURVALUE	to display the pixel value and x, y pixel coordinates at the TV cursor position as in AIPS
IMSTAT	to compute and display robust and normal statistics on the current residual image
SET WINDOW	to select a sub-image of the whole to be reloaded with better resolution — all boxes must be included.
RESET WINDOW	to select the full image and reload the display
TVBOX	to set the Clean boxes for this field beginning at the beginning as in AIPS. If existing boxes will be deleted, you are asked if you want that. If no, REBOX is done instead.
REBOX	to reset the current Clean boxes and create more as in AIPS
DELBOX	to delete some of the current Clean boxes as in AIPS
UNBOX	to make or re-make "UNClean" boxes for this field
CONTINUE CLEAN	to resume Cleaning now rather than wait for the time out period.
CHAR MULT	to change the character size used in the TV display.

A blank OBOXFILE adverb causes a temporary one to be created in $HOME. A TELL operation can change this adverb to a more permanent file. If TVBOX, REBOX, or DELBOX used, the new Clean boxes will be written to the text file, replacing any previously in that file. (All non-box cards in that file are preserved unchanged; a new OBOXFILE will be filled with the non-box cards from BOXFILE.)

If NFIELD > 1, a sufficient number of additional options appear of the form

SELECT FIELD n	to display field n, allowing its Clean boxes to be altered or
SELECT NEW FIELD	to prompt on the terminal for a new field number to be displayed, allowing its Clean boxes to be altered (when > 64 fields).
SELECT NEXT FIELD	to select the next higher numbered field (when > 64 fields).
SELECT LAST FIELD	to select the next lower numbered field (when > 64 fields).

Thus you can look interactively at the initial dirty images, place boxes around the brightest sources, and start the Clean. As it proceeds and weaker source become visible, you can expand the boxes and add more

to include other sources of emission. Do be careful, however. Boxes that are too tight around a source can affect its apparent structure. The author once made Cas A into a square when stuck with a too-tight box. If OVERLAP = 2, the SELECT FIELD options are displayed when all residual images are current, *i.e.*, at the beginning, but are replaced by the options

REMAKE IMAGES	to re-compute all fields using the current residual *uv* data and then to display all fields on the TV.
THIS IS FLD n	to indicate that only field n is displayed and available to have its boxes altered.
FORCE A FIELD	to prompt on the terminal for a field number, exit TV, re-compute and display that field with current residual data (if needed) and then Clean that field (only available if NFIELD > 64 or DOWAIT true).
STOP FLD n	to mark field n as Cleaned sufficiently.
ALLOW FLD n	to allow further cleaning of previously stopped field n.
CHECK BOXES	to see if Clean boxes in one field overlap with Clean boxes in other fields and drop some to avoid this situation.

We encourage use of DOTV TRUE G_R when you are Cleaning an image, especially for the first time or when using the options described in the next section. Watching the TV display as the Clean proceeds will help you to gauge how to set up control parameters for future Cleans and how long to iterate. It may also warn you about instabilities in the deconvolution if you compare the appearance of extended structures early and late in the Cleaning process. The instabilities referred to in § 5.2.4 were first seen while Cleaning with the TV option.

5.3.4 Experimental variations on Clean in IMAGR

Experimental variations of the familiar Cleaning methods have been introduced in IMAGR. One deals with the so-called "Clean bias" which causes the fluxes of the real sources to be underestimated. The other two deal with the inadequacies of Clean in modeling extended sources.

5.3.4.1 Clean-component filtering

It has been found that Clean will eventually assign some components to noise spikes in regions which do not have real sources, producing the "Clean bias." Real source flux is underestimated, presumably because "sidelobes" of the noise "sources" get subtracted from areas of real sources. The magnitude of the effect is rather variable and is not understood. There are two older tasks discussed in § 5.3.6 which deal with the problem. However, it would be better to remove "weak, isolated" (presumably spurious) Clean components as Clean proceeds rather than only after the fact. One cannot do this at every Clean major cycle, since all components are likely to be weak and isolated initially. But it is a good idea to do it a few times while *uv*-plane based Cleans still have the ability to respond to the filtering. Note that this option may be used to remove negative "bowls" surrounding sources imaged with too little short-spacing data. However, it does not remove the bowl that sits on the source itself and so the final flux will be too low. You should consider the multi-scale Clean instead.

To have IMAGR filter Clean components set IMAGRPRM(8) $\neq 0$. Then IMAGR will select only those Clean components having $>$ abs(IMAGRPRM(8)) Jy within a radius of abs(IMAGRPRM(9)) cells of the component. Note that this rejects all areas of negative flux, unless IMAGRPRM(8)< 0, in which case the absolute value of the flux near the component is used. You can change these parameters with TELL, but only if IMAGRPRM(8) was non-zero to begin with. A copy of the input data has to be made for this option and it is only made if

IMAGRPRM(8) is non-zero. If this option is selected, the output CC files will have been merged. Note that IMAGRPRM(8) should always be ≤ 0 for images of Q, U, and V Stokes parameters since negative brightnesses are valid. Filtering is done on restarts, when requested from the TV, on certain Cleaning failures, and near the exit. Clean is continued after this last filtering if IMAGRPRM(9)< 0, but usually terminates fairly quickly. When this option is available, an addition TV option will appear:

FILTER COMPS	to exit TV, filter all Clean components, and then re-compute all fields using new residual data.

This is the only way to filter before the end of the current Clean.

5.3.4.2 SDI modification of Clean in IMAGR

A modified Clean algorithm that attempts (often successfully) to suppress the striping and bumpiness in Cleans of extended sources has been developed by Steer, Dewdney and Ito (1984, *Astron. & Astrophys.* **137**, 159). In the *AIPS* modification of the algorithm, Clean proceeds normally until the residual image becomes rather smooth. It then takes many components at once from all high-residual cells rather than trying to decide exactly which *one* cell is the highest. This algorithm attempts to cut the top off the plateau of emission found in the residual image(s) in a relatively uniform way. Unfortunately, it is very expensive to determine the correct weights to use for each pixel in this algorithm. The technique used by IMAGR does apply larger weights to isolated pixels and pixels at the edges of the "plateau" but these weights are still not quite large enough to avoid a tendency to make a slight rim around the plateau. The next cycle of SDI Clean does trim this down and the method converges well for the extended sources for which the algorithm was designed. IMAGR allows you to switch back and forth automatically between BGC and SDI depending on the contrast between the brightest and median residual pixel in the Clean windows. (SDI is used when the contrast is low.)

To allow IMAGR to use the SDI algorithm, specify IMAGRPRM(4) > 0. SDI Clean will be used when the fraction of pixels in the Clean windows exceeding half of the peak residual exceeds IMAGRPRM(4). IMAGRPRM(19) is used to limit the depth of an SDI Clean cycle. When this option is used, additional TV options appear:

FORCE SDI CLEAN	to force the next Clean cycle to use SDI method.
FORCE BGC CLEAN	to force the next Clean cycle to use Clark method.

These allow you to force the choice of SDI or BGC methods for the next Clean major cycle. After that, it reverts to selecting the method based on IMAGRPRM(4) and the histogram of residual values.

5.3.4.3 Multi-scale modification of Clean in IMAGR

Clean has problems with extended sources because the point-component model is so far from the reality for them. Greisen experimented in the 1970s with modeling sources as Gaussians rather than points, but found that there are always point sources in the image which cannot be modeled sensibly with an extended component. ("Bull's eyes" get painted around every point object.) Wakker and Schwarz (1998, *Astron. & Astrophys.* **200**, 312) proposed a scheme in which a smoothed image and a difference image were Cleaned. This still used point-source models although the Clean beam used for restoring the smoothed-image components was extended. Cornwell and Holdaway (July 1999, Socorro imaging conference) described a scheme in which an image is Cleaned simultaneously at several scales.

A *uv*-based variation of this last algorithm is available in IMAGR. The multi-field capability is used to image for each of NFIELD fields, images at NGAUSS scales specified in the array WGAUSS in arcsec. The full-resolution image is convolved with a Gaussian of width WGAUSS(*i*) while a dirty beam appropriate to a component of that width is constructed. One of the WGAUSS must be zero if a point-source model is desired; a warning is issued if none of the resolutions is zero. OVERLAP = 2 mode is used. See EXPLAIN IMAGR for details

of this new option. Users have found that WGAUSS values increasing by factors of $\sqrt{(10)}$ are frequently optimal. The most important additional control parameter is IMAGRPRM(11) which down-weights higher scales to allow Clean to work on the higher-resolution images with roughly equal probability. FGAUSS is used to select the minimum brightness to be cleaned at each scale; higher values at higher scales are usually desired. Reasonable values of these three parameters will require running IMAGR to determine the point-source resolution and then to determine the peak brightnesses in each scale and the apparent noise levels after some Cleaning. The other available "knobs" for this algorithm may safely be left at zero.[3]

5.3.4.4 Spectral-index corrections

When using wide-band, multi-channel data to image continuum sources, serious effects are seen if the source intensities change as a function of frequency. If these changes can be modeled as spectral index images with or without "curvature," then IMAGR will allow the spectral-index effects to be compensated during Cleaning. First image each spectral channel (or group of closely-spaced channels) separately. Combine them into a cube with FQUBE, transpose the cube with TRANS, and solve for spectral index images with SPIXR. To use these images, set IMAGRPRM(17) to a radius (> 0) in pixels of a smoothing area and put the image name parameters in the 3rd and 4th input image names. Note that this algorithm is expensive, but that it can be sped up with judicious use of the FQTOL parameter.

5.3.5 Data correction options in IMAGR

There are a number of effects which degrade the usual image deconvolution, but which are, optionally, handled differently by IMAGR. These corrections are primarily for observations made with widely spaced frequencies over fields comparable to the single-dish field of view. If you have such data and hope to achieve high dynamic range images, then these corrections are for you. Otherwise skip to the next section.

5.3.5.1 Frequency-dependent primary-beam corrections

The primary-beam pattern of the individual telescopes in the interferometer scales with frequency. Therefore, each channel of multi-frequency observations of objects well away from the pointing center effectively observes a different sky. When a combined source model is produced, there will be residuals in the visibility data that cannot be Cleaned as the data does not correspond to a possible sky brightness distribution. If IMAGRPRM(1) is larger than 0, then a correction is made in the subtraction of Clean components from the *uv* data to remove the effects of the frequency dependence of the primary beam. If the array is known, *i.e.*, if it is the old VLA, new EVLA, ATCA, GMRT, or MeerKAT, the known primary beam parameters of the average telescope in the array are used. Otherwise, the primary beam is assumed to be that of a uniformly illuminated disk of diameter IMAGRPRM(1) meters. This correction is made out to the 5% power point of the beam with a flat correction further out. Note: this correction is only for the relative primary beam to correct to a common frequency and *does not* correct for the primary beam pattern at this frequency. Note that this algorithm is expensive, but that it can be sped up with judicious use of the FQTOL parameter. It adds essentially no cost when doing the spectral-index corrections, however.

[3] See "Aperture Synthesis Observations of the Nearby Spiral NGC 6503: Modeling the Thin and Thick Disks" 2009, AJ, 137, 4718, by Greisen, E. W., Spekkens, K. and van Moorsel, G. A. for a detailed description of this algorithm and an example of its use.

5.3.5.2 Frequency-dependent correction for average spectral index

If the sources observed do not have a flat spectrum, then the source spectrum will have channel-dependent effects on the Cleaning of a similar nature to the primary beam effects described above. This problem does not depend on position in the field except, of course, that the spectral index usually varies across the field. Normally, however, it varies around -0.8 rather than about 0. To the degree that the structure in the field can be characterized by a single spectral index, the amplitudes of the data can be scaled to the average frequency. This is done, before imaging, by scaling the amplitudes of the uv data to the average frequency using a spectral index of IMAGRPRM(2). For optically thin synchrotron sources, this spectral index is typically between -0.6 and -1.0. This correction cannot remove the effects of variable spectral index but allows a single correction which should usually be better than no correction at all. Note that a much more expensive correction and more accurate may now be made (see above).

5.3.5.3 Error in the assumed central frequency

If the frequency used to compute the u, v and, w terms is in error, there will be a mis-scaling of the image by the ratio of the correct frequency to that used. Since central frequencies are frequently computed on the basis of unrealistic models of the bandpass shape, the "average" frequency given in data headers is frequently in error. If IMAGRPRM(3) is larger than 0, it is assumed to be a frequency scaling factor for the u, v, and w that is to be applied before imaging. Again, this can only correct for some average error. Since individual antennas will have different bandpass shapes, no single factor can correct all of the error.

5.3.5.4 Array mis-orientation effects

Images made with a coplanar array not oriented towards the instrumental zenith will have a distortion of the geometry which increases in severity away from the phase tracking center. For non-coplanar arrays, the image is distorted rather than just the geometry. VLA snapshots are misaligned coplanar arrays, whereas VLA synthesis images cannot be considered to have been made with a coplanar array. Images made with mis-aligned coplanar arrays can be corrected using task OHGEO to remove the effects of this misalignment. Since this correction requires the knowledge of the observing geometry, in particular, the average parallactic and zenith angles, IMAGR computes these values and leaves then as header keywords for OHGEO to use.

5.3.5.5 Non-coplanar effects

IMAGR has a IMAGRPRM(4) option to attempt to correct for non-coplanar effects in imaging. If this worked, it would be very very slow. At this writing, it is not believed to work at all and is disabled in the code. See the explain information for further details. The DO3DIMAG option removes a good part of the non-coplanar effects by rotating the projected baselines to make each field tangent at its center.

5.3.5.6 Units mismatch of residuals and Clean components

In principle, the units of the residuals are different from those of the restored components. Both are called Jy per beam area, but the beam areas differ; that of a dirty image is — in principle — zero. If the area of the central lobe of the dirty beam is similar to the restoring beam area, then this effect is negligible. Similarly, if the Clean has proceeded well into the noise then this difference is of little consequence. However, if there is significant flux left in the residual image, then this difference may be important. If IMAGRPRM(5)> 0, IMAGR

will attempt to scale the residuals to the same units as the restored components. The principal difficulty is determining the effective area of the dirty beam. Operationally, this is done inside a box centered on the peak in the beam with half-width IMAGRPRM(6) in x and IMAGRPRM(7) in y. It may be better, in this regard, to use the default Clean beam and this option at this stage and change resolution and units later with CCRES; see § 7.6.4.

5.3.6 Manipulating Clean components

The list of Clean components associated with a Clean image can be printed with:

> TASK 'PRTCC' ; INP	to select the task and review its inputs.
> INDI n ; GETN ctn ↵	to select the Clean image, where n and ctn select its disk and catalog numbers.
> BCOUNT n_1 ; ECOUNT n_2 ↵	to list Clean components from n_1 to n_2.
> XINC n_3 ↵	to list only every n_3^{th} component.
> DOCRT FALSE ↵	to route the list to the line printer, or use TRUE to route the display to your workstation window.
> GO ↵	to execute the task.

Some users of the CC file for self-calibration suggest that only the components down to the first negative, or down to some factor times the flux at the first negative, should be used. The justification for this advice is the assumption that negative components occur near the noise level. This is not always the case. They also occur to correct for previous over-subtraction or for an object which does not lie on a cell. In any case, PRTCC will display the first negative component if it is found during the printing (*i.e.*, before or during the range printed). The task CCFND is designed solely to find the component number of the first negative and the number of the component having FACTOR times the component flux of that first negative. The total fluxes at these two positions in the file are also displayed.

You can plot the list of Clean components associated with a Clean image in various ways with TAPLT. For example, to plot the sum of the components as a function of component number enter:

> APARM 0 ; BPARM 0 ; CPARM 0 ↵	to clear input parameters.
> APARM(6) 1; APARM(10) 1 ↵	to have the component flux summed and plotted on the y axis.
> GO TAPLT ↵	to create the plot file.
> GO LWPLA ↵	to display the plot file on the laser printer.

TAPLT offers many options for plotting functions of table columns against each other. Enter EXPLAIN TAPLT ↵ for details.

You can compare the source model contained in the CC file with the visibility data in a variety of ways. UVSUB allows you to subtract the components of some or all fields from the data, producing a residual visibility data set. OOSUB is similar, but allows for the frequency-dependent corrections performed in IMAGR and for limiting the components subtracted to those inside or outside of the primary beam. Of course, IMAGR's workfile already contains these residuals with the CC files of all fields subtracted. Various display options can be used on these *uv* files; see § 5.3.1. VPLOT, described in § 9.4.3, will plot a CC model against visibility data, one baseline at a time, n baselines per page.

The algorithm used by all *AIPS* Cleans assigns to a component only a fraction (GAIN) of the current intensity at the location of that component. As a result, the list of components contains many which lie on the same pixels. CCMRG combines all components that lie on the same pixel. This can reduce the size of the list greatly and, hence, the time required for model computations in tasks such as CALIB (§ 5.4) and UVSUB. Do this with

> TASK 'CCMRG' ; INP	to select the task and review its inputs.

> INDI *n* ; GETN *ctn* ↵ — to select the Clean image, where *n* and *ctn* select its disk and catalog numbers.

> INVERS *m* ; OUTVER *m* ↵ — to select the input version of the Clean components and to replace it with the compressed version.

> GO ↵ — to execute the task.

Under a variety of conditions, the Clean component files produced by IMAGR will already be merged.

There should seldom be a need to edit Clean component files in detail. However, task TAFLG allows editing based on comparison of a function of one or two table columns with another function of another one or two columns. One interesting use for TAFLG would be to delete all components below some cutoff before running CCMRG. Enter EXPLAIN TAFLG ↵ for details.

It has been found that Clean will eventually assign some components to noise spikes in regions which do not have real sources and that this produces the so-called "Clean bias" which causes the fluxes of the real sources to be underestimated. This is presumably because "sidelobes" of the noise "sources" get subtracted from areas of real sources, but the magnitude of the effect is rather variable and is not understood. There are two tasks which can help. CCEDT copies a CC file keeping only those components which occur in specified windows. Then it merges the file (like CCMRG) and discards all merged components of flux below a specified cutoff. Under some circumstances, such filtering of Clean components before self-calibration can be a more effective way of obtaining convergence of hybrid mapping (mostly for VLBI) than restricting Clean windows in IMAGR.

The second task, CCSEL, explicitly addresses the Clean bias problem. It sums the flux of all components within a specified distance of each component and then discards those components for which this sum is less than a specified threshold. The idea is to eliminate "weak isolated" components which are likely to be those on noise points. You should run CCMRG before using CCSEL since the compute time increases quadratically with the number of components. IMAGR's internal algorithm for filtering is much more efficient.

5.3.7 Image-plane deconvolution methods

The previous sections have described the task IMAGR which implements Clean by subtracting model components in groups from the ungridded *uv* data and re-imaging. This can be rather expensive. If you have a significant number of visibilities contributing to a fairly small image, it may be faster to use an image-plane deconvolution method. The venerable APCLN implements the Clark Clean in the image plane. Clean components are found during "minor" iteration cycles by Cleaning the brightest parts of the residual image with a "beam patch" of limited size, just as in IMAGR. More precise Cleaning is achieved at the ends of "major" iteration cycles when the Fourier transform of the Clean components is computed, multiplied by the transform of the beam, transformed back to the image plane, and then subtracted from the dirty image. This method does a good job Cleaning the inner quarter of the image area, but artifacts of the Cleaning and aliasing of sidelobes do interesting things to the remaining 75% of the image. Make the dirty image using IMAGR and be sure to make it large enough to include all of the source in the inner quarter of the area. APCLN uses many of the now-familiar adverbs of IMAGR, including GAIN, FLUX, NBOXES, CLBOX, FACTOR, MINPATCH, MAXPIXEL, BMAJ, and more. APCLN recognizes only rectangular boxes and its DOTV option only displays the residual image with a pause for you to hit button D to end the Cleaning early. The input image and beam should have size an integer power of 2 and the reference pixel should be at $N_x/2$, $N_y/2+1$.

The subject of image deconvolution has been widely studied and many methods have been proposed for tackling it. Clean is renowned for yielding images that contain many artificial beam-sized lumps or stripes in smooth low-brightness regions. Point sources are a poor model for such regions. You should compare heavily Cleaned images with dirty, or lightly Cleaned, images to test that any features you will interpret

physically have not been introduced by these Clean "instabilities." The *AIPS* Clean tasks have an optional parameter PHAT that will add a small-amplitude δ-function to the peak of the dirty beam in an attempt to suppress these instabilities as described by Cornwell (*Astron. & Astrophys.* **121**, 281 (1983).)

A modified Clean algorithm that attempts (often successfully) to suppress these instabilities has been developed by Steer, Dewdney and Ito (*Astron. & Astrophys.* **137**, 159 (1984)). In this algorithm, Clean proceeds normally until the residual image becomes rather smooth. It then takes many components at once from all high-residual cells rather than trying to decide exactly which *one* cell is the highest. The algorithm is embodied in the well-tested *AIPS* task SDCLN, which is actually an enhanced version of APCLN. The source must be contained in the inner quarter of the image area as in that task. Type EXPLAIN SDCLN ↵ for information. SDCLN gives excellent results on extended sources, but is exceptionally CPU-intensive.

The most widely used, best understood, and probably most successful alternative to Clean is the Maximum Entropy Method ("MEM"). This is implemented in *AIPS* by the task VTESS. This requires a dirty image and beam, such as those produced by IMAGR with NITER set to 0, each twice the (linear) size of the region of interest (as for APCLN and SDCLN). The deconvolution produces an all-positive image whose range of pixel values is as compressed as the data allow. The final VTESS image is therefore stabilized against Clean-like instabilities while providing some "super-scale" wherever the signal-to-noise ratio is high. VTESS can also deconvolve multiple images simultaneously; see below.

There are three main reasons to prefer MEM deconvolution over all of the Clean deconvolution methods:

1. MEM can be much faster for images which have strong signals in many pixels. "Many" seems to be $\geq 512^2$ or so.

2. MEM produces smoother reconstructions of extended emission than does Clean.

3. MEM allows introduction of *a priori* information about the source in the form of a "default" image.

Because VTESS can produce excellent deconvolutions of extended sources in much less computation time than Clean, but requires careful control, we recommend studying the output of EXPLAIN VTESS ↵ before using the task. The NOISE parameter is particularly important; some have claimed that VTESS requires this to be within 5% of the correct value in order to deconvolve fully without biasing the total flux. (Use IMEAN (§ 7.3) to estimate the true rms.) Chapters 8 and 15 of the NRAO Summer School on *Synthesis Imaging in Radio Astronomy* also provide useful general background.

MEM can be used for quantitative work on regions of good (> 10) signal-to-noise ratio, if the dirty image is convolved with a Clean beam prior to deconvolution. Use the *AIPS* task CONVL for this purpose. The images may also be post-convolved, and added to the residuals, within VTESS. In many cases, the images of extended sources produced by SDCLN and VTESS are functionally identical. VTESS usually converges in *much less* CPU time, however, at the expense of leaving significantly larger residual sidelobes close to bright compact (point-like) features. To get around this deficiency of VTESS, first use Clean to remove the peaks of bright point-like features, then run VTESS on the residual image produced by this restricted Clean. (The *AIPS* Clean tasks will output a residual image if you set BMAJ < 0.) The Clean components may be restored to the final image with tasks CCRES or RSTOR.

VTESS can also combine information from different types of data. For example, single-dish data can be used to constrain the imaging of interferometer data, or many pointings covering one large object can be processed together. VTESS takes up to 4087 pairs of images and beams, together with some specification of the primary beam for each, either a circular Gaussian model or the VLA primary beam, and performs a joint maximum entropy deconvolution to get an image of one field. The images must all be in the same coordinate system, and a noise level must be known for each. The time taken is approximately the time VTESS would take for one input map and beam, multiplied by the number of map/beam pairs.

VTESS cannot be used on images which are not intrinsically positive, such as images of the Stokes Q, U, and V parameters. UTESS is a version of VTESS designed to deconvolve polarization images, for which a positivity constraint cannot be applied. For further information type EXPLAIN UTESS ↵

Two further alternatives to Clean have been implemented in *AIPS* as *experimental* tasks. These are algorithms due to Gerchberg and Saxton (APGS) and van Cittert (APVC). Type EXPLAIN APGS ↵, EXPLAIN APVC ↵ for further information on these tasks.

5.4 Self-calibration

The task CALIB was described in some detail in Chapter 4 as the tool to determine the instrumental gains on calibrator sources which were then interpolated in time and applied to your program sources. If you have sufficient signal to noise in the latter, you may now use CALIB to improve the dynamic range of your images. The assumption is made that your images have been degraded by antenna-based (complex) gain errors which vary too rapidly with time or direction to have been fully calibrated with the calibrator sources. CALIB compares the input *uv* data set with the predictions of a source model — a point-source initial guess or your current best set of Clean components — in order to compute a set of antenna-based amplitude and phase corrections as a function of time which would bring the data into better agreement with your current model. For an n-element array, there are $(n-1)/2$ times more observations than unknown antenna gains at any time, so the process is well-determined when n is reasonably large. Since this process uses the data to calibrate themselves, it is called self-calibration.

Do not use CALIB unless your data have enough signal-to-noise to warrant improvement. Ask yourself whether your externally-calibrated Clean images contain un-Cleanable artifacts well above the noise, and whether your source meets the criteria for self-calibration given by Tim Cornwell and Ed Fomalont in Lecture 9 of *Synthesis Imaging in Radio Astronomy*. Note that if your images are limited by receiver noise, self-calibration may produce erroneous results, including the fabrication of weak sources!

5.4.1 Self-calibration sequence and SCMAP or SCIMG

If you decide to use self-calibration, a good sequence of steps is:

1. Use UVPLT to make a plot file showing the shape of the visibility function as a function of baseline length in the externally-calibrated data set. (See § 6.3, especially § 6.3.1, for information about plotting in *AIPS*.) To reduce the scatter, you may average the data over time and/or over spectral channel before plotting. UVPLT normally prepares the plot in memory before writing it to the TV or plot file. In this way, even very large data sets do not require large amounts of time. Use LWPLA to get hard copy of the plot file.

2. If you can use a point-source model for the first iteration, *i.e.*, if a range of baselines sufficient to calibrate all antennas is dominated by a single component (flat visibility function well above the noise), go to step 6 directly. This is frequently done with VLBI data, but is less common with arrays for which the initial calibrations are better, such as the VLA.

3. If you must use a more complicated model, obtain a Clean-component representation of it by making and Cleaning an image of the externally-calibrated data using IMAGR. Leave the *uv* data in "TB" sort order for CALIB; IMAGR will sort them if it has to. Note that you may want to use a somewhat higher loop GAIN in a Clean to be used as an input model for an early iteration of self-calibration than you would for final deconvolution of a very extended structure. Task FACES can prepare an initial model for wide-field observations particularly at long wavelength.

4. Consider running CCMRG to reduce the number of components in the model. This improves the speed of the calibration and makes the first negative component be a real negative rather than a minor correction to previous positive components. Remember that merging the components does alter the model which is used to compute the gains unless you were going to include all components anyway. Note that IMAGR often merges the components automatically.

5. Use PRTCC or TAPLT (as in the example in § 5.3.6) to help you decide how many components from this Clean to include in the CALIB model. CCFND is also helpful. When you have decided this, determine the appropriate *uv*-limits for the gain solution by referring to the hard copy of the visibility function you made at step 1.

6. Plan your CALIB inputs using the information given in the following two sections. The first few iterations are usually used to correct only phases; amplitude is normally corrected only in the last one or two iterations.

7. Use CALIB to calculate the gain corrections. If DOAPPLY ≥ 0, it will apply them to produce a new, (hopefully) improved data set, and will also catalog the gain corrections as an SN extension to the *input uv* data file. You may set DOFLAG to produce and use new data flags based on closure failures.

8. Use SNPLT on the input data file with DOTV = TRUE to review the gain corrections before proceeding further. Use DOBLANK = 1 to plot failed solutions as well as good ones. To take hard copy for future reference, run SNPLT with DOTV = FALSE and then run LWPLA on the plot files (usually more than one) produced. To plot the extrema of the gains use OPTYPE = 'SUM' in SNPLT.

9. Ask whether the gain corrections were believable — were they smaller than at the previous iteration of CALIB, if any? If not, is there a good reason why not? Did you change input parameters such as the model, the type of solution, or the solution interval, in a way that may have forced larger corrections than before? Proceed only if you are reasonably sure you understand what is happening at this point — otherwise consult a local expert at your site.

10. If the corrections were believable, run IMAGR to produce a new Clean image. Lower GAINs and higher NITER to produce deeper and more careful Cleans are appropriate as the self-calibration progresses.

11. Go back to step 4 and repeat the whole process if your new Clean image is a significant improvement over the previous one (with comparable Cleaning parameters on both occasions). You may want to go back to step 1 and repeat the process from there if you have been using amplitude self-calibration and wish to check that your amplitude calibration has not drifted significantly. If the new Clean image differs little from the previous one, do not continue on with further iterations of steps 4 through 10 unless you feel you can make an informed change to the CALIB input parameters at step 6. Tasks UVDIF (§ 6.2.1), DIFUV, and UVADD may help you to decide whether there have been significant changes to your data due to the previous iteration of CALIB.

12. For very high dynamic range imaging, it may be useful to check for baseline-dependent (non-closing) multiplicative complex gains. Tasks BLCAL and BLCHN perform this calibration which is normally run to get a single solution for the entire range of times in the data set. In 31DEC23, procedure BLPOLCAL extends this operation to the cross-hand visibilities. See § 9.5.9 for additional discussion.

13. Images may also be used as models (CMODEL='IMAG'). This works well (in 31DEC21) if the image is a "model image" rather than a Clean image. The Fourier transform of the latter does not agree with the visibility data due to the convolution with the Clean beam. One could experiment, in 31DEC21, with a Clean image in UVSUB with OPCODE='MODL' followed by DGAUS which (partially) corrects for the deconvolution. A UVADD with OPCODE='DIV' will construct an "already divided" data set which is understood by CALIB.

If you have used the spectral-index (§ 5.3.4.4) and/or primary beam (§ 5.3.5.1) options in IMAGR, you may continue to use those options in self-calibration. The procedure OOCAL will divide your multi-facet, channel-dependent model into the visibility data with task OOSUB. It then runs CALIB on that output and copies the resulting SN table back to the original input file.

The tasks SCMAP and SCIMG attempt to implement this sequence inside a single task. SCIMG contains almost all of IMAGR and all of CALIB. SCMAP is similar, but limited to a single field for simplicity. They attempt to make the decision about the number of merged components and the range of *uv* spacings to use in each self-calibration based on σ times the rms in the residual image of the current Clean, where you provide the σ. The process is somewhat less flexible, but also less painful, than running CALIB and IMAGR multiple times. They do not let you change imaging parameters while they are running, but they do provide automatic and interactive methods to change and save Clean boxes and to set a variety of Cleaning and self-calibration parameters including loop gain, solution interval, solution smoothing interval, and whether negative components are included or terminate the list. They let you switch from phase-only to amplitude and phase self-calibration or they will do it automatically when the phase only stops converging. Both tasks offer the full editing options of task EDITR (see § 5.5.2) displaying the input and current residual *uv* data with a wide variety of data selection and editing options. The CALIB, IMAGR, and EDITR process is similar, but conceptually simpler, so it is the one described here.

5.4.2 Self-calibration with CALIB

CALIB is the heart of the *AIPS* calibration package. The inputs to CALIB are extensive and spread over several screen pages. This is because the routines in CALIB are used in many situations — general calibration, real-time interferometry and VLBI. The task solves for antenna-based complex gains *i.e.*, "self-calibration," whether the source being calibrated is a "calibrator" source (usually taken to be a point) or a "program" source (usually taken to be complex). The solutions that CALIB generates are stored in SN "solution" tables which are attached to the *input* data file. The SN tables can be plotted with SNPLT and listed with LISTR. They can be edited themselves with SNEDT or be used to edit the *uv* data with EDITA. If the corrections are usually small, the OPTYPE='A&P' option in SNFLG may perform the flagging you would do with EDITA, but with very little effort on your part.

The following input parameters are used by CALIB for self-calibration of a single-source *uv* data set:

> TASK 'CALIB' ; INP ↵	to specify the task and review the inputs.
> INDI *n1* ; GETN *ctn1* ↵	to select the 'TB' sorted *uv* database.
> IN2D *n2* ; GET2N *ctn2* ↵	to select the Clean model image(s) to use.
> INVERS m ↵	to specify the CC file version number to use from *every* model image; 0 means the highest.
> NMAPS q ↵	to specify the number of images with CC files to use for the model. If $q > 1$, the image class names are assumed to have the first three characters of IN2CLASS with the field number one given in the last three characters as is done by IMAGR.
> NCOMP = $n_1, n_2, \ldots$ ↵	to cut off the model at the $n_i^{\rm th}$ Clean component in the $i^{\rm th}$ image.
> SMODEL S, x, y, m ↵	to specify a point-source (or Gaussian or uniform spherical) model rather than a Clean component model. CALIB uses a source model (type m) of S Jy located at x, y arc-sec with respect to the pointing center. For a point model m = 0; see the help for details of the other types.
> SUBARRAY s ↵	to select the appropriate sub-array — SUBARRAY = 0 implies all sub-arrays.
> UVRANGE = x_1, x_2 ↵	to give full weight (in doing the gain solutions) only to data from projected baselines between x_1 and x_2 in kilo wavelengths.

> WTUV w ↵ — to set the weight for projected baselines outside the range UVRANGE(1) → UVRANGE(2). WTUV = 0 is interpreted as zero weight and should *not* be used.

> REFANT n_r ↵ — to select the reference antenna; for best results, choose one known to be good over most of the time range.

> SOLMODE 'A&P' ↵ — to solve for amplitude and phase corrections simultaneously.

> SOLMODE 'P' ↵ — to solve for phase weighted by amplitude, the default for single-source files.

> SOLMODE 'P!A' ↵ — to solve for phase ignoring amplitude.

> SOLTYP ' ' ↵ — to use a normal (non-linear) least squares solution.

> SOLTYP 'L1' ↵ — to use an "L1" solution method in which a weighted sum of the moduli of the residuals is minimized. The computed gain solutions are less influenced by wild data points, but there is some loss of statistical efficiency and a modest increase in compute time. See F. R. Schwab, VLA Scientific Memo #136 for further details.

> SOLTYP 'GCON' ; SOLMOD 'GCON' ↵ — to solve for amplitude and phase using least squares with a gain constraint — this requires GAINERR and SOLCON as well; see the help file.

> ANTWT $w_1, w_2, w_3, \ldots$ ↵ — to apply additional weights to each antenna (in order) in generating the solutions; 0 implies 1.

> APARM(1) = x_5 ↵ — to reject solutions from fewer than x_5 antennas; default is 6.

> APARM(2) = x_6 ↵ — to tell CALIB whether the data have already been divided by a model ($x_6 > 0$) or not.

> APARM(3) = x_7 ↵ — to solve for RR and LL separately ($x_7 \leq 0$) or to average RR and LL correlators before solving ($x_7 > 0$).

> APARM(5) = x_8 ↵ — to make separate solutions for each IF ($x_8 \leq 0$) or to average all IFs to make a single solution ($x_8 > 0$). It is better to do separate solutions unless you are desperate for signal to noise.

> APARM(6) = x_9 ↵ — to set the level of diagnostic information as 0 (very little), 1 (some including time and closure error statistics), 2 (more including individual closure failures), 3 (even more including S/N ratio), or more (too much or much too much).

> APARM(7) = x_{10} ↵ — to discard solutions having S/N ratios $< x_{10}$; default is 5.

> SOLINT = x_{11} ↵ — to set the length of the solution interval (in minutes); default is 10 seconds for single-source files.

> NORMALIZE = 1 ↵ — to scale the gain corrections by the mean modulus of all gains to keep the flux density scale from drifting; ≤ 0 lets the gains float free. NORMALIZE = 3 to normalize over antennas and polarizations, but not subarrays and IFs, is especially useful for wide bandwidth data.

> MINAMPER a_1 ↵ — to set the level of amplitude closure error regarded as "excessive" to a_1 per cent. If APARM(6)$\geq$ 1, summaries of the number of excessive errors by antenna are printed and, if APARM(6) $>$ 1, up to 1000 of the individual failures are printed. 0 means do not check or report amplitude closure errors of any sort. Note that amplitude closure errors are accumulated using logarithms so that gains of 0.5 and 2.0 are both errors of 100%. errors are reported only if thy are "significant" following CPARM(7).

> MINPHSER p_1 ↵	to set the level of phase closure errors regarded as "excessive." APARM(6) controls the display as for MINAMPER.
> CPARM(3) = a_2 ↵	to display a line when the average absolute value of amplitude closure errors is $> a_2$ % if $a_2 > 0$ and APARM(6) ≥ 1.
> CPARM(4) = p_2 ↵	to display a line when the average absolute value of phase closure errors $> p_2$ degrees if $p_2 > 0$ and APARM(6) ≥ 1.
> CPARM(5) = 1 ↵	to form scalar averages of amplitudes before doing solutions. This is useful only if the phases are bad, but the amplitudes have high signal to noise.
> CPARM(7) = N_σ ↵	to display excessive amplitude and phase errors individually only if they exceed MINAMPER or MINPHSER) and if their significance exceeds N_σ times the uncertainty implied by the data weights.
> DOAPPLY -1 ↵	to solve for a new SN table without writing a new corrected uv data set.

Other parameters are defaulted sensibly — type EXPLAIN CALIB ↵ for further information. In general, the *AIPS* philosophy is such that if you don't know what value to set for an adverb, leave it at the default — this will usually give you what you want, or at least something reasonable! There are two approaches to the self-cal loop. In one, a new uv file is written at each iteration and the next SN table should then be an incremental improvement over the previous. In the other, the initial single-source file is the input file in each iteration and the new SN table contains all of the corrections so far found. Both methods have things to recommend them; DOAPPLY controls the choice.

5.4.3 Considerations in setting CALIB inputs

In many cases, only a few input parameters to CALIB need be set, other than those selecting the uv data and the input model. The key parameters are NCOMP, UVRANGE, SOLINT and, if you are interested in polarization, REFANT.

It pays to be conservative when using NCOMP to select the number of Clean components which will comprise the input source model. Setting NCOMP too high will fossilize errors from the earlier calibrations in the model for the next one; after this, you are stuck with them as long as you continue feeding CALIB a model with as much Cleaned flux density. When calibrating Stokes I images, consider setting NCOMP in CALIB so that few negative Clean components are included. The first few iterations of CALIB should be phase-only calibration, since the tropospheric and ionospheric phase errors will almost always dominate amplitude errors due to the atmosphere or to system drifts. In these first iterations, it is prudent to be even more conservative, setting NCOMP so that the total Cleaned flux included in the model is between 50% and 80% of that at which the first negative Clean component appeared. CCFND will help you with this (§ 5.3.6). If your field is dominated by a few very strong, small-diameter regions, it is a good idea to make the first iterations of CALIB work on Clean components from these regions alone, restricting the range of baselines suitably by setting UVRANGE(1). Setting Clean windows in IMAGR or using CCEDT (§ 5.3.6) suitably will help you do this. Even later in the self-calibration cycle, it is probably still a good idea to eliminate weak, isolated Clean components. Try CCSEL for this.

It is always important to restrict the high-weight domain of the CALIB solution to the part of the uv plane that is described well by the model. In the early stages of self-calibration, the trustworthy part of your Clean model will almost always contain less flux density than was measured in the visibility function at the shorter baselines. Another way of putting this is that the large-scale structure of the source will be poorly represented by the model. You should therefore set UVRANGE(1) so that the total flux density in the

input model (the sum of the Clean components up to the Clean iteration selected by NCOMP) exceeds the peak visibility amplitude in your data at a baseline of UVRANGE(1) kilo wavelengths (read this off a plot file output from UVPLT). It is also important to give some slight weight to the rest of the *uv* plane so that some solution may be found for most all antennas including those having no baselines in the high-weight region.

SOLINT sets the length of the time interval, in minutes, over which the model and the data are averaged when computing the gain corrections. This must be *short* enough that the gain corrections can track the fluctuations produced by the atmosphere over the longer baselines with sufficient accuracy. It must be *long* enough that the variances of the computed gain corrections (which depend on the signal-to-noise ratios in the data over the *uv* range in which the model is being compared with the data) are acceptably small. These constraints vary from source to source, frequency to frequency, and (because of the "weather") from day to day. They may not in fact be reconcilable for weak sources, especially in the wider VLA configurations and/or at the higher frequencies. In many combinations of these circumstances, you may not be able to self-calibrate your data. See Lecture 9 in *Synthesis Imaging in Radio Astronomy* for details of how to make this assessment. In VLBI imaging, it may be helpful to use a point-source model and quite small SOLINT for the first iteration of self-calibration to remove the gross and rapid changes due to atmospheric fluctuations. With that problem removed, it may then be possible to use longer SOLINTs and more complicated models.

REFANT selects the number of the reference antenna for the gain solutions. For total intensity continuum calibration, the choice of this CALIB input is unimportant. It is always best, however, to choose a reference antenna that was stable and present in all data throughout the run, if only because this prevents propagation of noise or glitches in the reference antenna through the gain solutions (and plots of them) for the other antennas. For polarization work, it is important to select an antenna for which both polarizations were always present; otherwise any polarization calibration which preceded CALIB may be seriously compromised.

Note that CALIB should almost always be run with SOLMODE set to phase-only calibration for the first iteration or two. Consider turning on amplitude calibration by setting SOLMODE 'A&P' only when either the phase adjustments being made are generally small (*i.e.*, the worst cases being a few tens of degrees) or the new re-Cleaned image is clearly dominated by amplitude errors — which will give symmetric Y-shaped patterns around strong point sources for VLA observations. In general, you will want to set CPARM(2) = 1 when using SOLMODE 'A&P', to prevent drifting of the flux-density scale during amplitude self-calibration.

CALIB has a number of new options to deal with difficult data. The adverb WEIGHTIT controls how the data are weighted when being processed by the gain-fitting routines. The default is $w = 1/\sigma^2$ which may cause too much contrast between the highest weighted points and the lowest. This problem is much worse when self-calibrating extended sources than when doing the primary calibration on point sources. If you encounter many failed solutions, try WEIGHTIT = 1 which uses $w = 1/\sigma$. If you have trouble with bad data and failed solutions, consider trying the "robust" forms of SOLTYPE selected with 'R', 'L1R', and 'GCOR'. A robust solution is one in which a solution is found, outlier data are temporarily flagged, a new solution found, and the process repeated while gradually tightening the omission criteria. This should make for more reliable solutions when there are bad correlators or antennas and, as a side benefit, allows more permanent flagging of the data under control of adverb DOFLAG.

CALIB will also edit out bad data according to the following criteria:

1. there are too few antennas (APARM(1)) to form a solution,
2. the solution does not converge, or
3. the signal-to-noise ratio for a given antenna (APARM(7)) is too low.

The signal-to-noise ratio is calculated from the post-fit scatter of the residuals from the gain model. Note that the scatter will contain contributions from thermal noise *and* unmodeled source structure. This is a

good reason to restrict the *uv* range of the data. For further guidance and information on other CALIB inputs, type EXPLAIN CALIB ↵ and/or read Lectures 9 and 16 in *Synthesis Imaging in Radio Astronomy.*

We note again the recommendation from Chapter 4. It is best to self-calibrate the parallel-hand data of the target source before applying the polarization corrections (D terms) to the cross-hand data. The correction to the cross-hand data is the product of the D terms found by PCAL and the parallel-hand data. The latter are not fully correct until after the self-calibration step. There is almost always adequate signal for self-calibration when there is measurable polarization.

5.4.4 Evaluating the quality of the imaging

If your imaging were perfect, then subtracting the Clean component model from the visibility data should produce visibilities differing from zero only by the noise. Similarly, dividing the data by the model should produce "gains" of (1,0), namely a 1 "Jy" point source at the coordinate origin. There is a task called EVAUV to help you find out how close you have come to these ideal results. EVAUV first produces two data sets by subtracting the model from, and dividing the model into, your visibility data. It then determines the mean and rms of the real and imaginary parts of both data sets using a robust algorithm. Then it reports on the number of "bad" samples more than 3 rms from the mean. In 31DEC22, if SOLINT is not zero, it will report rms closure phase and amplitude of the divided data set.

Various optional plots may be produced. They are: (1) The average and rms of the model subtracted data in bins of *uv*-plane radius. (2) The average and rms of the model divided - 1.0 data in bins of *uv*-plane radius. (3) The histogram of image pixels over a pixel value range selected by the user or the first of the model images. (4) The histogram of image pixels over a pixel range of plus/minus 5 times the rms of the first of the model images. (5) The real vs imaginary visibilities of the model subtracted data over a range of APARM(1) times the amplitude rms. The plot is a contour in logarithmic intervals of the 2-D histogram image controlled by APARM. Contours are drawn at 0.5, 1, 1.5, 2, 2.5, 3, etc in the log of the counts. (6) The real and imaginary gains minus (1,0) of the model divided data over a range of APARM(2) times the amplitude rms. Contours are again logarithmic and are displayed on the plot.

Closure phases and amplitudes are useful in evaluating the quality of a calibration source or a gain file produced by dividing a model into the visibility data with UVSUB, OOSUB, or EVAUV. Tasks CLPLT and CAPLT will plot closure phases and amplitudes, respectively, as functions of time for specified phase triangles and amplitude quadrangles. In 31DEC22, task EVACL may be used to determine the rms closure phase and amplitude as a function of IF (spectral window). A good point source (or gain) should have values near zero for both. Task CLOSE will plot the rms phase or amplitude closure as a function of spectral channel. This may be useful in determining which channels should be used in continuum observations and which should be omitted.

5.4.5 Experimental extension of multi-field self-calibration

High-quality, multi-field images often suffer from position-dependent calibration effects. At lower frequencies, the strongest sources may well lie outside the main portion of the primary beam and so be effected by different instrumental gains including pointing errors, differences in atmosphere or ionosphere, etc. An experimental RUN file PEELR compiles a procedure by that name. One "interfering" field at a time, it performs a self-cal to improve that facet. After a list of fields is processed, it restores the original multi-field model to the corrected residual *uv* data. See HELP PEELR for details. It seems almost magical, but it really does improve the final images. A slight increase in the image rms is the price one pays for removing larger, more systematic problems.

5.5 More editing of *uv* data

5.5.1 General remarks on, and tools for, editing

There are many programs which aid in the processing, display, and editing of *uv* data. Summaries of this software may be listed on your terminal with:

> ABOUT UV ↵ to list all *uv*-related software.
> ABOUT EDITING ↵ to list all editing software.
> ABOUT PLOT ↵ to list all plotting software.

and are also in Chapter 13 of this *CookBook*. Type

> DOCRT -1 ; EXPLAIN *taskname* ↵ to print information about task *taskname*.

to get more information about any of the tasks mentioned below. The discussion below assumes that you have deduced that there are suspect samples in your data set and that you want to remove them. Read § 4.3.5 before investing large amounts of time in editing even at this stage.

There are facilities in CALIB, RFLAG, FLAGR, CLIP, CORER, UVMLN, FLGIT, DEFLG, and SNFLG to flag *uv* data in *AIPS* based on deviations from specified norms. There is also the task UVFLG to flag and unflag by antenna-IF or by correlator. The task UVPLT plots various combinations of *uv* data; see § 6.3.1. The task WIPER makes a similar plot on the TV and allows you to wipe away offending data. The task UVFND is also recommended for printing out suspicious portions of the database; see § 6.2.1. Note that CLIP examines the data correlator by correlator, but UVFND normally converts the data to Stokes components (using the same criteria as UVMAP) before checking that the amplitudes and/or phases are in range. To examine the correlators individually, use STOKES 'CORR' in UVFND, or to flag the data based on their values after conversion to true Stokes use STOKES = 'IQUV' in CLIP. Task FINDR is a companion to FLAGR intended to assist you in determining what is normal within your data.

CLIP is also useful for flagging discrepant data (*e.g.*, due to interference or malfunctions) on the basis of their deviations from the visibility predicted by a set of Clean components. In 31DEC23, it even offers an option to flag on excessive differences betwen the two parallel-hand polarizations. Tasks OOSUB or UVSUB will subtract the Fourier transform of a set of Clean components from visibility data. You may then use UVPLT to display the residual *uv* data set and CLIP to flag abnormally high points. You may wish to be cautious, and run UVFND to display such points before running an automatic CLIP task — be especially careful not to CLIP away evidence for real extended structure near the center of your *uv* plane! Before re-imaging, you must copy, with TACOP, the flag table produced by CLIP to the data set used for imaging. Note that IMAGR's workfile is also a *uv* data set from which the current Clean component model has been subtracted. It may also be used with UVPLT to help you to diagnose problems. If you run CLIP on it, you will need to append the resulting FG table to one on the imaging data set; use TAPPE.

FFT is another useful tool for finding suspicious data. Transform your image back into the *(u,v)* plane by running FFT and then display the results on the TV. Use image read-back verbs like CURVALUE and IMPOS (§ 6.4.5) to find the u and v values for abnormally high cells. Then use UVFND with OPCODE 'UVBX' to print the data surrounding these cells and UVFLG to delete any bad data. This method is particularly effective when applied to residual images from Clean. (You can instruct IMAGR to put out a residual image by setting BMAJ <0. CCRES can also make a residual image from a single-field normal image.)

Task UFLAG makes a *uv*-plane grid (with pill-box convolution) of your visibility data selected by adverbs much like those used in IMAGR. It shows you the vector and scalar averaged grids in amplitude and the vector averaged grid in phase and offers a variety of editing options. You may choose to mark an entire cell as "bad" or look at the visibilities contributing to cells you select and choose to mark some of them (rather than all) as bad. There are even automatic routines to do this last operation over cells meeting criteria that

you set. The entire process is described in an *AIPS* Memo.[4]

There is a task called FLAGR which goes through a data set determining what are normal rmses and weights and then flagging those that deviate excessively including clipping all those that have amplitudes or weights outside specified normal ranges. FLAGR is intended for use eventually in pipeline data-reduction procedures, but at present should be considered experimental, but potentially very valuable. Before calibration, try it on your calibration sources using default values for most adverbs plus SOLINT set to 2.5 times the basic integration time and VECTOR set to 0. Task FINDR is a companion intended to determine what is normal in the data and then to print those values and return selected adverb values to AIPS for use by procedures.

There is a task for automatic flagging called RFLAG. Since RFI tends to be highly variable over time and/or frequency, RFLAG accumulates statistics about this variation which it plots along with recommended clip levels. Running it a second time, with DOPLOT ≤ 0, causes the clip levels to be applied to make a large flag table. This appears to be a very successful program with EVLA wide-band data (§ 4.3.13). Task REFLG may then be used to compress and extend the flag table.

TVFLG, SPFLG, FTFLG, WIPER, IBLED and EDITR are TV-based, interactive editors. TVFLG is most suitable for data sets with large numbers of baselines, *e.g.*, the VLA, but it can be used usefully for VLBI data experiments with 10 or more antennas. TVFLG allows you a global overview of your data and can display the data for all baselines simultaneously as a function of time. This task is documented extensively in A 2025 *AIPS* Memo[5] and § O.1.6 of the *CookBook*. SPFLG is a very useful task for data with a significant number of spectral channels. It is effective in examining data for frequency-dependent errors and interference and can be an effective data editor for interferometers with a small number of baselines; see the new Memo and § 10.2.2 and § 8.1. FTFLG is like SPFLG, but combines all baselines into one plot. This is a much faster way to look for global RFI, but it must flag all baselines with each flag generated. IBLED has a different philosophy; it plots one baseline at a time in a graphical rather than gray-scale (image) fashion. It is able to average data over time, spectral channels, and/or IFs to make a more manageable amount of data and to measure the "decorrelation index" which is a measure of how variable the phase is over the averaging intervals. The capability of averaging IFs and displaying decorrelation may be of special interest for VLBI data sets. Otherwise, IBLED has been replaced in 15APR98 by EDITR, which also uses the graphics planes rather than gray-scale images but which can plot multiple baselines to a chosen antenna and can display two data sets at the same time. This is obviously more useful for smaller arrays — *e.g.*, VLBI, MERLIN, and the Australia Telescope. This task is described below.

The editing task WIPER should be used with caution. It makes a plot like UVPLT of almost any parameter of a *uv* dataset against any other parameter. The plot is displayed on the TV and you may "wipe away" any points you do not like one point at a time or many at a time with a "fat brush." The task will be very useful for fields with a well-behaved visibility function seen with good signal-to-noise. It may also be useful with data sets from which a fairly good IMAGR model has been subtracted with UVSUB. It allows you to plot and edit any reasonable STOKES polarizations in one execution and displays which baselines enter into each plotted point.

The task FGCNT may be used to count the number of correlators flagged by a particular flag table. Many of the tasks above can generate very large flag tables and can, if the choice of parameters was not right, flag way too much data. This task provides a way to check on this.

5.5.2 Baseline-based *uv*-data editing — EDITR

EDITR is a very effective editing tool from the beginning of data analysis on data sets with modest numbers

[4]Greisen, E. W. 2025, "Editing on a *uv* grid in *AIPS*" AIPS Memo 121 revised, https://www.aips.nrao.edu/aipsdoc.html

[5]Greisen, E. W. 2025, "Interactive flagging in *AIPS*", *AIPS* Memo 127, https://www.aips.nrao.edu/aipsdoc.html

of antennæ. Since it can display two data sets at the same time for comparison purposes, EDITR may also be used to good purpose with larger data sets during the self-calibration and imaging stage. The visibility amplitude or phase or the amplitude of the visibility with a running vector average subtracted may be displayed. 31DEC25 added a coherence parameter defined as the amplitude of the scalar averaged visibility divided by the vector averaged visibility minus one. The data for the selected baseline are shown in an edit window at the bottom of the display. Optionally, a second observable (*e.g.*, phase) from the selected baseline is shown in the same color in a window directly above the edit window. This option is controlled by the DOTWO adverb. Data for 0 to 10 other baselines to the selected antenna may be displayed in a different color in windows above these. A second *uv* data set may also be displayed along with the first. These data are not used for editing but may help you to select the data to be deleted. A "normal" choice for the second data set would be the residuals after Cleaning or UVSUB. A menu-like control interface is available to select the data antenna and time range to be edited and to select various forms of editing. Instructions, explanations, informative messages, and the results of various functions appear in the standard *AIPS* message window. When prompted for information, such as an antenna number, type it into your normal AIPS input window (which is where the prompt message should have appeared).

EDITR is for editing continuum data from one or more IFs. Multiple spectral channels may be averaged on input with the vector average used for display and editing; multiple IFs are kept separate. The data may also be averaged over time as they are read into memory. This is useful for improved signal-to-noise, but will cause the data flags to be less selective in time. The program will allocate sufficient dynamic memory to hold the selected data. If you have a very large data set and a modest computer, it would probably improve efficiency to limit the time range and run the task more than once to cover all time ranges.

To run it, enter:

> TASK 'EDITR ; INP ↵	to select the task and review the inputs.
> INDI *n1* ; GETN *ctn1* ↵	to select the 'TB' sorted *uv* single- or multi-source data set.
> DOCAL FALSE ↵	to apply no calibration. The SN or CL table from previous calibrations can be applied.
> FLAGVER *fg1* ↵	to apply flag table *fg1* to the data on input.
> OUTFGVER 0 ↵	to write a new flag table containing version *fg1* and all new flag commands generated.
> SOLINT = Δt ↵	to have the data averaged over a time interval Δt minutes. If you do not want averaging, set this parameter to a small value; the default is $1/6000 = 0.01$ second. Editing times are recorded with an offset of SOLINT/2 which may cause confusion when no averaging was actually done.
> DETIME T ↵	to set the initial scan length estimate to T minutes (which can be changed later interactively) and to set the interval regarded as a break in the regular time sequence of the data. Setting this parameter suitably helps the program do a better display, but its exact value is not critical.
> CLR2NAME ↵	to display only one data set.
> DOTWO TRUE ↵	to display a second observable from the main baseline.
> CROWDED 2 ↵	to allow all IFs and all polarizations to be displayed and edited at one time, with the 2 speeding the display.
> DO3COL TRUE ↵	to use color to separate IFs and polarizations when more than one is displayed and edited at one time.
> INP ↵	to review the other parameters, which we assume here to be set to their null values.
> GO ↵	to run the task.

You can average the data over spectral channels (the default will average all channels present). IFs are

edited separately; the default will include all IFs after which you can choose the one to edit interactively.

Since the display used by EDITR is very similar to the one used by EDITA displayed in § 4.3.11, we do not include a figure here; see Figure 4.2. The upper left corner of the display is reserved for displays of the selected data sample during editing while the bottom left corner is used for status information including flagging options. Menus, discussed below, appear down the left and right sides of the screen. The data are displayed in a stack of plots in the center of the screen. At the bottom are the data from the selected baseline in the primary observable; then the data from the primary baseline in a second observable (if DOTWO is true), and finally the data in the primary observable from 0–10 other baselines to the primary antenna. Data which have been flagged are shown in a different color. The data are plotted on a linear axis vertically, while the horizontal axis is monotonic but irregular in time. Tick marks are plotted at integer hours and the time interval of the edit area is indicated by times at the left and right ends of the axis. The time range displayed in all plots may be selected interactively and editing may therefore be done in crowded full time-range plots or in well separated short time-range plots. Surrounding the plot are various annotations describing the data plotted and the status of the various flags which control which data will be deleted on the next flagging command. If a second data set was specified, then data from that file are displayed in a different color in the same plot areas used for the primary data set.

The interactive session is driven by a menu which is displayed on the same screen as the data. Move the cursor to the desired operation (noting that the currently selected one is highlighted in a different color on many TVs) and press button A, B, or C to select the operation. Press button D for a short explanation of the selected operation. The right-hand column contains options to select which data are displayed and to select which data are flagged on the next flag command. The menus are changed to adapt to the input data in order to avoid, for example, offering options to select IF in a one-IF data set. The left-hand column contains 7 interactive modes for editing the data plus options to set the display ranges and scan averaging length, to turn on error bars in plotting samples, to review, alter, and re-apply the existing flag commands, to defer or force a TV display, to switch to entering commands from the keyboard instead of the menu, and to exit with or without applying the current flag commands.

The right-hand menu can contain

NEXT POL/IF	To switch to viewing the next correlator, switching to the other polarization and, if needed, incrementing the IF.
SWITCH POLARIZ	To switch to viewing and editing the other polarization, cycles through both if CROWDED was true.
SWITCH ALL POL	To switch functions from applying to one polarization to applying to both polarizations or vice versa.
ENTER IF	To select which IF is viewed and edited. When CROWDED is true, zero means all.
SWITCH ALL IF	To switch functions from applying to one IF, to applying to a range of IFs, to applying to all IFs.
SWITCH ALL TIME	To switch FLAG ABOVE and FLAG BELOW between all times and the time range of the frame.
ROTATE ALL ANT	To rotate functions from applying to (a) one baseline, (b) all baselines to the main antenna, and (c) all baselines.
SWITCH ALL SOURC	To switch between flagging only the current source and flagging all sources.
ENTER ANTENNA	To select the main antenna, baselines to which are displayed on the screen.
ENTER OTHER ANT	To select up to 11 other antennas to define the baselines to be displayed; enter 11 numbers, 0's are then ignored (to plot 5 enter the 5 plus 6 0's). The first one is used for the edit area.

NEXT BASELINE	To advance the list of other antennas, selecting the next one for the edit area.
NEXT ANTENNA	To select a new main antenna, one higher than the current main antenna. The "others" will also be adjusted if appropriate.
PLOT ALL TIMES	To display all data for the selected baselines.
SELECT FRAME	To select a window into the current data interactively.
NEXT FRAME	To select the next time range window of the same size as the current frame.
PREVIOUS FRAME	To select the previous time range window of the same size as the current frame.
SHOW AMPLITUDE	To display and edit amplitudes.
SHOW PHASE	To display and edit phases.
SHOW DIFF AMPL	To display and edit the amplitudes of the vector difference between the sample and its running mean.
SHOW COHERENCE	To display and edit the coherence parameter which is the ratio of the scalar average amplitude divided by the vector average amplitude - 1 over the "scan" length.
SHOW ALSO AMPL	To display amplitudes of the edit baseline for reference with the phase, difference amplitude, or coherence parameter edit window.
SHOW ALSO PHASE	To display phases of the edit baseline for reference with the amplitude, difference amplitude, or coherence parameter edit window.
SHOW ALSO DAMP	To display difference amplitudes of the edit baseline for reference with the phase, amplitude, or coherence parameter edit window.
SHOW ALSO COHE	To display the coherence parameter of the edit baseline for reference with the phase, amplitude, or difference amplitude edit window.
TV ZOOM	To alter the display zoom used while in the flag functions.
OFF ZOOM	To turn off any zooming.
2ND UV OFF	To disable the display of the $2^{\rm nd}$ *uv* data set.
2ND UV ON	To enable the display of the $2^{\rm nd}$ *uv* data set.

The data displayed are of a single polarization, single IF, and 1–11 baselines to a single antenna. If CROWDED is 1 or 2, then you may also choose to display and edit both polarizations and/or all IFs at the same time and input adverb DO3COL determines if polarizations and IFs are shown in different colors. The NEXT CORRELATOR cycles through all polarizations and IFs, show one at a time. The SWITCH POLARIZATION option switches the displayed polarization, the ENTER IF option prompts you if necessary for a new IF number, the ENTER ANTENNA option prompts you for a new primary antenna number, and the ENTER OTHER ANT prompts you for up to 11 other antenna numbers to select the main editing baseline and up to 10 secondary baselines to the primary antenna. (Note that you have to type in 11 numbers, but zeros are then ignored.) A flag command can apply to one or both polarizations and to one, some, or all IFs. It can apply to one baseline, to all baselines to the primary antenna, or to all baselines. The FLAG ABOVE and FLAG BELOW commands can apply only to the time range displayed in the data "frame" or they can apply to the full time range in the data set. The SWITCH ALL POL, SWITCH ALL IF, ROTATE ALL ANT and SWITCH ALL TIME options control these choices and the current state of these switches is displayed at the lower left of the TV screen. The task is able to zoom the display during interactive editing operations if you should need magnification to see what you are doing. The TV ZOOM and OFF ZOOM options let you control this. In larger data sets, however,

a more useful display is obtained by interactively selecting a narrower time range with the SELECT FRAME option. To step forward and back through the frames, use the NEXT FRAME and PREVIOUS FRAME options, respectively. To display *uv* data amplitude, select SHOW AMPLITUDE and to display *uv* data phase, select SHOW PHASE. You may also display the difference between the current data sample and a running vector average of the data centered on the current sample and extending no more than plus or minus the "scan length" divided by two. To display the amplitude of the vector difference, select SHOW DIFF AMPL. Such displays are particularly sensitive to short-term problems while ignoring longer-term changes due to source structure. To display and edit the coherence parameter select SHOW COHERENCE. Note that a short-term disturbance in the visibility may cause an apparently longer-term disturbance in this parameter because of the time averaging. Since it takes time to compute things for, and display, the second data set, you may wish to turn it off part of the time. The 2ND UV OFF and 2ND UV ON options control this choice.

The left-hand menu can contain

FLAG TIME	To delete one time at a time.
FLAG TIME RANGE	To delete one or more time ranges.
FLAG BELOW	To delete all displayed times with data below a cutoff value.
FLAG ABOVE	To delete all displayed times with data above a cutoff value.
FLAG AREA	To delete one or more areas in the data-value *vs* time plane.
FLAG POINT	To delete one sample at a time using both horizontal and vertical cursor position.
FLAG QUICKLY	To delete samples using only mouse clicks
CLIP EVERYTHING	To delete all displayed times with data above a cutoff value. All baselines are flagged in the IFs and polarizations currently displayed in the edit window at whatever level was set.
ENTER AMPL RNG	To select the display range for amplitude plots. Use $0-1$ for zero to maximum, 00 for minimum to maximum.
ENTER PHASE RNG	To select the display range for phase plots.
ENTER DAMP RNG	To select the display range for plots of the amplitude of the visibility minus a running vector average visibility.
ENTER COHE RNG	To select the display range for plots of the coherence parameter.
PLOT ERROR BARS	To plot error bars based on data weights.
SET SCAN LENGTH	To set the averaging time used to determine the running average in seconds.
LIST FLAGS	To list all flags now in the Flag Command table.
UNDO FLAGS	To undo one of the flag operations in the FC table
REDO FLAGS	To reapply all remaining flags after one or more have been undone
SET REASON	To set the "reason" string to be put in the *uv*-data flag table.
USE EXPERT MODE	To control the task from the keyboard instead of the menu.
HOLD TV LOAD	To stop updating the TV display with every change of parameter; change several, then select
DO TV LOAD	To update the TV display now and with each change of display parameter.
REPLOT	To do the current plot over again, recomputing the differences from the running mean if appropriate.
CHAR MULT	To change the character size on the TV display.
EXIT	To exit EDITR, moving the FC table to a *uv*-data FG table.
ABORT	To exit EDITR, deleting the FC table.

The first eight items select interactive flagging modes to delete all selected data at a single time, over a range of times, over all values below or above a specified value, or within a range of times and values (respectively). When one of these options is invoked, the screen zooms (if set to do so), a line or box appears in the editing window, and a display of the sample (source, time, value) under the cursor appears at the upper left. Follow the instructions in the message window to select and edit data. Note that this is a very good way to look at your data values even if you do not want to delete anything. The FLAG QUICKLY method is very efficient, but it requires caution in its use. Whenever the left mouse button is depressed, the sample closest to the cursor position is flagged. The CLIP EVERYTHING option is particularly powerful and should be used with caution. The next four options set the range of amplitudes, phases, difference amplitudes, and coherence displayed. These default to the full range in the data (separately and differently for each baseline) and can be set back to default by entering 0 0. The SET SCAN LENGTH option prompts you for a "scan" length in seconds used as the averaging interval for computing the running mean used in the difference displays. A longer scan length takes longer to compute, but is likely to be less noisy and more meaningful as an editing tool. If you are not using the difference display, set the scan length to a short interval. The running mean is not carried between sources and, as a result, is not normally carried across actual scan boundaries.

When you execute a flagging option, one or more lines are written to a flag command (FC) table attached to the input data set. If EDITR dies abnormally, this FC table can even be used in a later session. To list all of the flagging commands now in the table select the LIST FLAGS option. If you decide that you no longer want one of these flags, select UNDO FLAGS and enter the number (from LIST FLAGS) of the undesirable flag command. More than one flag command may apply to the same datum. After undoing flags, it is such a good idea to redo them that REDO FLAGS and tt REPLOT are now done automatically (in 31DEC20). This is to make sure that everything is consistent. When the flag commands in the FC table are entered into a normal flag table, a 24-character "reason" is attached which is both descriptive and can even be used in UVFLG when removing entries in the FG table. The SET REASON command prompts you for the reason to be attached to subsequent flag commands. The default reason is the task name, time and date. Normally, EDITR updates the display whenever anything is changed. If you are about to change more than one display parameter (*i.e.*, polarization, IF, antenna, other antennæ, frame) before doing more editing, select HOLD TV LOAD to defer the display update until you select the DO TV LOAD option. If the display appears not to be current, select the REPLOT option. Finally, you may exit the program with the EXIT or ABORT options. The former applies your editing to a flag (FG) table attached to the input data set, while the latter discards any editing commands you may have generated.

Note that value-dependent flagging (FLAG BELOW, FLAG ABOVE, FLAG AREA, and CLIP EVERYTHING) use the values currently plotted to make a list of value-independent flag commands, namely a single time for the specified antennæ, IFs, polarizations, etc. When a value-dependent flag operation is undone with UNDO FLAGS or redone with REDO FLAGS, it is these value-independent flags which are undone or redone. You may have to undo more commands and then repeat flag commands to get the results you could have gotten by doing the now desired value-dependent command in the first place. You need also to be careful with the ROTATE ALL ANT setting with these value-dependent commands. If one baseline is set, then the commands only apply to the current baseline. If one antenna is set, the commands apply to all baselines to the current main antenna, while if all antennas is set, the commands apply to all baselines. The first three set an area or a clip level, below or above which data are deleted, based on the value of the observable in the baseline in the main edit window. The values of the data from other baselines and antennas are ignored. CLIP EVERYTHING ignores the setting of the ROTATE ALL ANT parameter and applies the clip based solely on the data value in each baseline.

Be careful when choosing EXIT versus ABORT. The former applies the flag commands to a flag table attached to the input uv data, the latter causes the flag commands to disappear without a trace. After EXIT, of course, one may use, edit, or ignore the output flag (FG) table. For single-source files, it may be necessary to run SPLIT to apply the FG table to the data since only some tasks know how to apply FG tables (those with FLAGVER as an adverb).

The colors used by EDITR are those of the various graphics planes when it begins to run. You may change them with the AIPS verb GWRITE to more desirable colors. The planes are:

```
Plane     Default RGB      Use
  1     1.00  1.00  0.00   Main editing and secondary windows
  2     0.06  1.00  0.00   Comparison baseline data windows
  3     1.00  0.67  1.00   Menu highlight
  4     0.00  1.00  1.00   Edit and frame window boundaries
  5     1.00  0.18  0.18   Flagged data in all windows
  6     0.60  0.60  1.00   Menu foreground
  7     1.00  0.80  0.40   Second uv data set if present
```

You may wish to change the colors to ones that you can see better.

5.6 Imaging with OBIT, Complex imaging

AIPS contains "verbs" that access programs in the OBIT software package written by Bill Cotton (bcotton@nrao.edu). If that package is installed on your computer, which is easy for Linux and not too hard for Mac systems, then you may use these programs. Two of the verbs are used to translate EVLA data into *AIPS* format; see § 4.1.1. The other four verbs provide increasingly elaborate access to OBIT's imager program. They are called OBITMAP, OBITIMAG, OBITSCAL, and OBITPEEL. The first images a field of view in a single scale, the second allows many more imaging options including multi-scale, the third adds self calibration, and the fourth allows the self-calibration to be direction dependent. The *AIPS* group cannot take responsibility for the OBIT installation and programs, but much of the cleverness in *AIPS* was due to Bill and his later work deserves consideration. If nothing else, the tasks will run in a multi-threaded fashion which *AIPS* does not.

AIPS has a complex imaging option impletnted with the RUN file and procedure called CXIMAGR. It is a sequence of tasks UVIMG, FFT, and CXCLN. It is interesting, but the normal imaging described in this chapter is likely to be more satisfying.

5.7 Additional recipes

5.7.1 Cranberry banana bread

1. In a large saucepan, bring 2 cups **sugar** and 1 cup **water** to a boil, stirring to dissolve the sugar. Add 4 cups fresh **cranberries** and simmer over low heat for 10 minutes or until berries pop open. Cool. Drain the berries, reserving the juice and measuring 1 cup of berries for use in the bread.
2. Sift together 1 3/4 cup **flour**, 1/2 teaspoon **salt**, 2 teaspoon **baking powder** and 1/4 teaspoon **baking soda**.
3. In a large bowl, combine 2/3 cup **sugar**, 1/3 cup melted **butter**, 2 beaten **eggs**, 1/2 cup chopped **walnuts**, 1 cup mashed **banana**, and 1 cup cooked berries.
4. Add the flour mixture to the berry mixture, stirring until blended. Pour the mixture into a greased and lightly floured 9 x 5 x 3-inch loaf pan. Bake in a preheated, 350° F oven for 1 hour or until a toothpick inserted in the center comes out clean.

5. For a topping (optional), combine 1/4 cup **cranberry juice** from cooked berries, 2 tablespoons **sugar** and 2 tablespoons **Grand Marnier** in a small saucepan and stir over low heat until heated through. Poke a few holes in the baked loaf and pour on the topping.
6. Cool 10 minutes in the pan. Turn the loaf out on a rack and cool completely. Wrap in foil and store one day before slicing.

Thanks to Tim D. Culey, Baton Rouge, La. (`tsculey@bigfoot.com`).

6 Displaying Your Data

This chapter is concerned with the ways in which you may display your data. There are a number of tasks for generating "plot files" which contain graphics commands for the making of various displays of your *uv* and image data. All of these now offer a "preview" option to draw the plot directly on the $\mathcal{AIPS}$ TV, rather than putting the commands into a file. Once the files are created, a variety of tasks may be used to translate them into displays on various devices, such as the $\mathcal{AIPS}$ TV and graphics windows and PostScript printers. There are also verbs to display and manipulate your images on the $\mathcal{AIPS}$ TV and a single task TVCPS to capture that display, if desired, into a PostScript file for printing, recording on film, or even including in your scientific papers.

Several indices of the $\mathcal{AIPS}$ software are relevant to this discussion. To generate current lists of $\mathcal{AIPS}$ functions on your workstation window (or terminal) use ABOUT HARDCOPY ↵, ABOUT INTERACT ↵, ABOUT PLOT ↵, ABOUT TV-APPL ↵, and ABOUT TV ↵. Recent versions of these indices are reproduced in Chapter 13 of this *CookBook*.

6.1 Getting data into your $\mathcal{AIPS}$ catalog

By the time you reach this chapter, most of your data will probably already be loaded into your $\mathcal{AIPS}$ catalog either by reading an external tape or disk or by being generated by some $\mathcal{AIPS}$ task. Visibility data which are not presently on disk may be read by the $\mathcal{AIPS}$ tasks FILLM, UVLOD and FITLD; see § 4.1 and § 5.1 for details. Images that are generated by other imaging systems, (*e.g.*, images from non-NRAO radio telescopes or non-radio images) can be transported to $\mathcal{AIPS}$ by writing them out of the other imaging system on tape or disk in the standard FITS format. The tasks IMLOD and FITLD can then be used to read them into $\mathcal{AIPS}$. These tasks are also used to read images saved with FITTP and FITAB from previous $\mathcal{AIPS}$ sessions.

6.1.1 IMLOD and FITLD from FITS-disk

FITLD and IMLOD can read FITS-format images from external disk files or magnetic tape into your $\mathcal{AIPS}$ catalog. The disk-file option is indicated by setting the adverb DATAIN to a non-blank value. Disk image files must be read in only one at a time per execution of IMLOD, but FITLD can read more than one FITS-disk file if the file names are identical except for sequential post-pended numbers beginning with 1. DATAIN is a string of up to 48 characters that must completely specify the disk, directory, and name of the input disk file to your computer's operating system. See § 3.10.3 for a discussion of FITS-disk files.

One "feature" of $\mathcal{AIPS}$ complicates this otherwise straightforward disk analog of FITS tape reading. AIPS translates all of your alphabetic inputs to upper case (this was demanded by users who otherwise became confused between upper and lower cases).[1] So if your computer distinguishes upper and lower cases for disk, directory, or file names, you probably should do two things to prepare for this before running $\mathcal{AIPS}$. First, restrict your external disk file names to upper-case characters and numbers. Second, set an upper-case "environment variable" or "logical" to point to the disk and directory where your FITS-disk images are stored before you run AIPS. You may need help from your System Manager when doing this for the first time. A common strategy on UNIX machines is to create an upper-case logical name after logging in but before starting up AIPS:

% setenv MYLOGICAL *myarea* ↵ if using C-shell, or

[1] If you omit the close quote on the character string, it is not converted to upper case, allowing you to circumvent this $\mathcal{AIPS}$ limitation. The string without a close quote must, of course, be the last thing on the line.

`$ export MYLOGICAL=`*myarea* ↵ if using korn, bourne, or bash shells,

where MYLOGICAL is an all-upper-case string of your choice and *myarea* is the full path name of the disk directory that contains your FITS-disk data. *AIPS* usually provides a public disk area known as FITS which you may use.

Then, once inside AIPS, tell FITLD or IMLOD:

> DATAIN 'MYLOGICAL:IMAGE.DAT' ↵ to read in the FITS-disk file *myarea*:IMAGE.DAT.

To check that the file name is correct, type:

> TPHEAD ↵

Your terminal will then list information about the image header of the disk file.

> OUTDI *n* ↵ to specify writing the image to your *AIPS* catalog on disk *n*.

> OUTNAME '*your-chosen-name*' ↵ to specify the output disk file name in *AIPS*; the default is the image name on tape if FITTP was used to write the image to tape.

The string *your-chosen-name* can be any ($\leq$ 12-character) title that you want to use as the image name within *AIPS* and should be specified for images from other image-processing software systems. FITLD also allows you to specify the 6-character image "class" parameter. Use OUTCLASS '*abcdef*' ↵, if you wish to change the class from that in your input file as the image is read or if the image comes from a "foreign" system.

> OUTS -1 ↵ to keep the sequence number the same as that in the file; the default is the highest unique number for images with this name and class in your current *AIPS* catalog.

> NCOUNT *m* ↵ in FITLD only, to load *m* images consecutively starting with the file specified by DATAIN with a 1 post pended. The other files must be the same name but with sequential numbers post pended. If you use this option, do *not* specify the OUTNAME unless you want the same name for all the new images in your catalog.

> GO FITLD ↵ or IMLOD, to run the task.

If OUTNAME is left unspecified, it defaults to the "name" of the image read from the FITS header — either the name previously used in earlier image processing or the source name. If OUTCLASS is unspecified, it defaults to the Class previously used in earlier image processing or to a compound name (*e.g.*, IMAP, IBEM, QMAP, ICLN) which attempts to describe the image. These defaults are frequently good ones when you are loading multiple consecutive images with NCOUNT > 0. You may of course change the *AIPS* image and class names later by using RENAME (see § 3.3.3 of this *CookBook*).

IMLOD and FITLD can also read images from magnetic tape; see the instructions in § 3.9. Use the verbs MOUNT to mount the tape, AVFILE to advance the tape to the desired file, and TPHEAD to check that the tape is correctly positioned. Set NFILES=0, NCOUNT to the number of consecutive tape files you wish to load, and the say GO to either FITLD or IMLOD.

6.1.2 Image coordinates

Images of the celestial sphere must be rendered with some sort of coordinate "projection." *AIPS* pioneered the handling of such coordinates, supporting true projective coordinates called SIN (orthographic), TAN (gnomic), ARC (zenithal equidistant), and STG (stereographic) as well as a special version of orthographic suited for East-West interferometers (NCP). *AIPS* also supports modern[2] interpretations of GLS (SFL Sanson-Flamsteed), MER (Mercator), AIT (Hammer-Aitoff), CAR (Plate carrée). MOL (Molweide), and PAR

[2]Calabretta, M. R. and Greisen, E. W. 2002, "Representations of celestial coordinates in FITS," *Astr. & Ap.*, 395, 1077.

(parabolic) "projections." These are normally used to represent large sections of the celestial sphere. Users should be warned that reference pixel values other than zero lead to *oblique* projections.

Fits to the coordinates on optical images often lead to a modest amount of skew in the image. This is represented in the PC*i_j* or CD*i_j* FITS header cards. If these cards contain any significant skew, IMLOD and FITLD will tell you about it. In such cases, run DSKEW on the images after they have been loaded into *AIPS*. This is the only task that understands coordinate skew and it will re-grid the images for use by all other *AIPS* tasks.

6.2 Printer displays of your data

The most old fashioned way to look at your data — and the most exact — is simply to print it out and read the numbers. *AIPS* provides a variety of tasks and verbs to print visibility data, image data, tabular data, and miscellaneous other information. All of these tasks and verbs allow you to specify where the printed output goes using two adverbs, DOCRT and OUTPRINT. If DOCRT ≤ 0 and OUTPRINT is blank, then the output is placed in a temporary file and queued to the printer you selected when starting AIPS; see § 2.2.3. If printing directly to the printer is requested, most tasks and verbs count the number of lines of print and request permission to proceed if the count is large. (Type PRINTER n ↵ to change the line printer selection to that numbered n; type PRINTER 999 ↵ to see the devices available to you and their assigned numbers.) If DOCRT ≤ 0, a non-blank OUTPRINT specifies a text file into which the output is to be written; see § 3.10.1. The current output is appended to the file if it already exists. Thus, you can combine a number of printed outputs for later editing and/or printing. When DOCRT = -1, the output print file will contain full paging commands and headers. To suppress some of this, use DOCRT = -2 or to suppress almost all of it, use DOCRT = -3. This last is especially helpful when writing programs to read the text file. If DOCRT > 0, the output is directed to your workstation window or terminal. All printer verbs and tasks are able to respond to both the width and height of your workstation window. Set DOCRT = 1 to use the current width; set DOCRT $= n \geq 72$, to use n as the width of the display window. Since most print tasks display more information on wider windows, we recommend widening your window to 132 characters and specifying DOCRT = 1. The print routines will pause whenever the screen is full and offer you the choice of continuing or quitting. Thus, you can start what might be a very long print job, find out what you wanted to know after a few screens full, and quit without using up any trees.

6.2.1 Printing your visibility data

Before beginning calibration, it is a very good idea to make a -summary list of the contents of your data set. LISTR with OPTYPE = 'SCAN' will list the contents of each scan in the data set. LISUN produces the same output with the addition of the angle between each scan and the Sun. It also honors TIMERANG and SOURCES unlike LISTR in its SCAN mode. DTSUM also produces a listing summarizing the data set in either a condensed or full form.

The most basic display of your visibility data is provided by PRTUV which lists selected correlators in the order they occur in the data set:

> TASK 'PRTUV' ; INP ↵	to review the inputs.
> INDI n ; GETN ctn ↵	to select the disk and data set to print.
> CHANNEL c ; BIF 1 ↵	to print starting with channel c from IF 1.
> BPRINT m; XINC i ↵	to print every i^{th} visibility starting with the m^{th} visibility in the data set.
> DOCRT 1 ; GO ↵	to run the task with display on the terminal.

When you have seen enough, enter q ↵ or Q ↵ at the page-full prompt.

You may limit the sources, range of projected baselines, and times displayed and may select only one baseline or one antenna. PRTUV does not apply calibration or flagging tables. To get a similar display with all "standard" calibration, flagging, and data selection (optionally) applied, use the task UVPRT. LISTR also uses all of the calibration options to list the data in simple lists or in a display showing all the baselines at each time in a matrix form. SHOUV also lists calibrated visibility data with options to average all channels in each IF and to display closure rather than observed phases. ANBPL converts baseline-based amplitudes, phases, or weights into antenna-based values and prints and/or plots them. The display of antenna-based weights before and after amplitude self-calibration is a particularly useful tool for spotting calibration/instrumental problems.

There are a number of tasks used to diagnose possible problems in your data and to print information about them. UVFND examines a data set for excess fluxes, excess apparent V-polarization, or simply any data with a specified fringe spacing and position angle or a specified range in u and v. As it does this, it also checks for bad antenna numbers, bad times, and (optionally) bad data weights. CORER examines a data set for excessive mean values and rms in each correlator (after applying calibration and flagging and then subtracting a point source at the origin). RFI examines the rms fluctuations in the real and imaginary visibilities of each correlator looking for (and reporting) periods of apparent RF interference. UVDIF directly compares two data sets reporting any excess differences. It is useful for determining whether your latest operations (flagging, self-cal) have made a significant (or any) difference.

6.2.2 Printing your image data

The most basic display of an image is a print out of the numbers it contains. Such a display is provided by PRTIM:

> TASK 'PRTIM' ; INP ↵	to review the inputs.
> INDI *n* ; GETN *ctn* ↵	to select the disk and image to print.
> NDIG 3 ↵	to use 3 digits, printing numbers between -99 and 999 with appropriate power of 10 scaling.
> FACTOR 10 ↵	to raise the default scaling by a factor of 10, overflowing regions of high values to see low valued regions better.
> BLC 0 ; TRC 0 ↵	to see the whole image.
> XINC 2 ; YINC 2 ↵	to see every other column and every other row.
> DOCRT FALSE ; GO ↵	to print the image on the selected printer.

Other imaging tasks which can use the printer are BLSUM and ISPEC, which compute and print spectra by summing over regions of each plane in a data cube (see § 8.6), and IMFIT, JMFIT, and SAD, which fit one or more Gaussians to an image (see § 7.5). IMTXT writes an ASCII-formatted file containing an image.

6.2.3 Printing your table data

If you have any doubts about the contents of tables in *AIPS*, it is best to resolve them by looking at the contents of the tables involved. PRTAB is a very general task which will print the contents of any *AIPS* table file. For example, to print flag table version 1:

> TASK 'PRTAB' ; INP ↵	to review the inputs.
> INDI *n* ; GETN *ctn* ↵	to select the disk and catalog entry to print.
> INEXT 'FG' ; INVERS 1 ↵	to select flag table version 1.

> BPRINT 0 ; EPRINT 0 ; XINC 1 ↵ to print everything.
> DOHMS TRUE ↵ to print times in sexagesimal notation.
> DOCRT 1 ; GO to print the flag table on the terminal.

When you have seen enough, enter q ↵ or Q ↵ at the page-full prompt. For a table with a significant number of columns, PRTAB shows all rows for the first columns and then loops for the next set of columns. To see all columns for some rows, set a low EPRINT value or be very patient. Enter a list of column numbers in BOX to see only some of the columns. NCOUNT, BDROP and EDROP control which values are displayed in those columns having more than 1 value per row. Adverb RPARM lets you limit the display to rows having specific column values within specified ranges, while NDIG controls the format and accuracy used to display floating-point columns.

Some of the tables have specialized printing programs. These include PRTAN for antenna tables, PRTCC for Clean component tables, SLPRT for slice files, and LISTR with OPTYPE = 'GAIN' for calibration, solution, and system temperature tables. The verb EXTLIST will list information about various extension files, particularly slice and plot files (see below), which may be printed with PRTMSG. OFMLIST is a verb to print the contents of an *AIPS* TV color table. Finally, task TBDIF will compare columns of two tables and print information about their differences. PRTSY and SYPRT offer differeing views od the contents of EVLA SysPower tables.

6.2.4 Printing miscellaneous information

There is a variety of miscellaneous information which may also be sent to the printer in the same way. Verb PRTMSG prints selected contents of the *AIPS* message file; see § 3.2. Verb PRTHI prints selected lines from a history file; see § 3.4. Pseudo-verb ABOUT prints lists of *AIPS* symbols by category while pseudo-verbs HELP and EXPLAIN print information about a selected symbol; see § 3.8. Task PRTTP prints the contents of magnetic tape volumes and pseudo-tape disk files; see § 5.1.1 and § 3.9.4. Task PRTAC, which may also be run in a stand-alone mode, prints information selected from the *AIPS* accounting file.

Task TXPL will attempt to represent an *AIPS* plot file (see below) on the printer. This will not work well for complicated plots, but, for simple plots, it may be the only way someone running over a slow telephone line can see his/her data in plot form.

6.3 Plotting your data

The basic concept in *AIPS*' plotting is to use some task to create and write a device-independent plot file as a PL extension file to a cataloged image or visibility data set and then to use some device-dependent task to interpret that file for the desired output device. Plot files are not overwritten by subsequent plot tasks. Instead they make new plot files with higher "version" numbers. The device-dependent tasks include TVPL (*AIPS* TV devices including XAS), TKPL (Tektronix graphics devices including *AIPS*' TEKSRV server), TXPL (line printers), and LWPLA (PostScript printer/plotters). Tasks called PRTPL, QMSPL, and CANPL support antique Versatec, QMS, and Canon printer/plotters. To plot on a PostScript printer/plotter:

> TASK 'LWPLA' ; INP ↵ to review the inputs.
> INDI *n* ; GETN *ctn* ↵ to select the disk and catalog entry to print.
> PLVER *m* ; INVERS 0 ↵ to plot the $m^{\rm th}$ plot file only.
> OUTFILE ' ' ; GO ↵ to do the plot immediately.

LWPLA offers the option to save the file for later plotting or inclusion as encapsulated PostScript in other documents. It also has options to control scaling, output paper size, width and darkness of lines, and transformation of grey-scale intensities. It can write more than one plot file at a time and can append

new plots to existing output files. Multi-plot files are not "encapsulated" but may be printed and viewed with tools such as `gv` or `ghostview`. Note that PostScript files are text files and *AIPS* writes particularly simple PostScript so that it can be modified by the users. See HELP POSTSCRIPT for suggestions including information on deleting and adding labels and arrows and on converting the PostScript to other formats like jpg without loss of resolution.

LWPLA and all plot tasks offer some "coloring" options. These are illustrated in the color pages at the end of this chapter. The grey-scale plotting tasks, including GREYS, PCNTR, and KNTR, can now enhance the grey-scales with a transfer function and then pseudo-color them with a color table. See § 6.4.3 for a short discussion of "output-function memory" tables which may be read into the above tasks or to LWPLA using the adverb OFMFILE. Lines plotted on top of grey scales (*e.g.*, contours, polarization vectors, stars) may be "dark" when the grey scale intensity is high. LWPLA may be instructed to plot these as bright if adverb DODARK is false. All plot programs can draw lines of different types in both bright and dark forms. In LWPLA, if DOCOLOR is true, the array adverb PLCOLORS(i, j) controls the red, green, and blue colors ($i = 1, 2, 3$, resp.) of line types $j = 1 - 10$. The normal meanings of these types are:

1. Bright labeling, tick marks, surrounding lines
2. Bright lines, usually contours or model curves
3. Bright lines, usually polarization vectors
4. Bright lines, usually symbols such as stars, visibility samples
5. Dark labeling text inside plot area
6. Dark lines, usually contours
7. Dark lines, usually polarization vectors
8. Dark lines, usually symbols such as stars
9. Bright labeling outside the main plot area, e.g. titles, tick values and types, documentation
10. Background for the full plot

Some plot tasks have the ability to control the colors of their line drawing independent of these line types. These colors may be controlled only when making the plot file with the particular task. PCNTR and KNTR have acquired the ability to color each contour level under control of the adverb RGBLEVS. Using system RUN file SETRGBL, procedures CIRCLEVS, RAINLEVS, FLAMLEVS, and STEPLEVS are available to help you set the values of RGBLEVS. Examples are shown on the color pages at the end of this chapter. With LWPLA and these options one may prepare extremely effective displays — or hopelessly bad ones — for use in talks and, since the prices have become reasonable, even in journals. Note that most journals want color images in CMYK (cyan-magenta-yellow-black) rather than RGB; use DPARM(9) = 1 in LWPLA to get PostScript files with this color convention. Note that these two color representations usually require different "gamma" corrections; the adverb RGBGAMMA allows this control in LWPLA.

All *AIPS* plot tasks now offer a "preview" option. If you set DOTV = TRUE when running any plot task, then the plot appears immediately on the *AIPS* TV display and no plot file is generated. This option allows you to make sure that the parameters of the plot are reasonable and lets you avoid making files and wasting paper for quick-look plots. Additional options allow you to control which graphics channel is used for the line drawing (GRCHAN) and to select pixel scaling of the plot at your specified location on the TV screen (TVCORN). These two options allow you to view more than one plot at a time on the TV, usually for purposes of comparison. Each graphics channel on the TV has a different color and a complementary color is used when two or more channels are on at the same point. This allows for a fairly detailed and effective comparison of plots, all of which may be captured with task TVCPS (see below). Be aware that most tasks

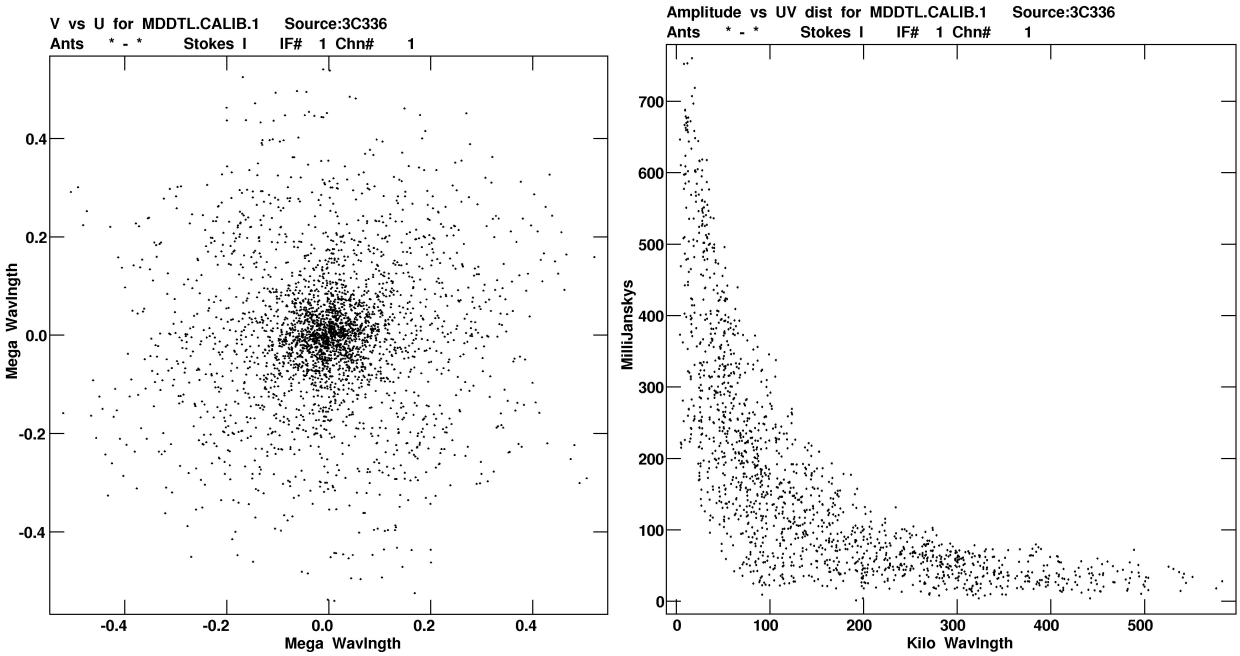


Figure 6.1: UVPLT displays of where the data were observed in the *uv* plane (left) and of the visibility amplitudes as a function of baseline length (right). The data on 3C336 were provided by Alan Bridle from observations made with the VLA on 6 December 1987.

now interpret GRCHAN = 0 as an instruction to use graphics channels 1 through 4 for line types 1 through 4 and graphics channel 8 for dark vectors. The comparison function is only achieved by specifying GRCHAN; see HELP GRCHAN for more information. Tasks that produce multiple plot files pause for 30 seconds at the end of each plot when DOTV = TRUE. This allows you to stop the task (TV button D), hurry it along (TV buttons B or C), or make it pause indefinitely (TV button A) until another TV button is pressed.

You can review the parameters of the plot files associated with a given image or visibility data set by typing:

> INDI *n* ; GETN *ctn* CR	to select the disk and catalog entry to print.
> INEXT 'PL' ; EXTLIST CR	to list summaries of the plot file contents.
> PLVER *m* ; PLGET CR	to recover all adverb values used when making the specified plot file.

Plot files (and other "extension files") are automatically deleted when an image is deleted by ZAP. However, large plot files should be deleted as soon as they are no longer needed:

> INP EXTDEST CR	to review the inputs required.
> INEXT 'PL' ; INVERS *m* CR	to set the type to PL (plot) and the version number to be deleted to m. $m = -1$ means all and $m = 0$ means the most recent (highest numbered).
> EXTDEST CR	to do the deletion.
> INVERS 0 CR	to reset the version number to its default — usually advisable.

Plot files are not amenable to the FITS format and so are not written by FITTP and FITAB. They may be copied from one catalog entry to another with TACOP.

6.3.1 Plotting your visibility data

The most basic plot program for visibility data is called UVPLT. It allows you to select the x and y axes of the plot from real, imaginary, amplitude, log of amplitude, phase and weight of the visibility, time, hour angle, elevation, azimuth, and parallactic angle, and projected baseline length, position angle, u, v, w, frequency, spectral channel number, and baseline number. It offers all of the usual calibration and data selection options and it plots the selected points individually and/or in a controlled number of bins along the x axis. It can display multiple IFs and spectral channels at once, including averaging groups of spectral channels. It can also time average the data (SOLINT minutes) prior to plotting. For example, to plot calibrator phases as a function of time for all baselines to one antenna:

> TASK 'UVPLT' ; INP ↵	to review the inputs.
> INDI *n* ; GETN *ctn* ↵	to select the disk and catalog entry of the data set.
> BPARM = 11 , 2 ↵	to plot time in hours on the x axis and phase in degrees on the y axis.
> SOURCES ' ' ; CALCODE '*' ↵	to select all calibrator sources.
> XINC 4 ↵	to plot only every fourth selected sample.
> ANTENNA 2,0 ; BASELINE 0 ↵	to do all baselines with antenna 2.
> DOTV = -1 ; GO ↵	to make a plot file of these data.

After UVPLT is running, or better, after it has finished:

> PLVER 0 ; GO LWPLA ↵	to plot the latest version on a PostScript printer/plotter.

There are several other tasks to plot your visibility data. VPLOT plots all of the parameters offered by UVPLT, but one baseline at a time with multiple baselines per page (*i.e.*, per plot file) and multiple pages per execution. CLPLT is a similar task but restricted to plotting closure phases around baselines involving 3 antennas as a function of time. CAPLT is a similar task but restricted to plotting closure amplitudes around groups of 4 baselines as a function of time. All three of these can also plot a source model (based on Clean components) as well as the observations. In 31DEC22, task CLOSE can plot the rms of closure phase or amplitude as a function of spectral channel. ANBPL can create plots of amplitudes, phases, or weights converted to antenna-based quantities; this function is helpful in diagnosing problems with your data. For spectral-line users, POSSM plots visibility spectra averaged over selected baselines and time intervals. It can also plot various spectral tables (bandpass and others), as well as the Fourier transform of visibility spectra (the auto- and cross-correlation functions). POSSM can also plot other spectral tables (BD, PD, CP, PC, and PP). For VLBI users (primarily), FRPLT plots visibilities versus time or, more importantly, the fringe rate spectrum. To examine the statistical distribution of your data, try UVHGM which plots histograms showing the number of samples or weights versus a wide variety of parameters. Optionally, it will fit a Gaussian to the histogram and plot the result. The task UVIMG will grid visibility data into an image on a variety of axes, while the task UVHIM will make an image of a two-dimensional histogram of your uv data. All image display functions below may then be used to view the results. For example, if the axes of the histogram image are the real and imaginary parts of your data, then the image will demonstrate the amplitude and phase stability present (or missing) in your data.

Two tasks, new in 31DEC21, compute and then plot statistical information from calibrated autocorrelation data. VBRFI computes the statistics as a function of frequency and antenna and then prints them to a text file and, optionally, plots them. PLRFI can plot one or more of the text files written by VBRFI. As the names suggest, these are intended to describe the RFI environment of the antennas.

A number of visibility plotting and editing tasks offer the DO3COLOR option to separate multiple IFs, spectral channels, sources, times, and the like when plotted in the same plot area. Task PLOTC will make plots showing the colors used by such tasks with labels to assist in identifying which color goes with which data parameter. These plots may be saved as plot files and shown directly on the TV just as with all other plot tasks.

Three tasks allow you to examine selected portions of a uv data set statistically. UVRMS averages the selected data in SOLINT time intervals and then can plot the average values as a function of time and as a histogram. A variety of statistical parameters are computed and printed. SPRMS determines and plots the mean and standard deviation of the selected data as a function of spectral channel. Note that both tasks will combine data from multiple sources, baselines, etc. if so instructed. RIRMS computes statistics over the included visibility data and then prints a matrix over baselines of the real and imaginary parts separately. Histogram plots and plots over time may be generated.

Two plot programs actually convert visibility data to the image plane for plotting. Observers of point objects which might vary with time either intrinsically or by scintillation (*e.g.*, stars, masers) might wish to try DFTPL, which plots the Fourier transform of the data shifted to a selected position as a function of time. Control over Stokes including polarization amplitude and popsition angle are available in 31DEC24. Tasks which save an image for later plotting include DFTIM which makes an image of such data as a function of frequency and time and ACIMG which makes a time and frequency image of autocorrelation data by antenna. VLBI spectral-line observers may need to use FRMAP, which performs imaging via fringe-rate inversion and plots the loci of possible source positions.

6.3.2 Plotting your image data

Image data may be drawn in a variety of ways including contours, grey or color levels, row tracing, and statistical. All plots are drawn with labeled tick marks although these may be suppressed with the LTYPE parameter. This LTYPE parameter is used to control the type of axis labels, the degree of extra labeling, the presence of the line giving the plot time and version, and also how metric scaling is done on the axis units. Read HELP LTYPE for details. For plots having significantly non-linear coordinate axes, *e.g.*, wide-field images, it is sometimes useful to draw a full, non-linear coordinate grid rather than just short lines at the edges of the plot. Tasks like CNTR and even the verb TVLABEL offer this option; enter DOCIRC TRUE ↵.

Plot symbols (*e.g.*, plus signs) may be drawn on the plots produced by CNTR, PCNTR, GREYS, KNTR and several of the other tasks mentioned below. In these tasks, the parameters which controls the plotting are STFACTOR, a scale factor for the symbols, and STVERS, a choice of the desired stars extension file. When using this option, there must be a table of "star" positions associated with the image being plotted. To create one, enter EXPLAIN STARS ↵ to learn the format of the input data file and the parameters for the task. See also Appendix Z or your local equivalent for instructions on editing text files. A star file may also be created by MF2ST from a model fit file produced by task SAD (see § 7.5.3 and § 10.4.4). Tasks IMFIT, JMFIT, and SAD may now also produce "star" extension files directly.

Example outputs of the following three tasks are given in Figure 6.2.

6.3.2.1 Contour and grey-scale plots

The most basic contour drawing task is CNTR. In addition to the usual image selection parameters, you may specify:

> TASK 'CNTR' ; INP ↵	to tell you what you may specify.
> BLC 250 , 230 ↵	to set the bottom left corner of plot at 250, 230 (in pixels with 1,1 at extreme bottom left of the image).
> TRC 300, 330 ↵	to set the top right corner of plot at 300, 330.
> CLEV 0 ; PLEV 1 ↵	to get contour levels at 1% of the peak image value.
> PLEV 0 ; CLEV .003 ↵	to get contour levels at 3 mJy.

> LEVS -1, 1, 2, 4, 6 ↵ — to get actual contours at -1, 1, 2, 4, and 6 times the basic level set by PLEV or CLEV. LEVS need not be integers, but very fine subdivisions cannot be represented accurately on the plot.

N.B., if you request more than one negative level with the LEVS input, you *must* use commas between the negative levels. Otherwise the minus sign(s) will be treated as subtraction symbols and the desired levels will be combined into a single negative level by the AIPS language processor. BLC and TRC can be initialized conveniently from the TV display using the cursor with the TVWIN instruction (see § 6.4.4). Then check:

> INP ↵ — to review what you have specified.

> GO ↵ — to run the task when you're satisfied with the inputs.

This generates a plot file as an extension to your image file, with the parameters you have just specified. Watch the *AIPS* monitor (which, on some systems, is your terminal) to see the progress of this task. If the "number of records used" in the plot file is over 200, the contour plot will be messy (unless the field is also large). In this case, check that you have not inadvertently set PLEV or CLEV, for example, to unrealistically low values. Printing a large, messy plot file on the printer can take a considerable length of time and will inconvenience other users. Consider plotting directly on the TV first (DOTV = TRUE) to check on your selection of contours.

PCNTR plots polarization vectors on top of contours and/or grey-scales. You may make a polarized-intensity image and a polarization position-angle image from the Q and U images (see § 7.1.2) or use the Q and U images themselves. Then:

> TASK 'PCNTR' ; INP ↵ — to review the input parameters.

> INDI *n1* ; GETN *ctn1* ↵ — where *n1* and *ctn1* select the disk and catalog numbers of the image to be contoured.

> IN2DI *n2* ; GET2N *ctn2* ↵ — where *n2* and *ctn2* select the Q or polarized intensity image.

> IN3DI *n3* ; GET3N *ctn3* ↵ — where *n3* and *ctn3* select the U or position-angle image.

> IN4DI *n4* ; GET4N *ctn4* ↵ — where *n4* and *ctn4* select the grey-scale image.

> PCUT *nn* ↵ — to blank out vectors less than *nn* in the units of polarized intensity.

> ICUT *mm* ↵ — to blank out vectors at pixels where the total intensity (image 1) is less than *mm* in the units of image 1.

> DOFRACT 1 ↵ — to convert the polarization lines to fractional polarization, dividing by the I image, before plotting.

> FACTOR *xx* ↵ — to set the length of a vector of 1 (in units of total or fractional polarization) to *xx* cell widths.

> DOCONT 1 ; DOVECT 1 ; DOGREY 4 ↵ — to request vectors plus contours of image 1 and grey scale of image 4.

> PIXRAN T_{min}, T_{max} ; FUNCTYP ' ' ↵ — to scale linearly the grey-scale values from T_{min} to T_{max}.

> OFMFILE 'RAINBOW' ↵ — to use the standard rainbow-colored OFM table to pseudo-color the grey scales.

> CBPLOT 1 ↵ — to plot the Clean beam in the lower left corner. See HELP CBPLOT for numerous options.

> PVPLOT 4 ↵ — to plot a polarization vector of stated length in Jy/beam in a box in the upper left corner.

> INP ↵ — to review your inputs and remind you of others. Most are similar to those in CNTR and sensibly defaulted.

> GO ↵ — to generate the plot file, which can then be routed to output devices via TKPL, TVPL, LWPLA etc.

Unless images 2 and 3 are of Q and U polarization, the lengths of the vectors are controlled by image 2 while the directions of them are controlled by image 3. Clearly this program can also be used for other

combinations of images, so long as one of them represents an angle. PCNTR and following tasks have the option to draw an image of the Clean beam under control of adverb CBPLOT. Polarization vectors may be plotted with the color representing the angle. The value of POL3COL, if greater than zero, is that angle represented in pure red from 0 to 180 degrees. A color "spray" is plotted to calibrate the eye. To clarify the length scale of polarization vectors, a vector may be drawn in a box in an image corner under control of PVPLOT with the length of that vector in Jy/beam shown in a text line below the plot. The ability to plot multiple spectral planes in colored contour or polarization vectors was also added. The adverb CON3COL controls this function. The ability to select the colors of each contour level with RGBLEVS has been added. These color functions are displayed at the end of this chapter in the color pages.

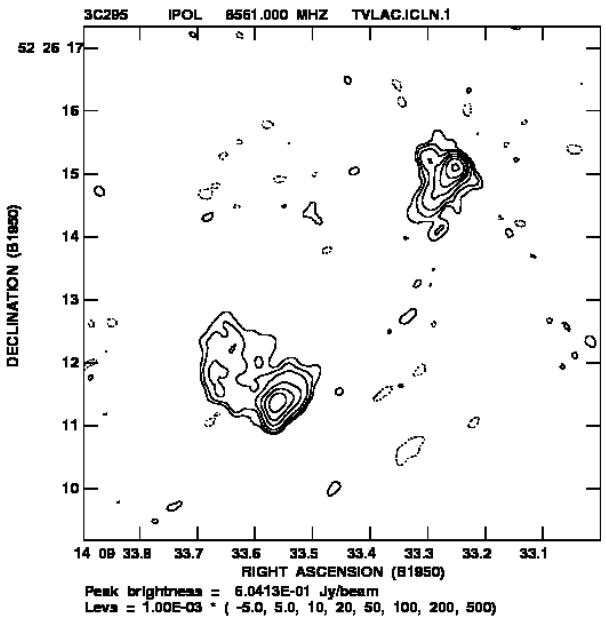
GREYS creates a plot file of the grey-scale intensities in the first input image plane and, optionally, a contour representation of a second input image plane. Like the other grey-scale plotting tasks, GREYS can interpret a true-color (RGB) image cube in its "true" colors. Unlike the others, it can construct the true-color image from 3 separate image planes. A sample set of inputs could be:

> TASK 'GREYS' ; INP ↵	to review the inputs.
> DOCOLOR 1 ↵	to specify that a "true-color" image is to be plotted.
> INDISK *n1* ; GETN *ctn1* ↵	to select the red image.
> IN3DISK *n3* ; GET3N *ctn3* ↵	to select the green image.
> IN4DISK *n4* ; GET4N *ctn4* ↵	to select the blue image.
> PIXRAN T_{minr}, T_{maxr} ; FUNCTYP 'SQ' ↵	to scale by a square-root function red values from T_{minr} to T_{maxr}.
> APARM $T_{ming}, T_{maxg}, T_{minb}, T_{maxb}$ ↵	to scale green and blue values similarly over ranges T_{ming} to T_{maxg} and T_{minb} to T_{maxb}, respectively.
> BLC 250 , 250 , 3 ↵	to select the lower left corner and the plane in the first image.
> TRC 320 , 310 , 12 ↵	to select the upper right corner in the first image and, with TRC(3), the plane in the second image.
> DOCONT TRUE ↵	to specify that contours are to be drawn.
> IN2D *n2* ; GET2N *ctn2* ↵	to select the contour image.
> PLEV 0 ; CLEV 0.005 ↵	to select 5 mJy/beam contour increments.
> LEVS -3 , -1 , 1 3 10 30 100 ↵	to plot contours at -15, -5, 5, 15, 50, 150, and 500 mJy/beam.
> DOWEDGE 2 ↵	to plot a 3-color step-wedge along the right-hand edge; 1 for along the top and 0 for no wedge.
> DOPROFIL 1 ↵	to plot a "profile" of the average of each column of the image.
> DOTV FALSE ; GO ↵	to make the plot file.

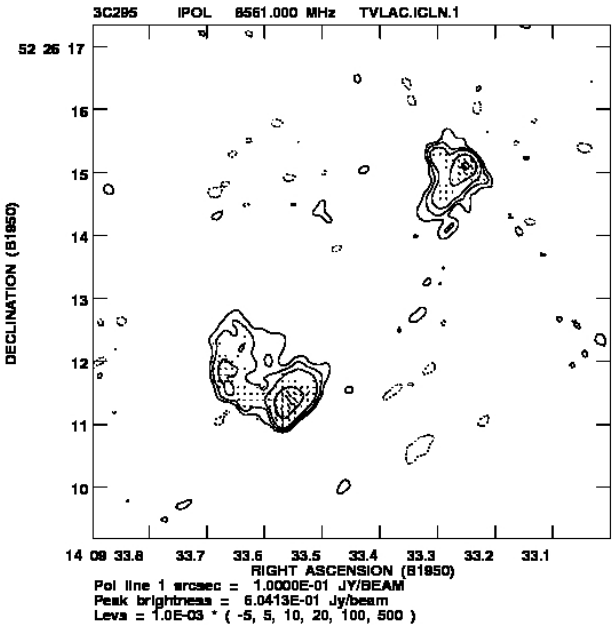
In 31DEC25, GREYS has the added option of plotting a "profile" at the top averaging all values in each column of the image and/or at the right averaging all values in each row of the image. This is illustrated in Figure **??**. New task PRPLT will plot up to 3 profiles generated by averaging an image cube along 2 axes as a function of the third axis.

When GREYS has finished, run LWPLA to view the plot file. Note that LWPLA has a variety of options which control the plotting and scaling of the grey-scale images without having to rerun GREYS. In this example case, you should remember to set FUNCTYPE = ' ' and DPARM = 0 (or at least the first 4 values to 0) in LWPLA to avoid additional scaling. You may wish to color the labeling, contours, and background with DOCOLOR=1 and PLCOLORS with LWPLA. The procedure TVCOLORS will set PLCOLORS to match the TV graphics-plane colors, DEFCOLOR will set PLCOLORS to the standard TV colors even in the absence of a TV display and OKCOLORS will set colors which work well with a white background, using less printer ink. See examples on the color pages at the end of this chapter.

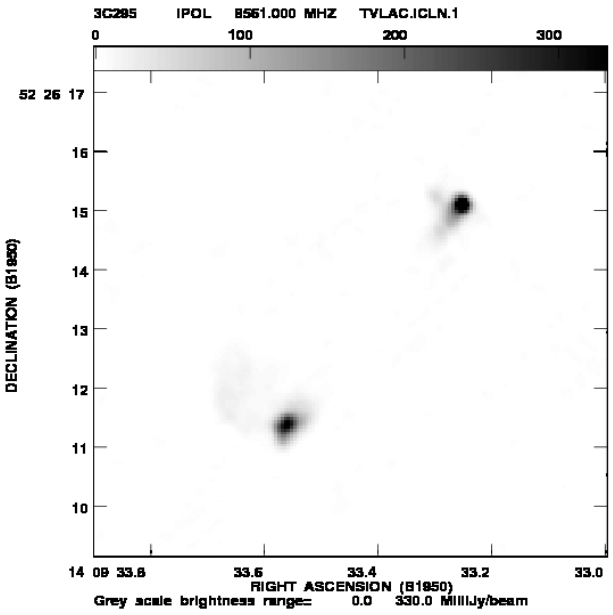
There are two other contour drawing tasks which offer additional options. KNTR is able to draw multiple contour, polarization, and/or grey-scale images in a single plot file, primarily to show multiple planes of a spectral-line cube; see § 8.5.4. KNTR uses a different, and probably superior, method of drawing the contour



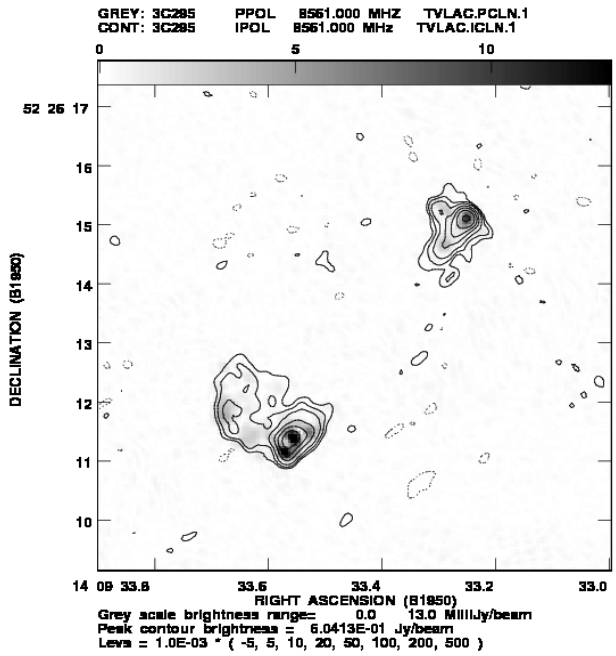
CNTR



PCNTR



GREYS



GREYS with contours

Figure 6.2: Contour, polarization, and grey-scale plots of an image

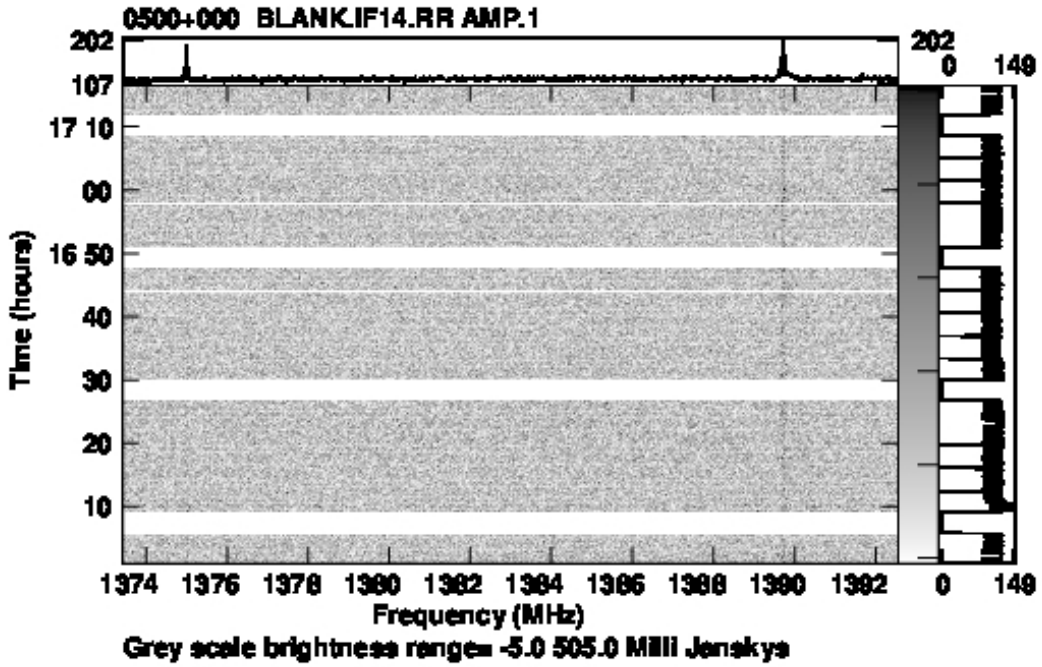


Figure 6.3: GREYS plot of the output of DFTIM with profiles at top and right-hand side (DOPROFIL = 3).

lines. It also can use color to represent different spectral channels and/or polarization angles or simply different colors for the different contour levels. CCNTR is virtually identical to CNTR except that it can draw extra symbols on the plot representing the locations and intensities of source model components found in CC (Clean component) or MF (model fit Gaussians from SAD, see § 7.5.3 and § 10.4.4).

6.3.2.2 Row tracing plots

There are a number of tasks which plot rows directly. Two of these are for use with single image planes while others are more intended for use with, *e.g.*, spectral-line data cubes transposed into velocity-ra-dec order. Of the former, PLROW is the simpler. It makes a plot file of all selected rows in an image plane. Each row is plotted as a slice offset a bit from the previous row. Low intensities which are "obscured" by foreground (*i.e.*, lower row number) bright features are blanked to keep the plot readable. Example inputs would be:

> TASK 'PLROW' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to select the image on disk *n* catalog slot *ctn*.
> BLC 100 ; TRC 300 ↵	to select the sub-image from (100, 100) to (300, 300).
> YINC 3 ↵	to plot only every $3^{\rm rd}$ row.
> PIXRANGE -0.001 0.050 ↵	to clip intensities outside the range -1 to 50 mJy.
> OFFSET 0.002 ↵	to set the intensity scaling such that 2 mJy separates rows of equal intensity.
> INP ↵	to check the inputs.
> GO ↵	to run PLROW.
> GO LWPLA ↵	to display the plot file on the laser printer after PLROW has finished.

The plot files produced by PLROW are a simple, special case of those produced by PROFL. This task makes a

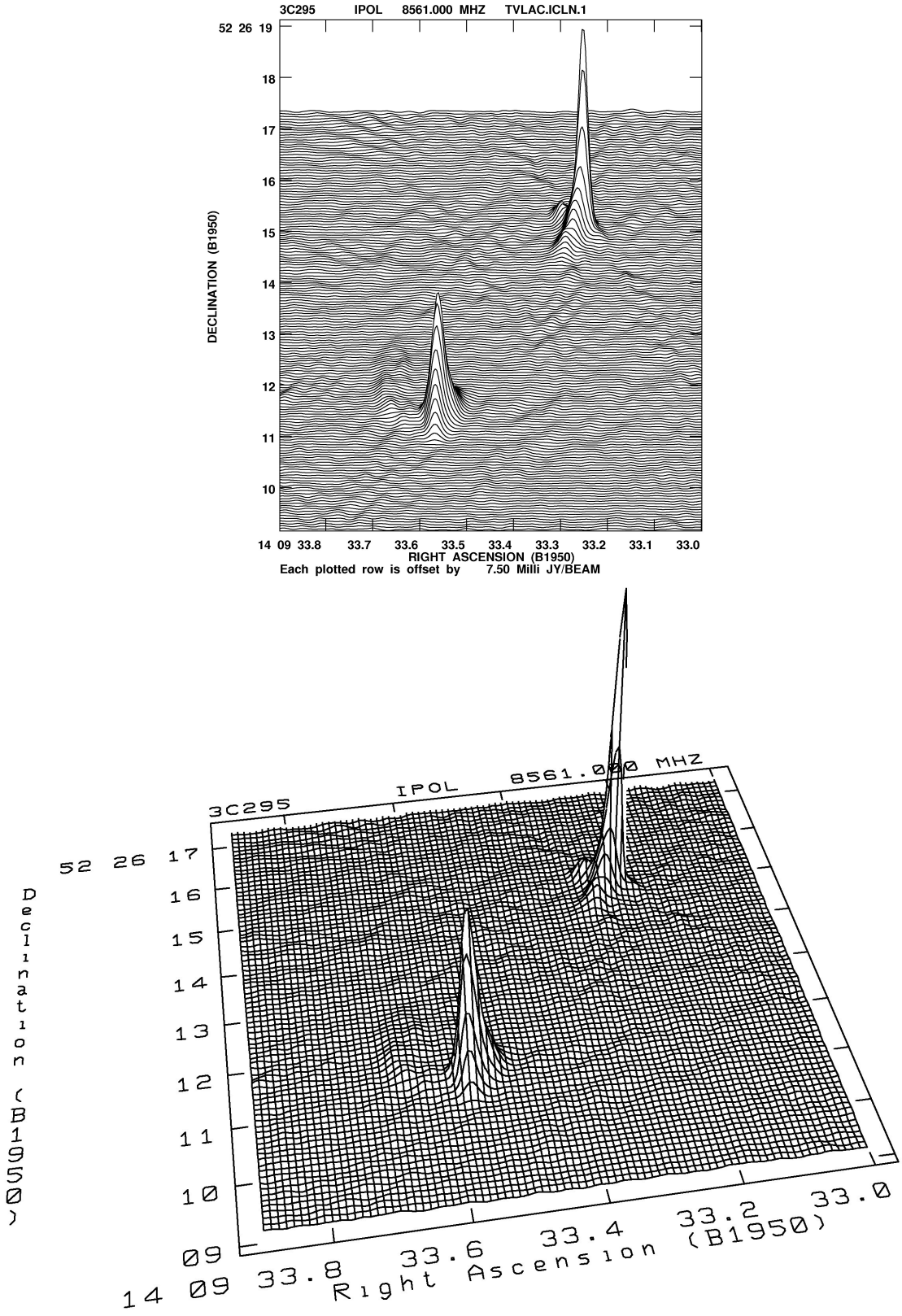


Figure 6.4: Row plots of an image with (bottom, PROFL) and without (top, PLROW) perspective

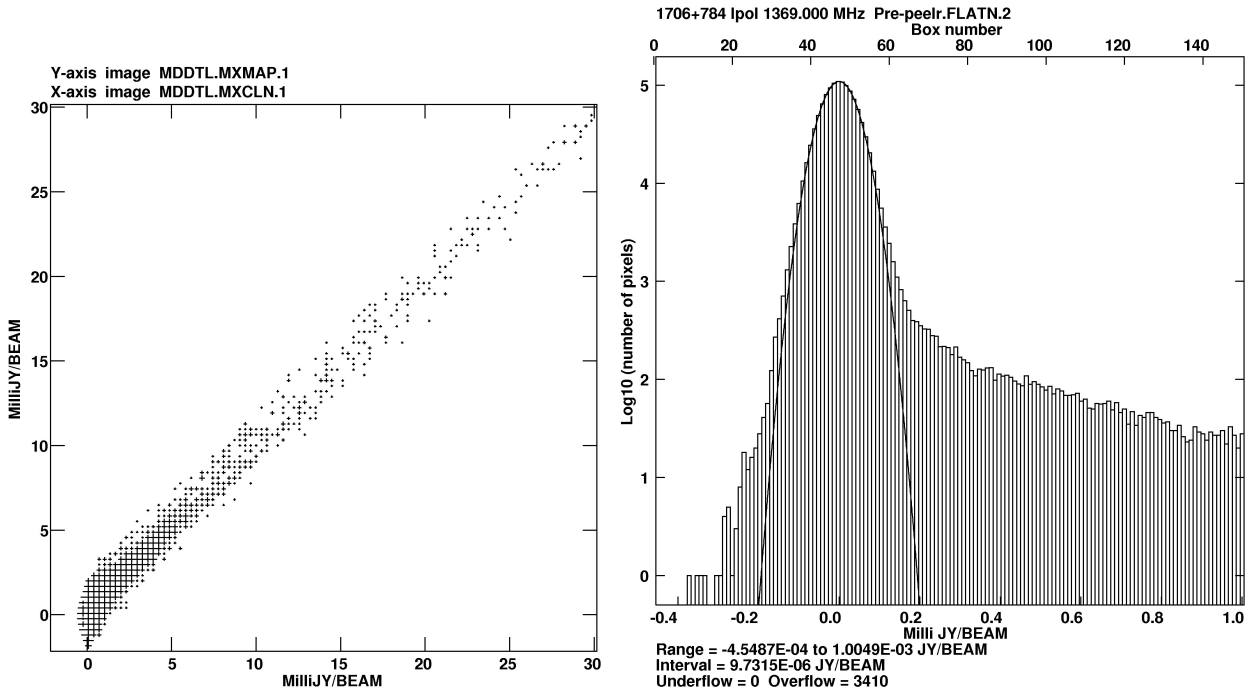


Figure 6.5: Plots of statistical parameters of an image.

plot file of a "wire-mesh" representation of an image plane complete with user-controlled viewing angles and correct perspective. Enter EXPLAIN PROFL ↵ for a full description. Both of these tasks are especially useful where the signal-to-noise ratio is high and examples of them are given in Figure 6.3.2.2.

In Chapter 8 we discuss the computation and use of "slices," one-dimensional profiles interpolated along any line in an image plane. Once a slice has been computed, it may be plotted by SL2PL on the TV or into a device-independent plot file.

Three other row-plotting tasks, PLCUB, ISPEC, and BLSUM, are designed primarily for spectral-line and other data "cubes" (see § 8.5.4 and § 8.6). PLCUB makes one or more plot files showing the intensities in each selected row. The row subplots are positioned in a matrix in the coordinates of the $2^{\rm nd}$ and $3^{\rm rd}$ axes of the cube. ISPEC averages circular or rectangular areas in each plane of a cube and plots the resulting spectrum. It can also save the output in a SLice file. RSPEC does the same except that it plots the robust or median absolute deviation ("MAD") rms in each plane rather than the data. RSPEC has options to write out a signal-to-noise image and/or a text file of channel weights as well. BLSUM can plot spectra from irregular regions selected on the TV with a "blotch" algorithm.

6.3.2.3 Miscellaneous image plots

IMVIM allows a variety of image comparisons by plotting the pixel values of one image against the pixel values of another image. The special options include binning the values (and plotting symbols proportional to the number of samples in a bin) and shifting one of the images in x and/or y with respect to the other. The former reduces large scatter diagrams to more manageable sets of numbers while the latter allows cross-correlation functions to be developed.

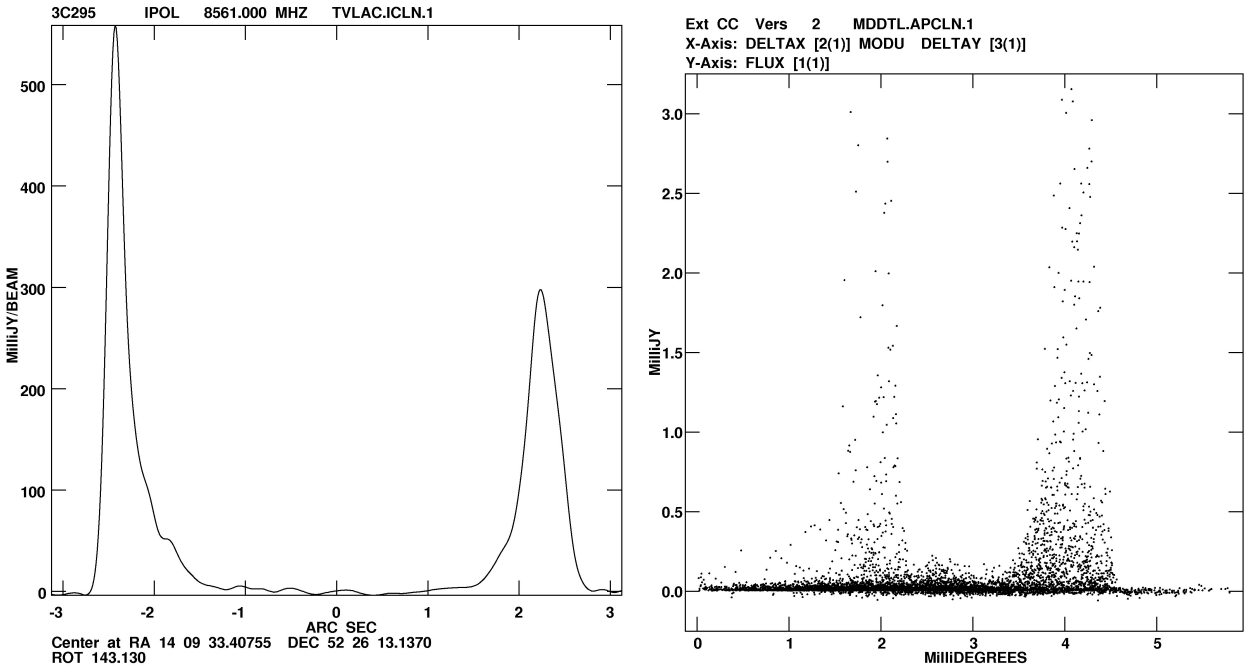


Figure 6.6: Slice and table plots.

IMEAN prints the mean, rms, and extrema inside or OUTSIDE a user-specified window in an image. It also prints the intensity at, and rms of, the noise peak in the histogram and returns these values as adverbs to AIPS. It also returns an array in which TRIANGLE(i) is the brightness level above which there are i per cent of the image pixels. This is often useful in setting PIXRANGE for optical images. Optionally, IMEAN plots histograms of image intensities over the window using a user-specified number of summing cells over a user-specified range of intensities. Optionally, it also plots the fit to the noise peak on the user-specified histogram. An example of this is also shown in Figure 6.5.

6.3.3 Plotting your table data

TAPLT is a very general task to plot information from *AIPS* table extension files. It can plot a histogram of a function of the values in one or two columns of the table and it can plot a function of one or two columns against another function of another one or two columns. The latter can be summed or averaged in bins or have every point plotted. At first blush, the inputs seem rather complicated, but the results may well justify some effort to understand them. For example, to plot Clean component fluxes as a function of radius from the image center:

> TASK 'TAPLT' ; INP ↵	to review the inputs.
> INDISK n ; GETN ctn ↵	to select the image on disk n catalog slot ctn.
> INEXT 'CC' ; XINC 1 ↵	to select every row in the Clean components file.
> BCOUNT 1 ; ECOUNT 0 ↵	to select all rows in the table.
> APARM = 2 , 1, 3 , 1 , 16 , 1 , 1 ↵	to plot the modulus of columns 2 and 3 on the x axis (*i.e.*, $\sqrt{\Delta_x^2 + \Delta_y^2}$) and column 1 on the y axis (*i.e.*, component flux).
> BPARM = 0 ; CPARM = 0 ↵	to use self-scaling of the plot and no scaling of the column values.

> DOTV TRUE ; GO to plot the fluxes on the TV screen.

You may need to set the scaling with BPARM after seeing the preview plot in order, for example, to bring out the details in the low-level components.

There are a few tasks intended to make plotting specific kinds of tables rather easier. SNPLT plots calibration, solution, system temperature and EVLA SysPower tables for selected antennas as a function of time, antenna elevation, hour angle, azimuth, or sidereal time. It will make several plots per page (one antenna per plot) and multiple pages if needed, or you can plot the most discrepant value over all antennas in a single plot. In one execution, polarizations and IFs may be plotted on separate plots or together on the same plots, optionally separated by color. SNPLT also has the option to separate sources by color instead. You can also plot multiple parameter types in the same execution. SNBLP plots calibration solutions much like SNPLT, but on a baseline basis, to investigate exactly how the solutions affect the visibility data. SNIFS is similar to SNPLT but is intended to compare solutions at different IFs, plotting the IF on the x axis with a variety of binning options.

POSSM will plot bandpass tables when APARM(8) is set to 2. Values 3 – 10 plot bandpass-like tables from BLCHN (BD), PCAL (PD antenna polarization and CP source polarization), VLB pulse-cal (PC), and RLDIF (PP phase difference). POSSM can also do multiple plots per page with all the usual data selection adverbs. BPLOT also plots BP and the other similar tables, with multiple times for one antenna or multiple antennae for one time on each plot. Combined with difference and coloring options, this allows close comparison of bandpasses as functions of time or antenna. BLPLT makes a similar plot of the BL table written by BLCAL. BPEPL overplots multiple times within a BP table using color to separate times. It is similar to BPEDT without the interactivity. In 31DEC23, BPPLT plots selected portions of bandpass tables as a function of time in a manner similar to SNPLT. It is particularly useful when BPASS is used instead of CALIB to provide the gain calibration. WETHR plots the data in a WX weather table including parameters computed from those data such as relative humidity. FGPLT plots the times of selected flags from a flag table. PRTAN can plot antenna locations. Task TEPLT may be used to plot the ionospheric Faraday rotation and other parameters used by TECOR and written in a TE table. New tasks ALBUS and RIFRM also write the TE table filling in some of the columns.

6.3.4 Plotting miscellaneous information

There are a few other tasks which create plot files, but which do not fit into the categories above. The most general of these is PLOTR which can plot up to 30 sets of *(x,y)* points input from a text file with coloring options. In 31DEC25. PLOTR can also read and plot a column of error bars if desired. Task PCHIS also reads a text file to produce a histogram plot of the values in any selected column of the file. CONPL is a task which plots *AIPS* convolving functions (used by various *uv*-data gridding tasks such as IMAGR) and their Fourier transform, expected signal-to-noise ratio, or convolution with a user-specified Gaussian. IRING integrates an image in concentric annuli about the user-specified object center with specified major axis position angle and inclination. The results may be placed in a plot file for later display. GAL calculates the orientation and rotation curve parameters of a galaxy from an image of the predominant velocities. The observed rotation curve is plotted together with the fitted model curve. LOCIT fits antenna location corrections to SN tables; the residual phases may be plotted as a function of sample number.

There are several tasks which create RGB cubes for later display by tasks such as GREYS and KNTR. These include RGBMP which does a weighted sum of the planes of a data cube, HUINT which uses two images as hue and intensity to construct an RGB cube, and TVHUI which interactively uses three images as intensity, hue, and saturation to construct an RGB cube by a different algorithm. Task SCLIM will scale and clip image planes to be used as inputs to LAYER, which produces an RGB cube from the colored sum of up to 10 input image planes using a complicated and general algorithm.

6.4 Interactive TV displays of your data

The *AIPS* TV display allows you to look at your image data in detail and to set parameters by pointing at interesting features visible on the screen. Although *AIPS* will run on a variety of hardware display devices (*e.g.*, I^2S Models 70, 75 and IVAS and DeAnza), it is now used almost exclusively with the X-Windows TV-simulation program called XAS. See § 2.3.2 for information on using this basic *AIPS* tool. If the XAS window has closed for any reason, you may restart it without exiting AIPS with the verb STARTTV or TVRESTOR. Information on TV verbs and tasks may be found by entering ABOUT TV ↵, ABOUT INTERACT ↵, and ABOUT TV-APPL ↵, or by consulting the corresponding sections of Chapter 13 of this *CookBook*.

6.4.1 Loading an image to the TV

The simplest way to load an image from your catalog to the TV and then to manipulate the display is with the procedure called TVALL:

> INP TVALL ↵ — to review the input parameters.

> INDI *n* ; GETN *ctn* ↵ — to select the disk and image name parameters from the catalog.

Use one of the following commands to specify the initial transfer function that converts your image file intensities to display pixel intensities:

> FUNC 'LN' ↵ — linear—this is the default.

> FUNC 'LG' ↵ — logarithmic.

> FUNC 'L2' ↵ — "extra" logarithmic.

> FUNC 'SQ' ↵ — square-root, often a good compromise

> FUNC 'NE' ↵ — negative linear.

> FUNC 'NG' ↵ — inverse logarithmic.

> FUNC 'N2' ↵ — inverse "extra" logarithmic.

> FUNC 'NQ' ↵ — negative square root.

> PIXRA *x1*, *x2* ↵ — to load only image intensities *x1* to *x2* in the units of the image. The default is to load the full range of intensities in the image.

(The slopes and intercepts of the display transfer functions can be modified later, but the above options let you choose initially between linear, square-root, logarithmic, and negative displays and restrict the range of intensities that is loaded). Then:

> TVALL ↵ — to load the selected image.

The image should appear on the TV screen in black-and-white. If you see a new image but it is not what you expected, hit button D to end TVALL and then review your inputs with:

> INP TVALL ↵

If no image appears, make sure that AIPS started up with your workstation assigned as the display; see § 2.2.3 for information on starting up AIPS and assigning the TV display.

After your image has been displayed on the TV by TVALL, your trackball with its buttons (which should be labeled A, B, C, and D) — or your workstation mouse with keyboard buttons A, B, C, and D) — can be used to modify the display transfer functions, coloring and zoom. Pressing button A alone enables black-and-white and color-contour coding of the image intensities, successively. Adjust the cursor position on the TV (using the trackball or mouse) to vary the slope and intercept of the display transfer function. TVALL will superpose a calibrated horizontal wedge on the image. This should help you to choose the optimum cursor setting for the display. Black-and-white displays are generally much more suitable than color for high-dynamic-range images, while color contouring may be used to accentuate interesting features. (Note also that a much wider range of image-coloring options is available outside TVALL by invoking TVPSEUDO, TVPHLAME, and TVHELIX.)

Pressing buttons B and C adjusts the zoom of the display: B to increase the magnification and C to decrease it. When these buttons are enabled, the cursor controls the position of the center of the zoomed field of view. Magnification factors of 1 through 16 are available on most workstations. Note that your terminal issues instructions when buttons are pressed, but that it is in the death-like grip of TVALL otherwise until you press button D to exit from it.

The size of the image which can be displayed depends on the size (in pixels) of your workstation screen or TV display and some other parameters which can vary from site to site. A typical Sun workstation can display up to 1024 by 900 pixels. Then, if the image is larger than about 800 pixels or so in the *y*-direction, portions of the labeling of the wedge (the units) will be omitted or superposed on top of the wedge (the tick numeric values). A useful technique for displaying large images is to load only alternate pixels. The command:

> TXINC 2 ; TYINC 2 ↵ — to load every other *x* and *y* pixel.

before TVALL would do this. Also use:

> TBLC = *n1*, *n2*, *n3*, ... ↵ — bottom left pixel to load.

> TTRC = *m1*, *m2*, *n3*, ... ↵ — top right pixel to load.

to limit the displayed field. A small image may be interpolated to fill the TV screen by setting TXINC = -1 ; TYINC = -1 ↵. Recent versions of XAS allow the verb TVROAM to function again. This verb loads adjacent portions of an image to as many as 16 memories and then "roams" with a split screen to allow you to view any contiguous portion of the loaded image. REROAM resumes roaming an already loaded roam-mode image, perhaps after adjustment of the transfer function of all channels with TVTRANSF and TVCHAN = 0. The verb ROAMOFF allows you to convert the final split-screen image into a normal, more useful single-plane image.

TVALL is a procedure that insures that the desired graphics and TV channels are on and cleared and the others off, then loads the image with verb TVLOD, loads a wedge with TVWEDGE, and labels the wedge with TVWLABEL. You do not have to use TVALL, which can be rather slow, simply to load a new image on the TV. Use TVLOD instead. TVWEDGE has a variety of options concerning the width of the wedge and its position and direction; type HELP TVWED ↵ for details.

The task TVHLD was completely revised to load images to the TV with histogram equalization. The task has interactive controls over the intensity range and method of computing the histogram and can write out a histogram-equalized image in arbitrary, non-physical units for display by the various plotting tasks.

6.4.2 Manipulating the TV display

There are a number of verbs which allow you to manipulate the display, including:

> STARTTV ↵ — to start the TV, message, and Tektronix servers when one or more of them have exited.

> TVINIT ↵ — to initialize the entire TV and TV image catalog. The image catalog is now kept in XAS and so is always current.

> TVON *n* ↵ — to turn on TV grey channel(s) with *n* being a bit pattern of the desired channels: $2^{(i-1)}$. Thus TVON 4 turns on channel 3. You may have to turn off other channels to see the desired channel, since the sum of two images may be rather odd.

> TVOFF *m* ↵ — to turn off channel *m*, where *m* is a bit pattern as with TVON.

> GRON *n* ↵ — to turn on one or more of the 8 graphics channels, where *n* is a bit pattern.

> GROFF *m* ↵ — to turn off one or more graphics channels.

> TVCHAN *n* ; TVCLEAR ↵ — to zero a TV channel.

> GRCHAN *m* ; GRCLEAR CR	to zero a graphics channel.
> TVZOOM CR	to set the zoom magnification and center interactively, follow instructions on the screen.
> OFFZOOM CR	to reset the zoom and zoom center to null.
> GREAD CR	to read the current color of a specified graphics overlay channel into RGBCOLOR.
> GWRITE CR	to change the color of a graphics overlay channel to that specified by RGBCOLOR. This may be done for aesthetic reasons or because the default colors may not show up well when captured by TVCPS and printed on a color printer.
> TVPOS CR	to read the TV cursor position, returning adverbs for use in procedures or other verbs.
> IM2TV CR	to convert an image pixel in PIXXY to the corresponding TV pixel.
> TVDATA CR	to display lots of information about your current TV display.
> TVFACTS CR	to display lots of information about your current TV display.

6.4.3 Intensity and color transfer functions

The *AIPS* model of a TV postulates two intensity transfer functions, called the LUT and the OFM, which are basically multiplicative. In most circumstances, the LUT is used for black-and-white enhancements and the OFM for coloring, but both can be used for either. To manipulate the LUT interactively, while leaving the pseudo-coloring alone, use the TVTRANSF verb. The cursor position controls the slope and intercept of the transfer function, buttons A and B switch a plot of the transfer function on and off, button C switches the sign of the slope, and button D (as always) exits. In 31DEC21, verb TVBANDW implements 3 types of black-and-white modifications of the LUT: a series of step wedges, sine waves, or stairs in intensity. To turn the LUT back to normal, enter OFFTRAN.

A rich zoo of color coding is available with TVPSEUDO, which alters the OFM while leaving the LUT alone. Repeated hits on button A select a variety of color triangles, button B selects a circle on hue, and repeated hits on button C select a variety of color contours. First-time users should experiment with the *AIPS* coloring options until they develop an intuitive feel for the effects of cursor settings on the image appearance. The wedge displayed by TVALL adjusts to the alternative colorations selected with TVPSEUDO, and it is helpful to watch changes in both the wedge and the image. A flame-like coloring is available with TVPHLAME, or variations on the scheme with repeated hits on buttons A or B. TVHELIX does a helix in color space attempting to make a monotonic increase in perceived intensity. The buttons change direction, number of rotations, and saturation. For images that have significant negative and positive values, TV2COLOR uses two color wedges increasing away from the user-adjusted central value. In these verbs the cursor position controls aspects of the coloring such as enhancements, richness, or cycles of hue. To turn off pseudo-coloring, enter OFFPSEUD.

A set of less well-known verbs is available to allow you to create, manipulate, and save desirable versions of the OFM table. OFMSAVE allows you to save a named OFM, OFMDIR lists all saved OFMs belonging to you or generally available from the *AIPS* distribution, OFMGET loads a named OFM, OFMZAP deletes a named OFM, and OFMLIST prints the current OFM. OFMCONT is an elaborate interactive verb which allows you to set the hue, intensity, and saturation of the OFM divided up into a number of color contours. Each of these contours can be a constant level or a step wedge. OFMADJUS is another elaborate interactive verb to alter pieces of the OFM, while OFMTWEAK is a simpler verb to stretch the OFM.

The OFM received particular attention in 31DEC25. A significant number of OFM tables described on the Internet were adapted and added to the *AIPS* distribution. A new pseudo-coloring verb OFMCOLOR was

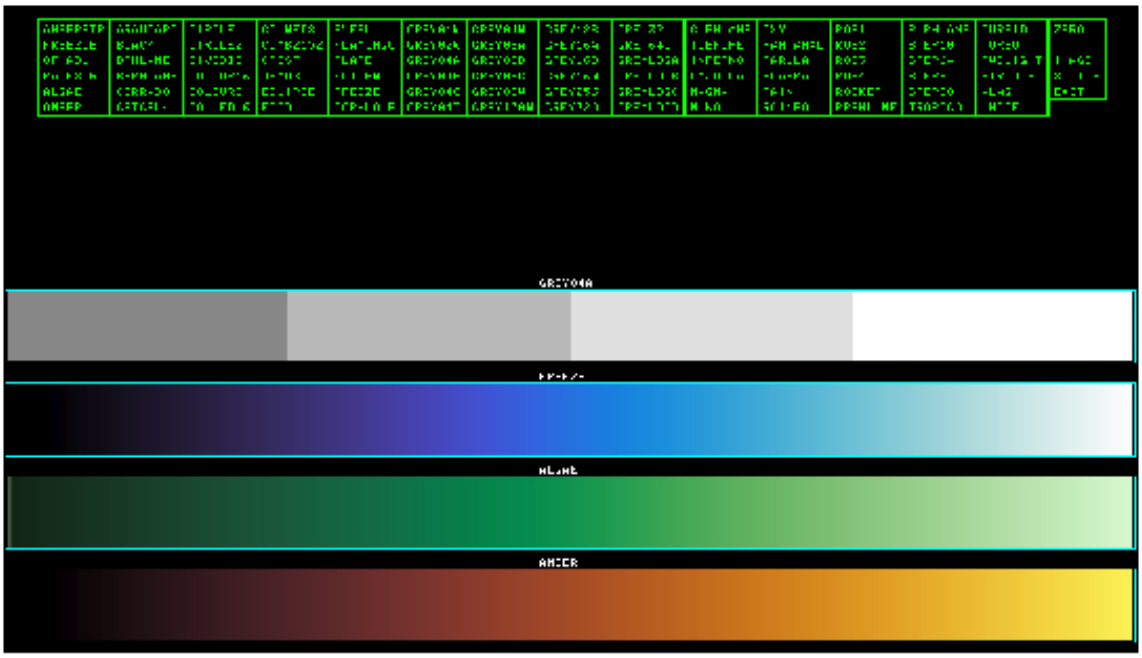


Figure 6.7: Example step wedge display from OFMPL

added to AIPS and to the menus of multiple image-display tasks such as TVSPC and XGAUS. This function begins by loading one of the new OFMs. You may then select different OFMs by hitting buttons A, B, or C. Each button selects from a list of 10 of the new OFMs. Having selected an OFM, you may then shift the center of the function using the horizontal position of the cursor. Another new verb OFMSTRCH does this stetching of the current OFM outside of OFMCOLOR. To view the available OFMs distributed with *AIPS* plus any that you may have developed, the new task OFMPL was developed. It has three displays: up to 4 step wedges, a plot of the red, green, and blue parts of the OFM, and an optional image colored by the OFM. All three of these show a menu at the top of the TV listing all available OFMs plus items to select the desired display. See an example in Figure 6.4.3.

6.4.4 Setting parameters with the TV

One reason to load the image to the TV is to set adverbs for use by other verbs and tasks. Verbs which use the TV cursor to set adverbs include:

> TVNAME ↵ — to set INDISK, INNAME, etc. to the name parameters of the image currently visible. If there is an ambiguity, you will be asked to move the cursor to the desired image and press a button.

> MFITSET ↵ — allows you to set the adverbs for IMFIT and JMFIT, namely the image name parameters, window, and initial guesses for the Gaussians from the image on the TV.

> SETXWIN (*dx*,*dy*) ↵ — reads pixel coordinate of the center of a *dx*-pixel by *dy*-pixel window and sets the adverbs BLC and TRC.

> TVWIN ↵	reads pixel coordinates from the next two cursor positions at which a trackball button is depressed. The TV graphics shows the current shape and position of the window. Button A allows you to switch to (re)setting the other corner while the other buttons exit after both corners have been set. TVWIN uses the pixel coordinates to set up the bottom left (BLC) and top right (TRC) corners of an image subsection, *e.g.*, for input to the contouring programs CNTR and PCNTR, to the mean/rms calculator IMEAN, and to many other tasks.
> SETSLICE ↵	works like TVWIN above to set BLC and TRC. Instead of a rectangle however, the display shows a diagonal line which is useful for setting the ends of slices.
> TVBOX ↵	is similar to TVWIN above except that it is used to set up pixel coordinates to define rectangular *or circular* Cleaning areas for the *AIPS* Clean tasks. The adverbs NBOXES and CLBOX are set. The circular option appeared in the 15JUL95 release and is not supported by APCLN.
> REBOX ↵	allows revision using the TV of the Cleaning areas set previously with TVBOX. Revises NBOXES too.
> DELBOX ↵	allows deletion using the TV of the Cleaning areas set previously with TVBOX. Revises NBOXES too.
> FILEBOX ↵	is REBOX for boxes in the text file used with IMAGR; multiple fields and many more boxes are allowed.
> DFILEBOX ↵	is DELBOX for boxes in the text file used with IMAGR; multiple fields and many more boxes are allowed.

Task FILIT performs FILEBOX and much more on a set of images, usually the multiple facets of a wide-field imaging. It offers a wide range of image display options including handling images larger than the TV, allows editing of the box information, creates boxes via the auto-boxing algorithm, and checks the multiple facets for overlapping boxes. This new task may be the quickest and easiest way to review a set of facet images.

6.4.5 Reading image values from the TV

There are several facilities for reading out intensity and position information from displayed images using the TV cursor:

> IMPOS ↵	displays the two coordinate values (*e.g.*, RA and Dec) from the cursor position when any button is depressed. Adverbs TVBUT and COORDINA are returned.
> IMXY ; IMVAL ↵	displays the image intensity and the two coordinate values (*e.g.*, RA and Dec) from the cursor position when any button is depressed. Adverbs PIXXY, TVBUT, PIXVAL, and COORDINA are set.
> CURVALUE ↵	continuously displays (in the upper-left corner of the TV) the pixel coordinates and the image intensity in user-recognizable units at the position selected by the TV cursor.
> TVFLUX ↵	displays image intensities and coordinates whenever a TV button is pressed, looping until button D is pressed. Adverbs for the first image name are set as well as PIXXY, TVBUT, PIXVAL, and COORDINA for the last pixel selected.

> TVDIST ↵ displays the angular length and position angle of the spherical vector between two pixels in one or two images shown on the TV. Name adverbs for input files 1 and 2 are set as well as adverbs PIXXY, PIX2XY and DIST.

> TVMAXFIT ↵ whenever a TV button is pressed, fits a quadratic function to the image to find the position and strength of an extremum, looping until button D is pressed. Adverbs for the first image name are set as well as PIXXY, TVBUT, PIXVAL, and COORDINA for the last object selected.

> TVSTAT ↵ determines the mean, rms, extrema and integrated intensity (if appropriate) in user-defined "blotch" regions within the image currently displayed on the TV. The regions are irregular polygons selected with the TV cursor. Type EXPLAIN TVSTAT ↵ for details. Adverbs PIXAVG, PIXSTD, PIXVAL, PIXXY, PIX2VAL, and PIX2XY are set.

> COPIXEL ↵ to convert between pixel and astronomical coordinates for an image; allows determination that a coordinate is not inside an image.

6.4.6 Labeling images on the TV

There are a number of facilities for labeling images on the TV including:

> CHARMULT /CR to scale the character size used by the TV by 1 through 5 times the standard size which is 7 by 9 pixels. Many menu-driven interactive tasks include this option when run on larger TV displays.

> TVLABEL ↵ to draw standard axis labels around the visible image. You may control the type of labeling and whether coordinates are shown as a grid or short tick marks. A Clean beam may be plotted. If more than one image is visible, you will be asked to indicate which one you want with the cursor and any button.

> TVWLABEL ↵ to draw axis labels around the visible intensity wedge.

> TVANOT ↵ to draw a text string into a grey-scale channel or a graphics plane at a location specified via an adverb or via the TV cursor.

> TVLINE ↵ to draw a straight line into a grey-scale channel or a graphics plane at locations specified in part or in whole via adverbs or via the TV cursor.

> TVILINE ↵ to draw a straight line into a grey-scale channel or a graphics plane between two image pixel coordinates.

> COSTAR ↵ to plot a "star" positions at a user-specified coordinate on the TV image.

> TVSTAR ↵ to plot "star" positions from an ST file on top of the visible image; see § 6.3.2.

6.4.7 Comparing images on the TV

It is often useful to compare two images, *e.g.*, to decide whether one contains artifacts that are not present in another at the same frequency, or to look for frequency-dependent features at constant resolution. *AIPS* provides several tools for such image comparisons.

The first tool is a capability for loading multiple images to the same plane (or channel) of the TV device. The parameter TVCORN specifies where the bottom left corner of the image or image subsection will be positioned in the TV frame by TVLOD or TVALL. If TVCORN is left at zero, TVLOD and TVALL adjust it to center the displayed image. You may however use TVCORN to control loading successive images to different regions of the display with successive executions of TVLOD. For example, the following commands would load two 512 by 512 pixel images from slots 1 and 2 on disk 1 *side-by-side* on channel 1 of a 1024 by 900 TV display:

> INDI 1; GETN 1 C_R	to select the input disk and the first image.
> TVCH 1 C_R	to select TV channel 1 for the loading.
> TVCORN 1 193 ; TVLOD C_R	to load the first image.
> GETN 2 C_R	to select the second image.
> TVCORN 513 193; TVLOD C_R	to load the second beside the first.

You could then adjust the color coding, transfer function, etc. for both images simultaneously with TVFIDDLE or TVTRAN. You may load as many as 256 images to a single TV plane with this technique, which is therefore a powerful method for making "montages." The number of simultaneous images is limited mostly by your image sizes, the need to avoid overlaps (which are allowed if you want) — and your ability to do the arithmetic for appropriate TVCORN settings! You are also limited by the need for all the images in one plane to share that plane's transfer function. Judicious use of the PIXRANGE and FUNC inputs to TVLOD permits making useful montages of disparate images, however.

A second tool is the classic "blink" technique from optical astronomy. TVBLINK allows you to load images to two different planes of the TV memory and then to alternate the display rapidly between the two. The two images described above could be "blinked" against each other by the following command sequence:

> INDI 1; GETN 1 C_R	to select the input disk and first image.
> TVINIT; TVCORN 0	to clear the TV and restore the default positioning.
> TVCH 1; TVLOD C_R	to load the first image on plane 1.
> GETN 2 C_R	to select the second image.
> TVCH 2 ; TVLOD C_R	to load the second image on plane 2.
> TV2CH 1 ; TVBLINK C_R	to blink planes 1 and 2.

The rate and duty cycle of the blinking, and the transfer functions applied to the planes, are controlled interactively with the TV cursor. Instructions for these operations appear on your terminal while TVBLINK is running.

The task PLAYR provides a menu-driven method to enhance and blink two images and to develop and save TV color tables ("OFMs").

For data cubes (*e.g.*, frequency or time sequences of images), the verbs TVMOVIE and TVCUBE combine the two previous techniques. These are described in more detail in § 8.5.4. Both verbs load one or more image planes with as many planes from the cube as possible (and as requested). Then they display each frame in sequence with interactive controls over the frame rate in movie mode, the chosen frame in single-frame mode, and the brightness, contrast, and color of the displayed images. TVMOVIE makes a somewhat more efficient movie sequence, but TVCUBE makes a better montage by using a more normal arrangement of the image planes. The DOALL adverb allows these verbs to loop over more than just the third axis if the "cube" has more than three multi-pixel dimensions.

Certain real TV displays used to provide powerful tools to compare images using color as well as intensity to represent real information. XAS implements some of these tools on workstations capable of displaying a

virtually unlimited number of colors. Perhaps the most powerful of these is the "hue-intensity" display in which one image controls the intensity of the displayed image and the other controls the hue. A number of uses for this algorithm are obvious, including spectral-line moment images (intensity from total line integral and velocity setting color), polarization images (intensity from the total or polarized intensity image and polarization angle setting color in a circular scheme), and depolarization observations (color set by a two-frequency depolarization image). The best implementation of this algorithm has been in the interactive verb TVHUEINT:

> INDI *d1*; GETN *ctn1* ↵	to select the intensity image.
> TVINIT; TVCORN 0	to clear the TV and restore the default positioning.
> TVCH 1; TVLOD ↵	to load the first image on plane 1.
> INDI *d2*; GETN *ctn2* ↵	to select the hue image.
> TVCH 2 ; TVLOD ↵	to load the second image on plane 2.
> TVCH 1 ; TV2CH 2 ; TVHUEINT ↵	to display a full-color view where the intensity is controlled by image 1 and the hue by image 2 and to interactively adjust that display.

Instructions for altering transfer functions and reversing the roles of the two images appear on your terminal while TVHUEINT is running. You may create a labeled two-direction step wedge to accompany the hue-intensity display with the verb HUEWEDGE.

Until the 31DEC14 release, the only way to capture this image was with TVCPS, which is fundamentally limited by the size and resolution of the TV display. Control over labeling, character font, number of pixels, and the like is severely limited. To overcome these limitations, a new task called HUINT was written. It performs the functions of TVHUEINT with the option of saving the resulting image and a two-dimensional step wedge as *AIPS* cataloged images. The task enters an interactive mode in which the user selects options from a menu shown on a graphics overlay plane. Options are to enhance each of the images individually, to select linear, square-root, or one of 2 logarithmic transfer functions for intensity or hue, and to exit with or without final updates and an output file. The output image(s) can then be made into plot files with standard plot tasks like GREYS and KNTR and then converted into PostScript by LWPLA. HELP POSTSCRIPT contains some guidance on how to combine PostScript renderings of the color image and the color step wedge into a single file. Note that the images saved by HUINT are not in any recognizable physical units. Also note that KNTR should be run with DOCOLOR=1; DOWEDGE=0 FUNCTYPE=' ' and LWPLA should be run with RGBGAMMA=1 and no extra enhancements.

The two functions above require a display that can do an almost unlimited number of colors. Unfortunately, some older workstations can display only 256 simultaneous colors (or even fewer). XAS supports both kinds of workstation. For these limited workstations, there are two tasks, also discussed in § 8.6, which attempt to recover much of this capability by trying to optimize color assignments over the limited range available. The first of these, TVHUI produces a composite display in which the intensity is set by one image, the hue is derived from another image, and the saturation is optionally derived from a third image. An interactive menu allows you to enhance each of the images individually, to select linear, square-root, or logarithmic transfer functions for the intensity and hue images, to select the sub-image used during interactive enhancements, to repaint the full image, and to exit with or without writing out the final three-color image. The task has the option of writing an image of a hue-intensity step-wedge as well. A number of uses for the saturation portion of this task are obvious, including spectral-line moment images (line width setting saturation) and polarization images (polarization intensity setting saturation). TVHUI is also useful on full-color displays, but for such displays, you may do much of the work of this task in HUINT. TVHUI attempts to write out the full range of input intensities scaled by the selected transfer functions. Thus, its output images are different from those of HUINT and may, or may not, be preferable in particular cases.

There is also an *AIPS* task called RGBMP which writes three-color cubes using weighted sums over a data cube. Three-color cubes also arise when digitizing color photographs of real scenes. The second task, TVRGB, can be used to display these three-color cubes or to generate a three-color display from any three *AIPS* image planes. Common examples of the latter are the superposition of radio continuum and/or

line data on optical or X-ray images, and color-coding of effective temperatures or spectral indices from 3-channel continuum data. TVRGB can also be used to color-code different types of depolarization effects from multi-frequency polarimetry. Like TVHUI, TVRGB offers a simple menu to enhance each of the images individually or all together, to select the window specifying the sub-image which is used during interactive enhancements, to repaint the full image on the TV, and to exit. TVRGB does not write an output image per se, but it can be instructed to write out a full 24-bit color PostScript plot file to be sent to a color printer. Its display (or any other TV display including that of TVHUI) can be captured and sent to a color printer; see § 6.4.10 below.

On full-color workstations, three-color images may be displayed by loading each color plane to a separate TV memory. Then each memory is turned on in the desired color only using TVON with the usually ignored COLORS adverb. If the red image is in TV channel 1, the green in 2 and the blue in 3, the verb TV3COLOR is a short-cut for all the parameter setting.

> FOR TVCH=1:3; TBLC(3)=TVCH; TVLOD; END; TV3COLOR C$_R$

In 31DEC25, the procedure TV3LOD was added to make this type of display even more convenient.

6.4.8 Slice files and the TV display

In Chapter 8 we discuss the computation and use of "slices," one-dimensional profiles interpolated along any line in an image plane. Once a slice has been computed, it may be plotted by TVSLICE on the TV display in your choice of graphics channel. A second slice may be plotted on top of the first with TVASLICE. The TV graphics display is used to prepare initial guesses for SLFIT, which fits Gaussians to slices. The verbs involved are:

> NGAUS n ; TVSET C$_R$	to set the number of Gaussians to be fitted to n and then to prepare an initial guess at the parameters by pointing at the peaks and half width points on a graphics plot of the slice.
> TV1SET j C$_R$	to revise the initial guess for the j^{th} Gaussian.
> TVGUESS C$_R$	to plot the initial guess of the model on the TV graphics, erasing any previous plot.
> TVAGUESS C$_R$	to add a plot of the initial guess of the model to the current slice plot on the TV graphics.
> TVMODEL C$_R$	to plot the fit model on the graphics device, erasing any previous plot.
> TVAMODEL C$_R$	to add a plot of the fit model to the current slice plot on the TV graphics.
> TVRESID C$_R$	to plot the data minus the fit model on the TV graphics, erasing any previous plot.
> TVARESID C$_R$	to add a plot of the residuals (data minus model) to the current slice plot on the TV graphics.
> TVCOMPS C$_R$	to plot the individual components including baseline of the fit model on the TV graphics, erasing any previous plot.
> TVACOMPS C$_R$	to add a plot of the individual components including baseline of the fit model to the current slice plot on the TV graphics.

The units for slice model parameters are those of the plot, so it is convenient to set them with these verbs. These same operations may also be done on the Tek graphics device (§ 6.5.2), but modern X-Windows emulations of such devices seem to have problems with cursor reading. The TV verbs allow multiple colors for plot comparisons using different GRCHANs.

6.4.9 Other functions using the TV

There are a number of tasks which use the TV to give the user real interactive input to the operation based on the images displayed by the task on the TV. These include BLANK (§ 8.6) to blank out non-signal portions of an image, BLSUM (§ 8.6) to sum images over irregular blotch regions plotting or printing out summed spectra (and saving SLice files), TVFLG (§ O.1.6) to edit visibility data based on grey-scale displays of some function of the visibility with baseline on the x axis and time on the y axis, SPFLG (§ 8.1, § 10.2.2) to edit visibility data based on grey-scale displays of some function of the visibility with spectral channel for all IFs on the x axis and time on the y axis, EDITR (§ 5.5.2) to edit visibility data based on plots of visibility versus time, 1–11 baselines at a time, EDITA (§ 4.3.11) to edit visibility data based on plots of system temperature (TY or SY tables) or antenna gains (SN or CL tables), WIPER to edit visibility data using UVPLT-like displays, and SNEDT to edit TY, SY, SN, and CL tables themselves. TV-based tasks have appeared to edit visibility data displayed on a $u-v$ grid (UFLAG), to edit visibility or bandpass data based on graphical displays of bandpass solutions (BPEDT), and two tasks to edit pulse-cal tables (PCFLG, PCEDT).

The imaging tasks, IMAGR, SCIMG, and SCMAP display the results of the computation at its current stage on the TV and provide a menu of interactive options to the user. The menu includes the usual display enhancements, the ability to choose among the images being computed, the ability to set Clean windows in those images, and the ability to end the computation at its current stage. The computation will resume when instructed via the menu or after a period of inactivity. A number of older iterative tasks use the TV to display the results of the computation so far and then prompt the user to hit button D within some number of seconds to stop the computation. Tasks that do this include APCLN, SDCLN, VTESS, and several less significant tasks. UVMAP uses the TV simply to draw a picture indicating which cells are sampled in the uv plane. All of these tasks are described in Chapter 5.

The multi-source fitting task SAD was provided with a companion task TVSAD which allows interactive setting of the initial guess parameters for each island being fit. It allows re-trying the fits and adjusting the island boundaries, and can be run non-interactively until the fit finds a problem, after which the interactive mode resumes. *AIPS* Memo 119[3] describes the operation of this task in detail.

There is a set of related tasks for analysis of data cubes transposed so that the first axis is the one on which baselines or Gaussians are to be fit. In the case of a spectral-line cube, the image would be transposed so that velocity is the first axis. TVSPC is an interactive task which displays spectra from one or two transposed cubes at pixels chosen on the TV from some image of the overall field. A third cube may also be displayed with the spectral plane chosen by the cursor on the displayed spectra. See the associated *AIPS* Memo[4]. There is now an additional task TVIEW which does an interactive display of an image cube in all three tranpostions; it also has an *AIPS* Memo[5]. One can also use XPLOT (with DOTV 1) first to get an idea of what the profiles really look like. It uses a flux cutoff to determine which profiles to display and prompts you for permission to continue after each plot. XBASL is used to remove $n^{\rm th}$-order polynomial baselines from each spectrum. It has a batch mode of operation and an interactive mode which uses the graphics display to plot each spectrum and to accept guidance on which channels to use in determining the baselines.

XGAUS fits up to 8 Gaussians plus a linear baseline to each spectral profile. It builds a table of solutions at each pixel and may be re-started until you are satisfied, after which images of the fit Gaussian parameters and their uncertainties may be written to disk. In its interactive mode, it plots each selected spectrum on the TV graphics planes and accepts guidance on the initial guesses for the Gaussians. After all pixels have been done, it displays images of the fit parameters and offers a variety of options for editing and correcting the results. Tasks similar to XGAUS have been written to fit polarization spectra (RMFIT) and to fit for Zeeman splitting (ZEMAN) of the line viewed simply or as a set of Gaussians found by XGAUS. Plot tasks XG2PL and

[3] Greisen, E. W. 2014, "TVSAD: interactive search and destroy" AIPS Memo 119, https://www.aips.nrao.edu/aipsdoc.html
[4] Greisen, E. W. 2025, "Exploring Image Cubes in *AIPS*," AIPS Memo 120 revised, https://www.aips.nrao.edu/aipsdoc.html
[5] Greisen, E. W. 2025, "Further Exploration of Image Cubes in *AIPS*," AIPS Memo 124 revised, https://www.aips.nrao.edu/aipsdoc.html

RM2PL have been written to display these spectral fits in standard *AIPS* plot files. See *AIPS* Memo 118.[6] Corresponding tasks AGAUS and ZAMAN are available to perform similar analysis of absorption-line image cubes. See *AIPS* Memo 122.[7] In 31DEC24, a similar task called TVSPX has appeared to assist in fitting spectral indexes to image cubes. In 31DEC25, another similar task LISPX fits spectral indices to spectra in text files. See *AIPS* Memo 126.[8]

6.4.10 Capturing the TV

Having done all the work to prepare the absolutely perfect display on your TV screen, it would be a good idea to capture it before someone, such as the local power company, does a TVINIT. See § Z.1.2.2 for a discussion of Unix tools to do this. We recommend, however, TVCPS to capture the image on your TV, including graphics overlay channels if they are on, and to write the result to an encapsulated PostScript file. This file can be printed immediately on a black-and-white or color printer or on any other device which understands PostScript. It can also be saved for later printing or inclusion in other documents. TVCPS was used to make the picture on the title page of this *CookBook*, the picture of TVFLG's display in Chapter 4, and the picture of a right ascension - velocity - declination data cube in Chapter 9. TVCPS bases its picture on the current size of your TV display. If you are using a workstation with XAS, be sure to adjust the size of the display window to encompass all of your image plus a modest border. If you leave a large border, you will get a large border in your output. TVCPS understands both pseudo- and full-color XAS displays. Your TVCPS session could look like:

> TASK 'TVCPS' ; INP ↵	to review the inputs.
> OUTFILE 'MYAREA:TV.PIX ↵	to save the output in a file called TV.PIX in an area defined by the logical MYAREA; see § 3.10.1.
> OPCODE 'COL' ↵	to make a color picture.
> APARM 8.5 , 11 ↵	to set the output device size to 8.5 by 11 inches, appropriate to standard quarto paper.
> GO ↵	to run the task when the inputs are set.

TVCPS has an option to add a character string below the image with adverb REASON. If your image is too large to fit on the TV, you can instruct TVCPS to read the image from disk with DOTV = -2 ↵, using adverbs TBLC, TTRC, TXINC, and TYINC. When doing this, you should turn off the graphics display (GROFF 32767 ↵) since it is not aligned properly. This option works when multiple images are visible at the same time, but only when they are in separate TV planes as fully overlapped 3-color images. Some color printers and recorders have rather different transfer characteristics than the workstation screen. TVCPS offers the option to remove the "gamma correction" used for your workstation and to apply a different one appropriate to your color recorder. TVCPS now offers the option to represent colors using the CMYK (cyan-magenta-yellow-black) used in printing rather than the familiar RGB colors system. CMYK displays often require different gamma corrections from those used for RGB. To see the effects of the gamma correction, try the verb GAMMASET in AIPS. You may need to use GWRITE to select better colors for the graphics overlay planes as well. The default black background may be set to and color with adverb RGBCOLOR.

§ Z.1.2.2 contains a discussion of Unix tools to capture the TV screen which work even when the *AIPS* display task has control of the display. The import program is perhaps the more useful of these.

[6] Greisen, E. W. 2025, "Modeling Spectral Cubes in *AIPS*," AIPS Memo 118 revised, https://www.aips.nrao.edu/aipsdoc.html
[7] Greisen, E. W. 2025, "Modeling Absorption-line Cubes in *AIPS*," AIPS Memo 122 revised, https://www.aips.nrao.edu/aipsdoc.html
[8] Greisen, E. W. 2025, "Spectral index fitting in *AIPS*," AIPS Memo 126, https://www.aips.nrao.edu/aipsdoc.html

6.5 Graphics displays of your data

In the dim dark past, Tektronix invented some nice graphical display devices and an inconvenient but functional way to talk to them. This communication language became so embedded in software that workstation vendors now provide X-Windows windows that understand it. *AIPS* also arose from this dim dark past and once upon a time talked to those lovely green screens. To retain the graphics capability, we now provide a TEKSRV server which will provide a Tektronix-like graphics screen on which certain *AIPS* tasks and verbs plot. When AIPS starts up on workstations, it brings up a window called TEKSRV. Leave this window in its iconic state; it is only a marker for the presence of the server. The first time you write to TEKSRV, it will create and open a window called TEKSRV (Tek) in which the plot is done. You can re-size this window within some limits and the plot will automatically re-size itself. When the workstation cursor is in the Tek window, it changes to a diagonal arrow pointer. When an *AIPS* task or verb tries to read from the Tek window, this pointer becomes a plus sign. You should position the pointer to the desired location *without* touching the keyboard or the mouse buttons. When the pointer is exactly where you want it, press any mouse button or any key (*except* RETURN) to return the pointer position to the program. Note that the functions using the graphics display are not quite as friendly as those on the TV. This is due to the inability to erase a piece of a plot without erasing all of it. You can erase the full screen with TKERASE, which will keep the window from redrawing a big plot on every expose event. In recent years, we have found unfortunately that cursor reading from Tektronix emulation screens can be unreliable. Therefore, every TK function described below also has a TV version, described in a previous subsection. The TV has many other advantages over the Tektronix emulator and now is nearly as fast. The verb STARTTV will restart this server if it has, for some reason, exited.

6.5.1 Plotting data and setting values with the graphics display

TKPL interprets *AIPS* plot files to the graphics window or device. The graphics screen can be used to read back data values and set adverbs, much like the TV:

> TKXY ↵	to read the graphics cursor position, setting adverb PIXXY; requires a contour or comparable image to be shown on the screen.
> TKXY ; IMVAL ↵	to read the graphics cursor position and return the image value and coordinates of the selected position.
> TKPOS ↵	to read the graphics cursor position and return the image coordinates of the selected position.
> TKWIN ↵	to read the graphics cursor position twice, first setting BLC and then setting TRC.
> TKBOX(*i*) ↵	to read the graphics cursor position twice, first setting the lower left and then the upper right of Clean box *i*.
> TKNBOXS(*n*) ↵	to set NBOXES to *n* and then set all *n* Clean boxes using the graphics cursor.

All of these verbs require a graphics display of a plot file produced by CNTR, PCNTR, GREYS, or SL2PL. The window procedures don't make much sense with a slice plot, but they will work.

6.5.2 Slice files and the graphics display

In Chapter 8 we discuss the computation and use of "slices," one-dimensional profiles interpolated along any line in an image plane. Once a slice has been computed, it may be plotted by TKSLICE on the graphics display. A second slice may be plotted on top of the first with TKASLICE. The graphics display is used to prepare initial guesses for SLFIT, which fits Gaussians to slices. The verbs involved are:

> TKVAL C$_R$	to return the flux level pointed at by the graphics cursor.
> NGAUS n ; TKSET C$_R$	to set the number of Gaussians to be fitted to n and then to prepare an initial guess at the parameters by pointing at the peaks and half width points on a graphics plot of the slice.
> TK1SET j C$_R$	to revise the initial guess for the j^{th} Gaussian.
> TKGUESS C$_R$	to plot the initial guess of the model on the graphics device, erasing any previous plot.
> TKAGUESS C$_R$	to add a plot of the initial guess of the model to the current slice plot on the graphics device.
> TKMODEL C$_R$	to plot the fit model on the graphics device, erasing any previous plot.
> TKAMODEL C$_R$	to add a plot of the fit model to the current slice plot on the graphics device.
> TKRESID C$_R$	to plot the data minus the fit model on the graphics device, erasing any previous plot.
> TKARESID C$_R$	to add a plot of the residuals (data minus model) to the current slice plot on the graphics device.

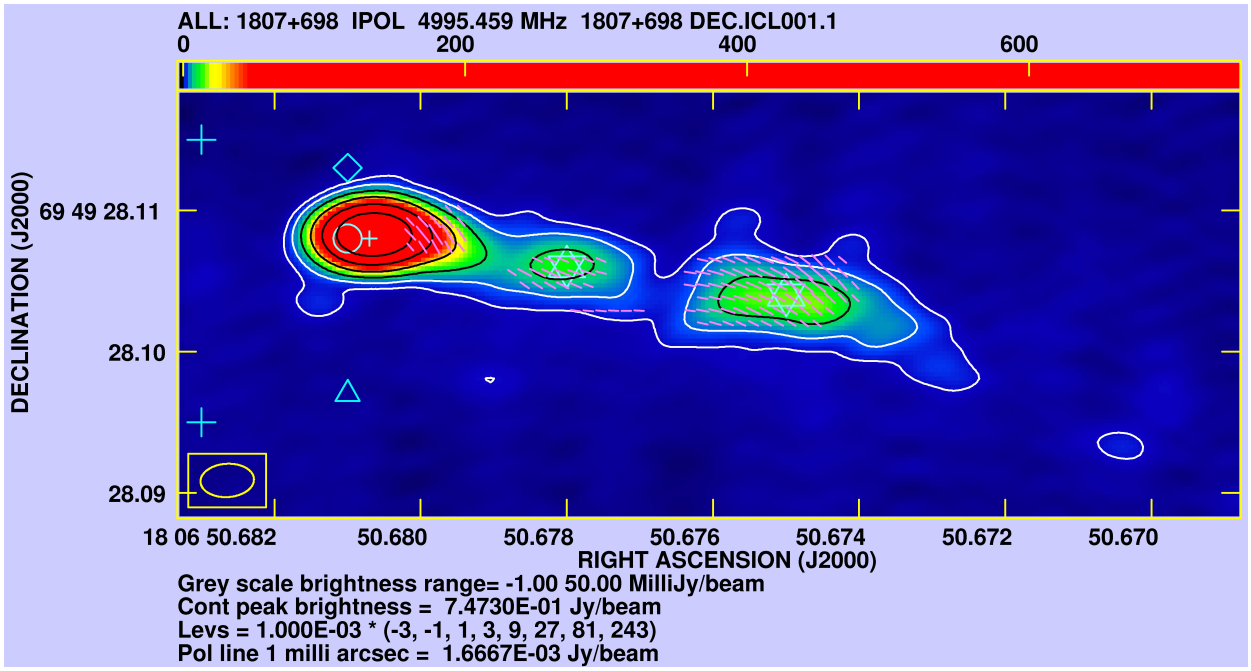
The units for the slice model parameters are fairly problematic, so we recommend using these graphical input and output functions. At least, they all have the same strange ideas. See § 6.4.8 for the verbs that allow this same processing using the TV display.

6.5.3 Data analysis with the graphics display

Tasks XPLOT and XBASL have modes which use the graphics display. They also can use the TV display which is now as fast as the graphics and more reliable.

6.6 Examples of color plotting

The following six pages are in color and should be printed on a color PostScript printer if possible. They will also print on a black and white printer but will then not show what they are supposed to illustrate.

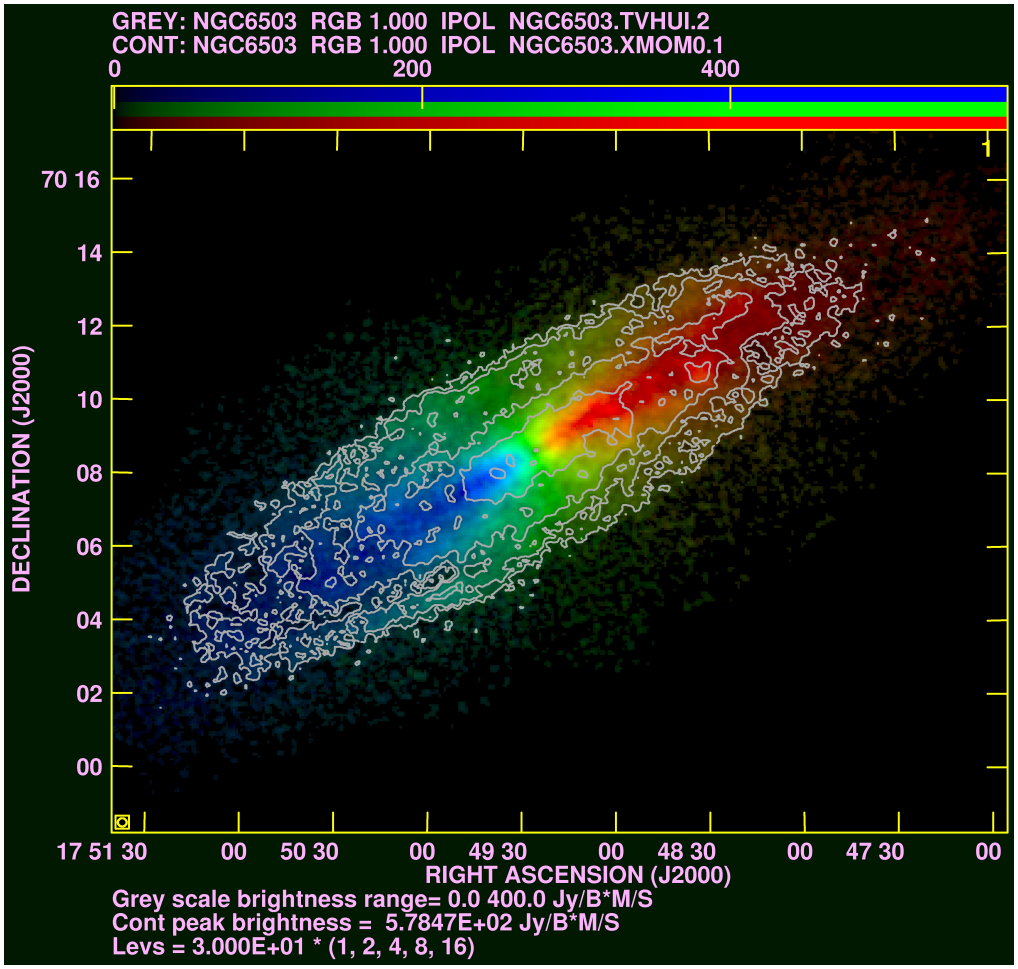



```
KNTR:  Task to generate a plot file for a contour and grey plot
DOCONT        1                    > 0 => do contours
                                   (1 or 2 => which name)
DOGREY        1                    > 0 => do grey scale
                                   (1 pr 2 => which name)
DOVECT        1                    > => do polarization vectors
                                   (1 or 2 => which is IPOL)
                                   Contour or grey or IPOL
INNAME     '1807+698 DEC'          First image name (cube?)
INCLASS    'ICL001'                First image class
INSEQ         1                    First image seq. #
INDISK        3                    First image disk drive #
                                   Polarization intensity image:
IN3NAME    '1807+698 DEC'            (name) blank => INNAME
IN3CLASS   'QCL001'                  (class) blank => 'PPOL'
IN3SEQ        1                      (seq. #) 0 => high
IN3DISK       3                      Disk drive #,  0 => any
                                   Polarization angle image:
IN4NAME    '1807+698 DEC'            (name) blank => INNAME
IN4CLASS   'UCL001'                  (class) blank => 'PANG'
IN4SEQ        1                      (seq. #) 0 => high
IN4DISK       3                      Disk drive #,  0 => any
PIXRANGE   -1.00E-03     0.05      Min,Max of image intensity
FUNCTYPE   'SQ'                    Image intensity transfer func
OFMFILE    '                                  '
LTYPE         3                    Type of labeling: 3 standard
DOALIGN       1                    > 0 => images must line up
CLEV       0.001                   Absolute value for levs
LEVS         -3          -1        Contour levels (up to 30).
              1           3           9          27
             81         243        *rest 0
FACTOR     1000                    Mult. factor for Pol vector
XINC          3                    X-inc. of Pol vectors. 0=>1
YINC          3                    Y-inc. of Pol vectors. 0=>1

PCUT       0.001                   Pol. vector cutoff. P units.
ICUT       0.001                   Int. vector cutoff. I units.
DOWEDGE       3                    = 3 => put on top using full
                                         range of image values
STFACTOR      1                    Scale star sizes: 0 => none.
CBPLOT        1                    Position for beam plot:&
                                     1: lower left (default)
DARKLINE   0.33                    Switch to dark lines when
                                   grey-scale > DARKLINE 0-1

LWPLA:   Sends plot file(s) to a PostScript printer or file
INNAME     '1807+698 DEC'          Image name (name)
INCLASS    'ICL001'                Image name (class)
INSEQ         1                    Image name (seq. #)
INDISK        3                    Disk drive #
PLVER         2                    Version # of PL file. 0=>last
FUNCTYPE   '  '                    'NE','LG','NG', 'SQ', 'NQ'
                                   else linear
DPARM      *all 0                  (1,2) Clip recorded grays
DODARK        1                    Paint dark vectors as "dark"
OFMFILE    'RAINBOW'               Color grey scales....
DOCOLOR       1                    Use PLCOLORS ?
PLCOLORS      1          1         Line, character, background
              0          1         colors - see HELP.
              1          1            1          0.5
              1          0            1          1
              0          0            0          0
              0          0            0          0
              0          0            0          0
              0          0            0          0.8
              0.8        1
```

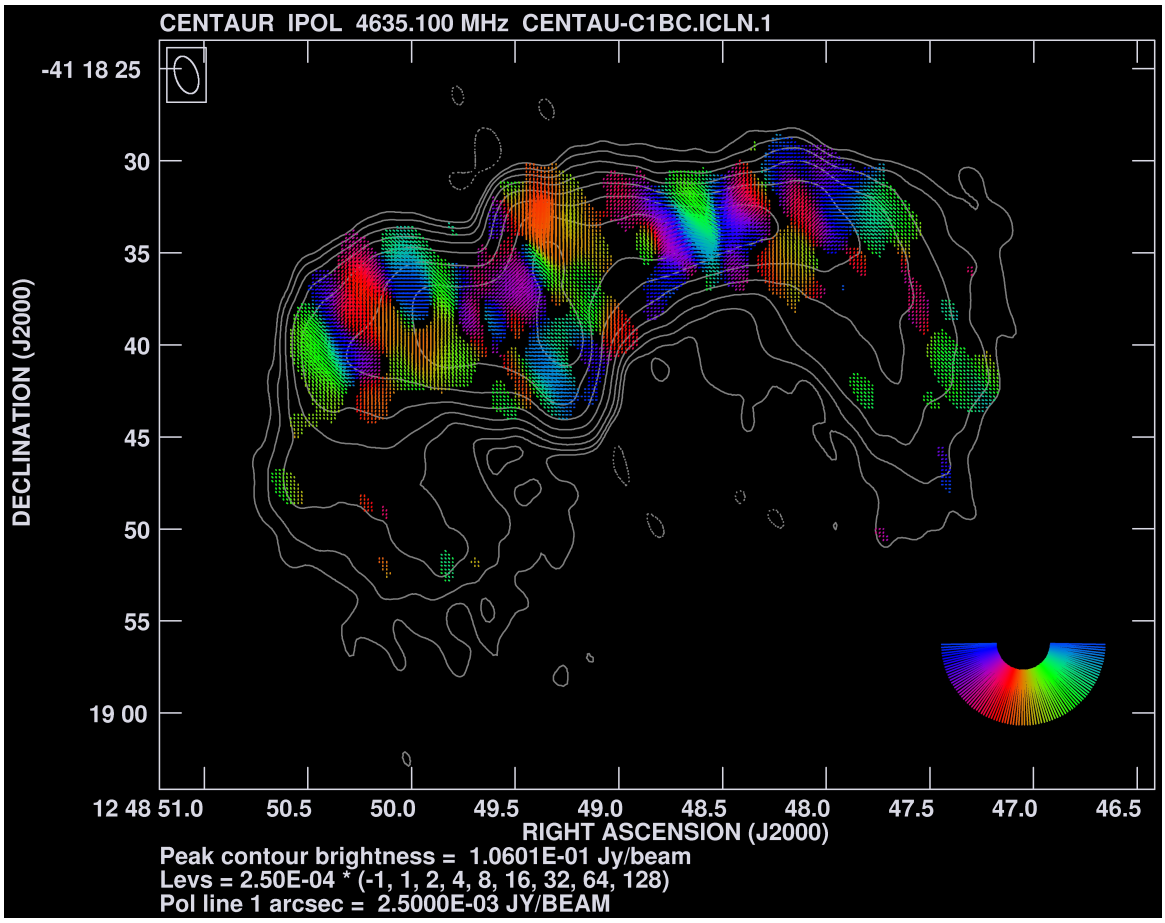
Figure 6.8: KNTR does polarization lines, contours, and grey-scale. Then LWPLA converts the grey-scale to pseudo-color and colors the lines making dark contours dark but dark polarization lines and stars bright. Data courtesy of Greg Taylor.




```
KNTR:  Task to generate a plot file for a contour & grey plot
DOCONT        2                    > 0 => do contours
                                   (1 or 2 => which name)
DOGREY        1                    > 0 => do grey scale
                                   (1 pr 2 => which name)
INNAME     'NGC6503     '         First image name (cube?)
INCLASS    'TVHUI '               First image class
INSEQ         2                   First image seq. #
INDISK        3                    First image disk drive #
                                   Contour or grey or IPOL
IN2NAME    'NGC6503     '         Second image name
IN2CLASS   'XMOM0 '               Second image class
IN2SEQ        1                   Second image seq. #
IN2DISK       3                    Second disk drive #
PIXRANGE      0          400       Min,Max of image intensity
FUNCTYPE   'SQ'                   Image intensity transfer func
                                     'SQ' Square root
DOCOLOR       1                    Do RGB images as 3-color?
LTYPE        -3                    Type of labeling: 3 standard
                                   <0 -> no date/time
DOALIGN       1                    > 0 => images must line up
                                      (see HELP DOALIGN)
DOBLANK      -1                    Draw boundary between blanked
                                   areas and good areas?
DOWEDGE       3                    = 3 => put on top using full
                                          range of image values
CBPLOT        1                    Position for beam plot:
                                     1: lower left (default)
DODARK        1                    Plot dark vectors as black?
DARKLINE      0.33                 Switch to dark lines when
                                   grey-scale > DARKLINE 0-1

LWPLA:   Sends plot file(s) to a PostScript printer or file
INNAME     'NGC6503     '         Image name (name)
INCLASS    'TVHUI '               Image name (class)
INSEQ         2                   Image name (seq. #)
INDISK        3                    Disk drive #
FUNCTYPE   '  '                   'NE','LG','NG', 'SQ', 'NQ'
                                   else linear
DPARM      *all 0                  (1,2) Clip recorded grays
DODARK        1                    Paint dark vectors as "dark"
OFMFILE    *all ' '                Color grey scales....
DOCOLOR       1                    Use PLCOLORS ?
PLCOLORS      1          1         Line, character, background
              0          0.67      colors - see HELP.
              0.67       0.67        0           0
              0          0           0           0
              0          0           0           0.67
              0.67       0.67        0           0
              0          0           0           0
              1          0.7         1           0
              0.1        0
```

Figure 6.9: KNTR interprets the output of TVHUI as a three-color RGB image and overlays moment-0 contours. LWPLA adds coloring to the lines, using a less than pure white for both bright and dark contours so that they are not so dominant. Data courtesy of Eric Greisen, Kristine Spekkens, and Gustaf van Moorsel.

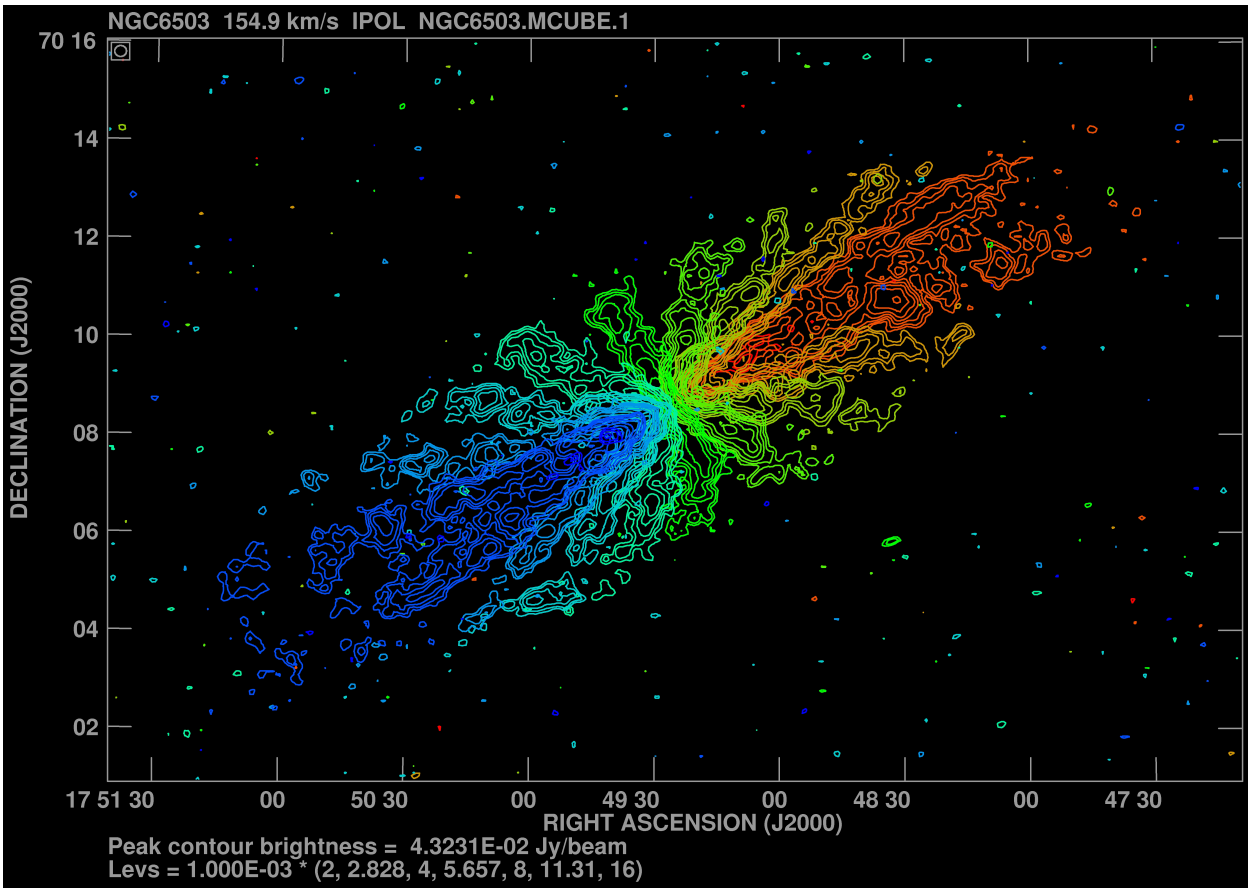



```
PCNTR:  Task to generate plot file for contour plus pol. vectors
DOCONT        1                    Draw contours? > 0 => yes
DOVECT        1                    Draw pol. vectors? > 0 => yes
DOGREY       -1                    Draw grey-scale image?
                                   Total intensity image:
INNAME     'CENTAU-C1BC '            Image name (name)
INCLASS    'ICLN  '                  Image name (class)
INSEQ         1                      Image name (seq. #)
INDISK        1                      Disk unit #
                                   Polarization intensity image:
IN2NAME    'CENTAU-C1BC '            (name) blank => INNAME
IN2CLASS   'QCLN  '                  (class) blank => 'PPOL'
IN2SEQ        1                      (seq. #) 0 => high
IN2DISK       0                      Disk drive #, 0 => any
                                   Polarization angle image:
IN3NAME    'CENTAU-C1BC '            (name) blank => INNAME
IN3CLASS   'UCLN  '                  (class) blank => 'PANG'
IN3SEQ        1                      (seq. #) 0 => high
IN3DISK       0                      Disk drive #,  0 => any
LTYPE        -3                    Type of labeling:
                                   <0 -> no date/time
CLEV       2.500E-04               Absolute value for levs
LEVS         -1          1         Contour levels (up to 30).
              2          4            8          16
             32         64          128        *rest 0
FACTOR     1000                    Mult. factor for Pol vector
                                      (see HELP)

ROTATE        0                    Angle to rotate Pol vector
                                      (in degrees)
XINC          1                    X-inc. of Pol vectors. 0=>1
YINC          1                    Y-inc. of Pol vectors. 0=>1
PCUT      1.250E-04                Pol. vector cutoff. P units.
ICUT      2.500E-04                Int. vector cutoff. I units.
POL3COL     160                    Color polarization vectors
                                   value in degrees = red
CBPLOT        4                    Position for beam plot:
                                     4: upper left

LWPLA:   Sends plot file(s) to a PostScript printer or file
RGBGAMMA      1          1         Gamma correction to apply
              2.5
DPARM      *all 0                  (1,2) Clip recorded grays
                                      before FUNCTYPE (0 to 1)
DOCOLOR       1                    Use PLCOLORS ?
PLCOLORS      0.85       0.85      Line, character, background
              0.9        0.5       colors - see HELP.
              0.5        0.5          0          0
              0          0            0          0
              0          0            0          0
              0          0            0          0
              0          0            0          0
              0.85       0.85         0.9      *rest 0
```

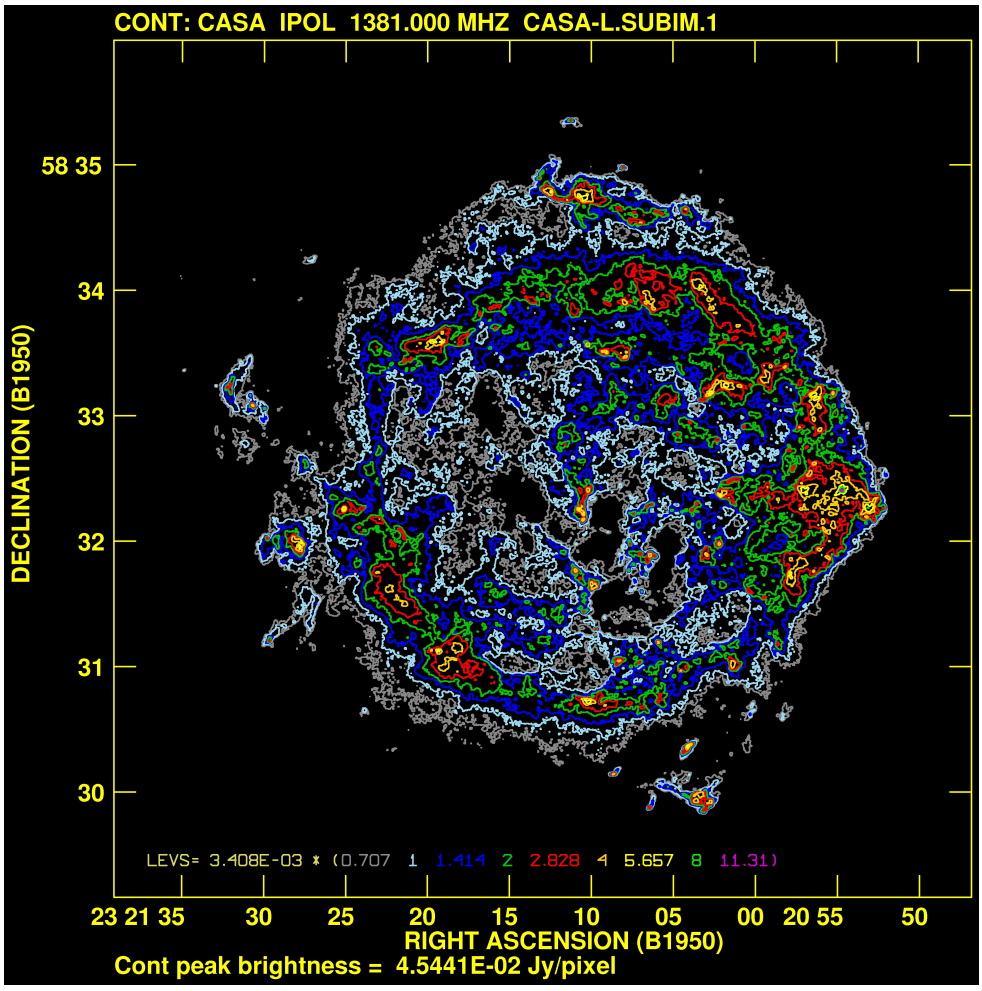
Figure 6.10: PCNTR plots contours and polarization vectors of Centaurus A. Color is used to show the complex changes in polarization position angle since the angles of short lines cannot be seen accurately. Data courtesy of Greg Taylor. For a discussion of this amazing pattern see Taylor, G.B., Fabian, A.C., & Allen, S.W. 2002, MNRAS, 334, 769, astro-ph/0109337 "Magnetic Fields in the Centaurus Cluster."




```
PCNTR:  Task to generate plot file for contour plus pol. vectors
DOCONT       1                  Draw contours? > 0 => yes
DOVECT      -1                  Draw pol. vectors? > 0 => yes
BLC          51        167     Bottom left corner of images
             39       *rest 0
TRC         397        393     Top right corner of images
             89       *rest 0
CON3COL      5                  > 0 => overplot contours in
                                color of multiple planes
                                ZINC is CON3COL.
CLEV       0.001                Absolute value for levs
LEVS         2          2.8284 Contour levels (up to 30).
             4          5.6569     8          11.3137
            16       *rest 0
CBPLOT       4                  Position for beam plot:
                                  4: upper left
```

```
LWPLA:   Sends plot file(s) to a PostScript printer or file
LPEN         3                  Pen width (dots).
RGBGAMMA     1          1       Gamma correction to apply
             2.5
DPARM      *all 0               (1,2) Clip recorded grays
DOCOLOR      1                  Use PLCOLORS ?
PLCOLORS     0.6        0.6     Line, character, background
             0.6        0.06275 colors - see HELP.
             1          0          1          0.6706
             1          0          1          1
             0          0          0          0
             0          0          0          0
             0          0          0          0
             0.6        0.6        0.6      *rest 0
```

Figure 6.11: PCNTR plots contours every fifth plane from a data cube using colors related to the velocity. LWPLA adds coloring to the labeling and background and applies a gamma correction to blue. Data courtesy of Eric Greisen, Kristine Spekkens, and Gustaaf van Moorsel.

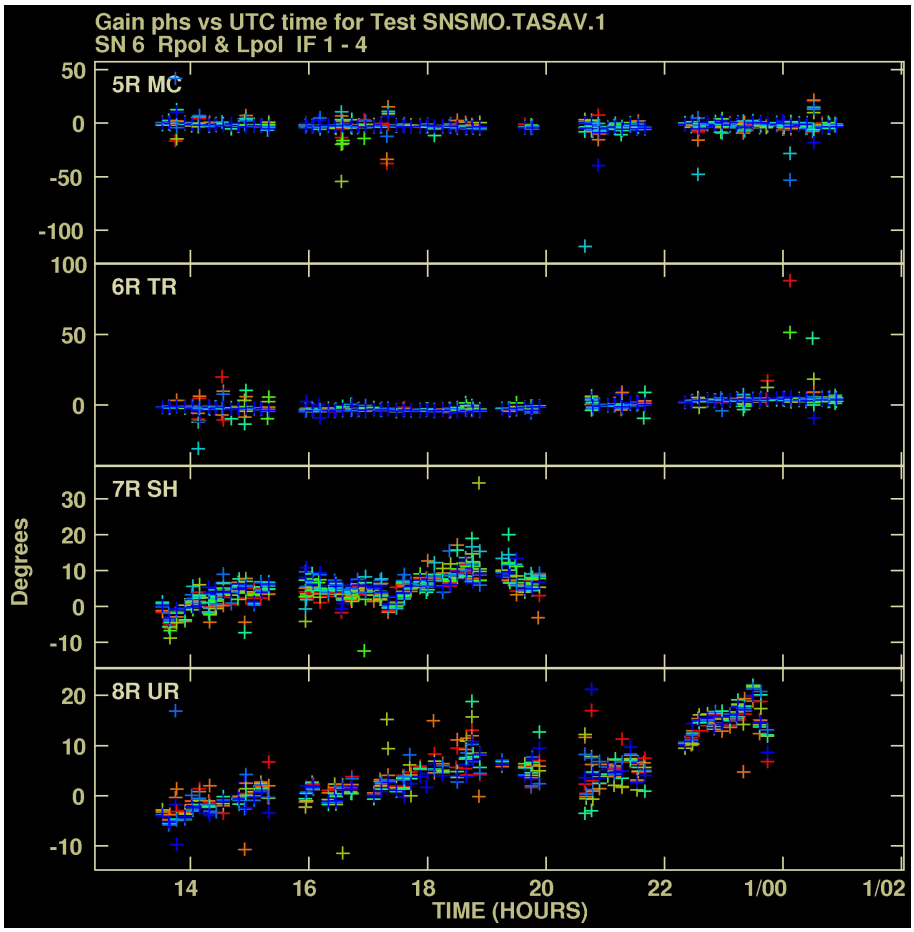



```
KNTR:  Task to generate a plot file for a contour & grey plot
DOCONT        1                    > 0 => do contours
                                   (1 or 2 => which name)
DOGREY       -1                    > 0 => do grey scale
                                   (1 pr 2 => which name)
DOVECT       -1                    > => do polarization vectors
                                   (1 or 2 => which is IPOL)
LTYPE        -3                    Type of labeling: 1 border,
                                   2 no ticks, 3 standard, 4 rel
                                   to center, 5 rel to subim cen
                                   6 pixels, 7-10 as 3-6 with
                                   only tick labels
                                   <0 -> no date/time
                                   special values for RGBLEVS
CLEV        0.003408               Absolute value for levs
                                      (used only if PLEV = 0).
LEVS          0.7071      1        Contour levels (up to 30).
              1.4142      2           2.8284      4
              5.6569      8          11.3137   *rest 0
CON3COL       0                    Color the contours by plane

RGBLEVS       0.2562      0.2562   Color each value of LEVS
              0.2562      0.3511      0.7297      0.9035
              0           0           1           0
              0.6205      0           1           0
              0           1           0.6205      0
              1           1           0           0
              0.8503      0           0.6594      0
              0.6594   *rest 0

LWPLA:   Sends plot file(s) to a PostScript printer or file
LPEN          3                    Pen width (dots).
RGBGAMMA      2.2         2.2      Gamma correction to apply
              2.2
OFMFILE    *all ' '                Color grey scales....
DOCOLOR       1                    Use PLCOLORS ?
PLCOLORS      1           1        Line, character, background
              0         0.06275    colors - see HELP.
              1           0           1           0.6706
              1           0           1           1
              0           0           0           0
              0           0           0           0
              0           0           0           0
              1           1        *rest 0
```

Figure 6.12: KNTR plots contours of Cassiopeia A with each contour level separately colored under control of adverb RGBLEVS. The values of RGBLEVS were set by a procedure call STEPLEVS(10) made available by RUN SETRGBL. Image is from the *Images from the Radio Universe* CD, 1992, NRAO with the particular image from Anderson M., Rudnick, L., Leppik, P, Perley, R. & Braun, R. 1991, ApJ, 373, 146.




```
SNPLT:  Task to plot selected contents of SN, TY, PC or CL file.
INEXT      'SN'                 Input 'SN','TY','PC', 'CL'
INVERS        6                 Input table file version no.
STOKES     '    '               Stokes type to plot: R, L,
                                    RR, LL, RRLL, DIFF, RATO
BIF           0                 First IF to plot, 0=>1.
EIF           0                 Last IF to plot 0 -> highest
ANTENNAS      5          6      Antennas to plot 0=>all
              7          8      *rest 0
NPLOTS        4                 Number of plots per page
XINC          1                 Plot every XINC'th point
OPTYPE     'PHAS'               Data to be plotted:
                                'PHAS','AMP ','DELA','MDEL',
                                'RATE','TSYS','TANT','ATM ',
                                'GEO ','DOPL','SNR ','SUM '
                                'CCAL', 'DDLY', ' '=phas.
OPCODE     'ALSI'               Type of plot:
                                'IFDF' => diff BIF and EIF
                                'IFRA' => ratio BIF and EIF
                                'ALIF' => combine all IFs
                                'ALST' => combine all Stokes
                                'ALSI' => all IFs & Stokes
DO3COL        1                 > 0 use 3-color symbols for
                                ALIF, ALST, ALSI OPCODEs and
                                SUM OPTYPE.
XAXIS         0                 Variable data is to be
                                plotted against, 0=>time.
FACTOR        1.5               Scale plot symbols by FACTOR
                                0 -> 1
LTYPE        -3                 Type of labeling: 1 border,
                                2 no ticks, 3 - 6 standard,
                                7 - 10 only tick labels
                                <0 -> no date/time

LWPLA:  Sends plot file(s) to a PostScript printer or file
PLVER         0                 Version # of PL file. 0=>last
INVERS        0                 PL file version #, upper
                                limit if > PLVER
ASPMM         0                 Arc sec. per mm. 0=self scale
LPEN          3                 Pen width (dots).
RGBGAMMA      1          1      Gamma correction to apply
              2.5
DPARM      *all 0               (1,2) Clip recorded grays
                                    before FUNCTYPE (0 to 1)
DOCOLOR       1                 Use PLCOLORS ?
PLCOLORS      0.85       0.85   Line, character, background
              0.65       0.6    colors - see HELP.
              0.6        0.6       0          0
              0          0         0          0
              0.75       0.75      0.5        0
              0          0         0          0
              0          0         0          0
              0.75       0.75      0.5     *rest 0
```

Figure 6.13: SNPLT plots phases for four antennas with color indicating polarization and IF channel. Stokes 1, IF BIF is pure red changing through yellow, green, and cyan to Stokes 2, IF EIF as pure blue. When all symbols lie on top of each other, the last one (pure blue) will dominate.

7 Analyzing Images

In order to obtain useful astronomical information from the data, software exists for the analysis of images, combining of images, estimating of errors, etc. Only a few of the programs are described in any detail here; the others should be self-explanatory using the HELP and INPUTS files for the tasks listed in Chapter 13. A complete list of software in $\mathcal{AIPS}$ for the analysis of images may also be obtained at your terminal by typing ABOUT ANALYSIS ↵.

7.1 Combining two images (COMB)

The task COMB is a general purpose program for combining two images, pixel by pixel, to obtain a third image. Many options are available and, as a first example, we illustrate inputs to subtract a continuum image from a spectral line image cube.

7.1.1 Subtracting a continuum image from an image cube

A common method to obtain a spectral data cube containing only line signal without any continuum emission is to create a line-free continuum image C, and subtract it from the data cube L. For a more general discussion and alternative methods see §8.3. COMB can be used to this purpose as follows:

> TASK 'COMB' ; INP ↵	to review the required inputs.
> INDI 0 ; MCAT ↵	to help you find the catalog numbers of C and L.
> INDI *n1* ; GETN *ctn1* ↵	to select the L image cube from disk *n1* catalog slot *ctn1*.
> IN2D *n2* ; GET2N *ctn2* ↵	to select the C image from disk *n2* catalog slot *ctn2*.
> OUTN '*xxxxx*' ↵	to specify *xxxxx* for the name of the continuum-free image cube.
> OUTC '*ccc*' ↵	to specify *ccc* for the class of the continuum-free image cube, *e.g.*, LCUBE.
> OPCODE 'SUM' ↵	to select the addition algorithm.
> APARM 1, -1 ↵	to specify that we want $+1 \times L - 1 \times C$.
> GO ↵	to compute the continuum-free, line-only output cube.

Once COMB task has terminated with the message COMB: APPEARS TO END SUCCESSFULLY, you should find the requested image in your catalog:

> MCAT ↵	to list the images in your catalog.

7.1.2 Polarized intensity and position angle images

As a second example, we derive the polarization intensity and angle from the Q and U Stokes parameter images. To compute a polarized intensity image, enter:

> TASK 'COMB' ; INP ↵	to review the required inputs.
> INDI 0 ; MCAT ↵	to find the catalog numbers of the Q and U images.
> INDI *n1* ; GETN *ctn1* ↵	to select the Q image from disk *n1* catalog slot *ctn1*.
> IN2D *n2* ; GET2N *ctn2* ↵	to select the U image from disk *n2* catalog slot *ctn2*.

> OUTN *'xxxxx'* ↵ to specify *xxxxx* for the name of the polarized intensity image.
> OUTC *'ccc'* ↵ to specify *ccc* for the class of the polarized intensity image, *e.g.*, PCLN.
> OPCODE 'POLC' ↵ to select the $\sqrt{Q^2+U^2}$ algorithm with correction for noise.
> BPARM *ns1* , *ns2* ↵ to specify the noise levels of the 2 images.
> GO ↵ to compute the corrected, polarized intensity image.

AIPS will write the message TASK COMB BEGINS followed by a listing of the POL. INTENSITY algorithm. While it is running, you can prepare the inputs to make a polarization position angle image. Type:

> OUTN *'yyyyy'* ↵ to specify *yyyyy* for the name of the polarization angle image.
> OUTC *'ddd'* ↵ to specify *ddd* for the class of the polarization angle image, *e.g.*, PSIMAP, CHICLN.
> OPCODE 'POLA' ↵ to select the $\frac{1}{2}\tan^{-1}\left(\frac{U}{Q}\right)$ algorithm.

Once COMB has finished, enter:

> GO ↵ to compute the polarization angle image.

7.1.3 Other image combination options

COMB may also be used to re-scale images, and to compute spectral indices, optical depths, etc. Type:

> HELP COMB ↵ to review the available options.

The OPCODE options are:

'SUM '	Addition	$a_1M_1 + a_2M_2 + a_3$	
'SUMM'	Addition	$a_1M_1 + a_2M_2 + a_3$	except blanked pixels replaced with 0
'MEAN'	Average	$a_1M_1 + a_2M_2$	except M_j where M_i blanked
'MULT'	Multiplication	$a_1M_1M_2 + a_2$	
'DIV '	Division	$a_1M_1/M_2 + a_2$	
'SPIX'	Spectral Index	$a_1\ln(M_1/M_2)\,/\,\ln(\nu_1/\nu_2) + a_2$	where $M_1 > a_3$ and $M_2 > a_4$
'OPTD'	Opacity	$a_1\ln(a_3M_1/M_2 + a_4) + a_2$	where $M_1 > a_5$ and $M_2 > a_6$
'POLI'	RMS sum	$a_1\sqrt{M_1^2 + M_2^2} + a_2$	
'POLC'	RMS sum	$a_1C(M_1, M_2)\sqrt{M_1^2 + M_2^2} + a_2$ where C is a noise-based correction for Ricean bias	
'POLA'	Arctangent	$a_1\tan^{-1}(M_2/M_1) + a_2$	where $\sqrt{M_1^2 + M_2^2} > a_3$
'REAL'	Real part	$a_1M_1\cos(a_2M_2) + a_3$	
'IMAG'	Imaginary part	$a_1M_1\sin(a_2M_2) + a_3$	
'RM '	Rotation measure	$a_1RM(M_1, M_2) + a_2$	
'CLIP'	Clipping	M_1	except blanked where $a_1 > M_2 > a_2$ or $a_2 > a_1 > M_2$ or $M_2 > a_2 > a_1$

where the a_i are user-adjustable parameters — specified by APARM — and M_1 and M_2 are the images selected by INNAME, *etc.* and by IN2NAME, *etc.*, respectively. COMB may be instructed to write an image of the estimated noise in the combination in addition to the direct result of the combination. These noise images may be used as inputs to COMB and, *e.g.*,BLANK to control later computations. When combining numerous images together, set DOHIST=-2 to prevent history files from becoming much larger than the images.

CONVL can compute the cross-correlation of two images and report the location of the peak cross-correlation.

7.1.4 Considerations in image combination

COMB can use images of the uncertainties in the first two input images to control the computation of the output. The new task RMSD may be used to create an image of the rms in an image, computing the rms self-consistently (robust, histogram, and median absolute deviation methods available) in windows surrounding each pixel. Task FLATN can compute weight and noise images corresponding to mosaiced images.

For some applications of COMB, undefined pixel values may occur. For example, if the spectral index is being calculated and the intensity level on either image is negative, the index is undefined. In this case, the pixel value is given a number which is interpreted as undefined or "blanked." Blanking also arises naturally in operations of division, opacity, polarization angle, and clipping and, of course, the input images may themselves be blanked. In addition, the output image can be blanked (set BPARM(4) = 0) whenever either M_1 <APARM(9) or M_2 <APARM(10). Alternatively, blanking may be done on the basis of the estimated noise (set BPARM(4) = 1) or signal-to-noise ratio (set BPARM(4) = 2) in the combination. See HELP COMB ↵ for a description of these options and certain limitations in their use. With APARM(8) = 1 ↵, the user may specify that all undefined pixels are to be assigned an apparently valid value of zero, rather than the "magic" undefined-pixel value. Alternatively, the task REMAG can be used to replace blanked pixels in the output image with a user-specified value.

When combining two or more images, COMB, PCNTR, *et al.* must decide which pixels in the $2^{\rm nd}$ image go with which pixels in the $1^{\rm st}$ image. The user input parameter DOALIGN controls this process. A value of 1 requires the two headers to be correct and sufficiently similar that an alignment by coordinate value is possible. A value of -2 tells the programs to ignore the headers and align by pixel number. Enter HELP DOALIGN ↵ for details and intermediate options. In some cases, the images may have been created on different grids which are correctly described in the headers. The observations, for example, could have differed in the phase reference position or projective geometry used or the imaging could have been done with different axis increments. Such images should *not* be combined directly. Instead, the header of one should be used as a template for re-gridding the other. tasks HGEOM and OHGEO provide this service with good interpolation methods. See § 7.6.1 and type EXPLAIN HGEOM ↵ or EXPLAIN OHGEO ↵ for more information.

7.2 Combining more than two images (CMPLX, SUMIM, SPIXR, STACK)

The task CMPLX is new in 31DEC21. For pairs of images which represent a complex quantity (either real and imaginary or amplitude and phase), CMPLX will perform various complex mathematical operations on one such pair (conjugate or swap the type) or combining two such pairs (various sums, multiplications, and divisions).

The task SUMIM is used to sum or average any number of images. Since *AIPS* has only a limited number of adverbs of the kind INNAME, IN2NAME, etc., SUMIM requires that all input images have identical INNAME and INCLASS. The input images are then specified by INSEQ (the sequence number of the *first* input image), IN2SEQ (the sequence number of the *last* input image), and IN3SEQ (the increment in sequence number). All input images have to reside on the *same* disk.

> TASK 'SUMIM' ; INP ↵	to review the required inputs.
> INDI 0 ; MCAT ↵	to help you find the catalog number of the first input image.
> INDI *n* ; GETN *ctn* ↵	to select the first input image from disk *n* catalog slot *ctn*.
> IN2SEQ *s* ↵	to specify the sequence number of the last image to be included.
> IN3SEQ 0	to specify the increment in sequence number (= 1).
> OUTN '*xxxxx*' ↵	to specify *xxxxx* for the name of the output image.

> OUTC *'ccc'* ↵	to specify *ccc* for the class of the output image.
> FACTOR f ↵	to specify the factor with which to multiply each image before adding. $f = 1$ leads to summation, $f = 0$ defaults to the inverse of the number of input images (average)
> GO ↵	to start SUMIM.

This is a very noisy way to make a line-sum image. For more serious work, use BLANK (§ 7.4) and XMOM (§ 8.6) instead.

A popular analysis technique has been developed in which images centered upon objects of some chosen kind are simply added up even though the objects' emission is less than the noise. The belief is that, if there is real emission just below detectability in individual images, the summed image will reduce the noise and reveal the weak emission. The task STACK attempts such an analysis doing a weighted mean or median of a matched set of images ignoring the coordinates.

The task SPIXR is intended to fit spectral indexes optionally including curvature to a cube of image planes. This cube is build with FQUBE to make an FQID axis out of irregularly spaced frequencies. The cube is transposed by TRANS to put the FQID or, if build with MCUBE, frequency axis first. Then SPIXR will do a least squares fit for spectral index. Again, the results will be noisy unless the initial images have been blanked and converted to similar spatial resolution. The DPARM and CPARM options in SPIXR may also be used to blank pixels with bad solutions. An interactive version of SPIXR called TVSPX has appeared in 31DEC24. It allows you to interact with the outputs to attempt to get better solutions by omitting selected channels. In 31DEC25, a similar task called LISPX fits spectral indices to spectra in text files rather than image files. The new task IM2TX writes such text files with the sum, average, or rms over a region in an image cube while TX2IM reads such text files to create a one-dimensional image on *AIPS* disk. Such images play nicely with SLICE. SLFIT, and all the slice display verbs. Another new task SPXMD creates model image cubes with spectral indices. See *AIPS* Memo 126.[1] The task SPFIX may be used to construct a model image cube based on the output of SPIXR. In 31DEC24, UNSPX may be used to correct the input cube (or, for example, a matching polarization cube) for the spectral index.

7.3 Image statistics and flux integration

The task IMEAN is used to determine the statistics of the image inside, or outside a specified rectangular or circular area. It derives the minimum and maximum value and location, the rms, the average value and, if the image has been Cleaned, an approximate flux density within the area. A typical run might be:

> TASK 'IMEAN' ; INP ↵	to list the input parameters.
> INDI n ; GETN ctn ↵	to select the image file from disk n catalog slot ctn.
> BLC $n1$, $n2$; TRC $m1$, $m2$ ↵	to set the window from ($n1$,$n2$) to ($m1$,$m2$) — or use TVWIN with the cursor on the TV.
> DOHIST TRUE ↵	to make a plot file of the pixel histogram.
> PIXRANGE $x1$, $x2$ ↵	to set the range of the histogram from $x1$ to $x2$.
> NBOXES n ↵	to set the number of boxes in the histogram.
> DOALL 1 ↵	to include pure 0 pixels in the statistics, a 31DEC24 option. Otherwise they are ignored.
> GO ↵	to run the task.

A circular aperture may be specified with BLC = -1,$radius$; TRC = X_c, Y_c. IMEAN attempts to determine the true noise of the image by fitting the peak of the histogram and reports both that result and the one found by including all pixels within the window. IMEAN now gets the initial guess for the true noise from a

[1] Greisen, E. W. 2025, "Spectral index fitting in *AIPS*," AIPS Memo 126, https://www.aips.nrao.edu/aipsdoc.html

robust computation over the input window and returns the adverbs PIXSTD and PIXAVG from the histogram fit to the AIPS program. IMEAN now fills an additional output adverb such that TRIANGLE(i) is the brightness level exceeded by only i per cent of the image.

The statistics will appear in the *AIPS* window. For a hard copy type:

> PRTASK 'IMEAN' ; PRTMSG ↵ with PRIO ≤ 5.

To see the histogram of the intensities, an example of which is shown in § 6.3.2.3, type one of:

> GO TKPL ↵ to display the histogram in the Tektronix window.

> GO LWPLA ↵ to display the histogram on a PostScript printer.

The verbs TVSTAT and IMSTAT provide similar functions to IMEAN without the histogram and true rms options. Both return their results as AIPS parameters PIXAVG (mean), PIXSTD (rms), PIXVAL (maximum), PIXXY (pixel position of the maximum), PIX2VAL (minimum), PIX2XY (pixel position of the minimum). IMSTAT uses the same file name, BLC, and TRC parameters as IMEAN including the circular aperture convention. It is useful to prepare the initial rms guess for that task although the PIXSTD it returns will often be a factor of several too large. TVSTAT, however, works on the image plane currently displayed on the TV and is not limited to a single rectangular area. Instead, the TV cursor is used to mark one or more polygonal regions over which the function is to be performed. Type EXPLAIN TVSTAT ↵ for a description of its operation.

The interactive task BLSUM employs a method similar to that of TVSTAT. The TV cursor is used to mark a region of interest in a "blotch" image. Then BLSUM finds the flux in that region not only in the blotch image but also in each plane (separately) of a second image. More than one region of interest may be done in any given execution of the task. In spectral-line problems, the blotch image is often the continuum or the line sum while the second image is the full "cube" in almost any transposition. The spectrum obtained may be saved as a SLice file for further analysis and display. The spectra may be saved as true plot files as well as printer plots and the flux summation may be weighted by the values (or their squares) found in the blotch image. Numerous continuum applications also exist (*e.g.*, polarization, comparison across frequency). Type EXPLAIN BLSUM ↵ for a description of the operation.

The verb IMDIST is used to measure the angular distance and position angle between two pixel positions in up to two images. The separation is returned as adverb DIST. Verb TVDIST allows you to select the two pixels interactively from the TV display.

The verb IMCENTER may be used to determine the intensity-weighted centroid of a rectangular or circular portion of an image. The verb returns adverbs PIXXY, COORDINA, and ERROR giving the pixel and physical coordinates of the centroid and an indicator of success or failure.

7.4 Blanking of images

In order to determine accurate flux values in images, or moments of velocity profiles, it is desirable to restrict the integrations to pixels that contain emission, or, in other words, to exclude pixels that contribute only noise. If this is not done, the inclusion of noisy pixels will increase the rms in the derived integrated value to an unacceptable extent. The task BLANK gives the user the opportunity to replace pixels containing pure noise with values that *AIPS* and its tasks interpret as *undefined*. The decision whether a certain pixel contains pure noise, or carries some emission, can be made subjectively (using the TV) or in a more objective fashion (see below for an example). In all cases, BLANK creates an output image which is a copy of the input image with some pixels replaced by undefined values, or — if the user specifies it — by the value zero.

The most straightforward use of BLANK is to apply a cutoff to the input image, e.g. let BLANK replace with an undefined value every pixel in the input image that lies below a specified. *e.g.*, 3σ noise level. This

effectively removes almost all noisy pixels. The disadvantage is that this method also removes any *signal* below the 3σ noise level. Since a substantial fraction of the total flux may be "hidden" in pixels below 3σ, this method prevents an accurate total flux determination. Another straightforward use of BLANK is to remove all pixels outside a user-specified radius. This allows blanking regions for which the primary-beam corrections, and hence the noise levels, are large. A new OPCODE = 'BOX' lets one simply blank all pixels inside or outside a list of up to 10 circular or rectangular boxes.

A better way to perform the blanking is one which is not based on the pixel values in the input image itself, but on those in a *second* input image. Typically this is a convolved (spatially and/or in velocity) version of the input image, which has a higher signal to noise for extended emission than the input image. In the example given here we have the input image I_1 of full spatial resolution, and a convolved version of this input image I_2 with a linear beam size roughly twice full resolution. Careful inspection of this second image has shown that there are no outlying noise peaks above *f* mJy/beam. BLANK is then run as follows:

> TASK 'BLANK' ; INP ↵	to review the required inputs.
> INDI 0 ; MCAT ↵	to help you find the catalog numbers of I_1 and I_2.
> INDI *n1* ; GETN *ctn1* ↵	to select I_1 from disk *n1* catalog slot *ctn1*.
> IN2D *n2* ; GET2N *ctn2* ↵	to select I_2 from disk *n2* catalog slot *ctn2*.
> OUTN '*xxxxx*' ↵	to specify *xxxxx* for the name of the blanked output image.
> OUTC '*ccc*' ↵	to specify *ccc* for the class of the blanked output image.
> OPCODE 'IN2C' ↵	to specify that the blanking is performed using pixel values in a *second* input image.
> DPARM(3) $sim\ -f\ ,\ f$ ↵	to set DPARM(3) and DPARM(4) to specify that all pixels with fluxes in the second input image in the interval *(-f,f)* should be blanked.
> GO ↵	to compute the blanked output image.

The task REMAG can be used to replace blanked pixels by a value to be specified by the user.

The *AIPS* TV display may be used to do a more subjective blanking with this task. Set OPCODE 'TVCU' ↵ to display the image, one plane at a time in any transposition. You will be prompted to set "blotch" regions (much like TVSTAT and BLSUM) to define the areas to be blanked. This is one method for having different regions of signal at different spectral channels. There are also four windowing methods for blanking spectral-line cubes which have been transposed to have the frequency axis be first. In these methods, a window (range of spectral channels) about the peak signal in each spectrum is retained.

The task RMSD may be used to write a version of the input image blanking pixels below N times the rms in the image, computing the rms self-consistently in windows surrounding each pixel.

7.5 Fitting of images

There are three tasks and two verbs which estimate the position and intensity of a component on a two-dimensional image. The simplest and fastest methods are the verb IMCENTER and MAXFIT. The latter fits a two-dimensional parabola to the maximum within a few pixels of an image position, and gives the peak and its position. The tasks IMFIT and JMFIT are similar and fit an image subsection with up to four Gaussian components with error estimates. Tasks SAD and TVSAD attempt to automate the process of finding and fitting Gaussian components in an image. Additionally, in one dimension, the task SLFIT fits Gaussian components to slice data and the task XGAUS fits Gaussian components to each row of an image cube.

7.5.1 Centroid fits (IMCENTER) and parabolic fit to maximum (MAXFIT)

You may determine a centroid for a region in an image with the verb IMCENTER. Set the name parameters for the desired image and then define the region with adverbs BLC and TRC, perhaps using TVLOAD; TVWIN ↵. Set FLUX if you wish to limit the computations to pixel values greater than FLUX. Then

> IMCENTER; IMVAL ↵ to find the value at the centroid.

MAXFIT's speed makes it useful for simple regions. Type:

> EXPLAIN MAXFIT ↵ to get a good explanation of the algorithm.

The inputs should be self-explanatory. The IMSIZE parameter can be important in crowded fields. MAXFIT can be used conveniently by first displaying the image on the TV and then typing:

> IMXY ; MAXFIT ↵

First the cursor will appear on the TV. Move it close to a maximum, press the left mouse button, and hit button A, B, C, or D. The fit will appear in your *AIPS* window. Adverb values PIXXY, PIXVAL, COORDINA, and ERROR will be set appropriately. Adverb FSHIFT will also be set to suggest the shift needed to place the source directly on a pixel.

7.5.2 Two-dimensional Gaussian fitting (IMFIT, JMFIT)

A more sophisticated least-squares fit of an image is obtained with IMFIT, which fits an image with up to four Gaussian components and attempts to derive error estimates. A linear or curved, two-dimensional "baseline" may also be fitted. A sample set-up is as follows:

> TASK 'IMFIT' ; INP ↵ to list the input parameters.

> INDI *n* ; GETN *ctn* ↵ to select the image from disk *n* catalog slot *ctn*.

> BLC *n1*, *n2* ; TRC *m1*, *m2* ↵ to set the area to be fitted as (*n1*,*n2*) to (*m1*,*m2*) — or use TVWIN with the cursor on the TV.

> NGAUSS 2 ↵ to set the number of components to be fitted to 2.

> CTYPE 1, 1 ↵ to have both components be Gaussians.

> GMAX 0.34 0.20 ↵ to give estimates of peak intensity in Jy.

> GPOS 200, 100, 210, 110 ↵ to give estimates of the pixel locations of each component.

> GWID 6 4 20 6 4 20 ↵ to give estimates of component sizes in pixels. In this case, each component has a FWHM of 6 by 4 pixels with the major axis at position angle 20 degrees.

> DOWID FALSE ↵ to hold all of the widths constant (if required).

> STVERS 0 ↵ to have the results places in a new "stars" extension file for later plotting.

> INP ↵ to review inputs.

> GO ↵ to run the task.

To improve accuracy, include as small an area as possible in the fit. In some cases, it is useful to hold some of the parameters constant, particularly when fitting a complex clump of emission with several components. The parameters can interact. Error estimates are given for each component based on the image rms in the image header or determined by robust means over the full input image plane. IMFIT will sometimes fail to converge in complicated regions. When this happens, you might try using the task JMFIT, which is very similar in function, but uses a different mathematical method to minimize the rms. Comparison of the results of IMFIT and JMFIT will sometimes be instructive. The tasks will correct the results for the effects of the primary beam and bandwidth smearing if you wish. These tasks return adverbs FMAX, FPOS, and FWIDTH with the answers to the fit, DOMAX, DOPOS, and DOWIDTH with the uncertainties in the fits, and FSHIFT with

the shift needed to place the first component directly on a pixel. It is wise to treat the results of MAXFIT, IMFIT and JMFIT with caution. The estimates of the errors, in particular, are based on theory and on trials of deconvolution over a range of widths.

The verb MFITSET may be used to set many of the input adverbs for IMFIT and JMFIT from the image displayed on the TV. The verb sets the image name adverbs, then directs you to set the fitting window (like TVWINDOW), and then to point at the peak of Gaussian component 1 (sets GMAX(i) and GPOS(*,i)), then its major axis half-power point (sets GWIDTH(1,i) and GWIDTH(3,i)), then its minor axis half-power point (sets GWIDTH(2,i)). Push buttons A, B, or C to set each of these points or push button D to stop the process and set NGAUSS. Instructions will appear on your terminal.

Alternatively, use RUN INPFIT ↵ (see § 12.2.1) to obtain a procedure which will help to supply input parameters to IMFIT. This RUN file loads a procedure called INPFIT into AIPS. To invoke it, load the image which you want to fit onto the TV with TVALL and type INPFIT (3) ↵ to specify three components. The procedure will prompt you to set the desired sub-image window with the TV cursor (it uses verb TVWINDOW) and then to point the TV cursor at the peaks of each of the Gaussians, click the left mouse button when the cursor is correctly placed, and push button A, B, C, or D. The inputs GMAX, GPOS, BLC, and TRC are set in this way.

7.5.3 Source recognition and fitting (SAD, TVSAD)

The task SAD (§ 10.4.4) attempts to find all sources in a sub-image whose peaks are brighter than a given level. It searches the sub-image specified by BLC and TRC for all points above this level and merges such points in contiguous "islands." For each island, initial estimates of the strength, size, and number of components are generated. Then the fitting algorithm used in JMFIT is called to determine the least square Gaussian fit. Solutions which fail to meet certain criteria can be retried as two components and, if they still fail, rejected. SAD is a task with many adverbs, a full description of which would be beyond the scope of this *CookBook*. Enter EXPLAIN SAD ↵ for a full description of this task and its parameters. The effects of bandwidth smearing and the primary beam may be corrected. SAD produces a Model-Fit extension file which may be converted to a stars file (§ 6.3.2) with MF2ST. The MF file may be printed with MFPRT in formats suitable for STARS and in formats which may be used, with task BOXES, to prepare Clean boxes for input to the imaging tasks. SAD can now directly attach a "stars" extension file to the input image containing the fit Gaussians. Plot tasks can then overlay them on images.

A version of SAD called TVSAD has been written along with an *AIPS* Memo[2] was written describing its use in detail. Like SAD it finds source islands and makes an initial guess. At this point it displays an interpolated image of the area surrounding the island and marks it with the island boundary and the initial guesses (plotted as ellipses at the half-power point). You may choose from a TV menu to enhance the display, to change the island boundary, to change one of the initial guesses, to change the number of Gaussians guessed and set their values (like verb MFITSET), to have the fit attempted, or to reject this island and go on to the next. After the fit, a residual image is displayed with the island boundary and fit Gaussians marked. At this stage you may accept the fit, go back to the previous menu to try again, or reject this island and go on the next island. You may turn off the TV interaction and have the task run automatically. It will turn the TV interaction back on if it finds a fit that does not meet the acceptance criteria. Finally, TVSAD writes out the residual image, printer displays, and extension files exactly like SAD.

[2]Greisen, E. W. 2014, "TVSAD: interactive search and destroy" AIPS Memo 119, https://www.aips.nrao.edu/aipsdoc.html

7.5.4 Gaussian fits to slices (SLFIT)

You can generate a one-dimensional slice (profile) through any plane (characterized by the first two coordinates) of an image file using the *AIPS* task SLICE. The output file is appended to the image file as an SL extension file. Slices are computed along lines in the two-dimensional image joining any valid pair of points selected by BLC and TRC. A slice along the third axis, at a single pixel (not necessarily integers) in the first two axes, may also be prepared. Slices directly along any single axis may be saved without interpolation if desired. If the slice is directly along an FQID axis, SLICE will output a slice which is linearly sampled in frequency. Tasks ISPEC, BLSUM, and TVSPC may also save slice extension files. These are somewhat different in that they can represent a sum over an area in the sky as a function of frequency. Nonetheless, the slice file display and fitting software is of considerable use with these as well. The set of software dealing with slice file analysis and display can be obtained on your terminal by typing ABOUT ONED ↵. The list is also given in Chapter 13.

To generate a slice:

> TASK 'SLICE' ; INP ↵ to review the inputs to SLICE.

Use INDISK and GETNAME to select the input image. The beginning (BLC) and ending (TRC) points for the slice can be specified conveniently using the TV cursor if the image to be sliced is first displayed on the TV with TVLOD or TVALL. To set these points with the TV, type:

> SETSLICE ↵

then set the TV cursor to the desired beginning point for the slice, press the left mouse button, and repeat for the ending point for the slice. Note that, for slices, BLC need not be below or to the left of TRC. Finally:

> GO ↵ to generate the slice file.

Slice files may be output as ASCII text files using the OUTTEXT adverb. They contain detailed pixel and coordinate information as well as the interpolated slice values. They may also be printed later by SLPRT including any one of the models fit to it. Slice files are archived in your disk catalog as SL extensions to the image file from which they were derived. Running SLICE again with new parameters does not overwrite the slice file, but makes another with a higher "version" number. To review and/or delete slice files, follow the instructions for EXTLIST and EXTDEST of plot files in § 6.3 above, but use INEXT 'SL' ↵ in place of INEXT 'PL' ↵.

When SLICE has terminated, the file may be plotted in the TV display on your workstation using:

> INP TVSLICE ↵ to review the inputs to verb TVSLICE.

> INEXT 'SL' ; EXTL ↵ to find the intensity range and number of points in the interpolated slice.

The default scales will plot all slice points on a vertical scale from the slice minimum to the slice maximum. You can alter the part of the slice that is plotted and the vertical scale by specifying, for example:

> BDROP 100 ; EDROP 225 ↵ to drop 100 points from the beginning and 225 points from the end of the plotted portion of the slice.

> PIXRANGE -0.001 0.004 ↵ to set the range of the vertical axis to be -1 to 4 mJy/beam.

> TVSLICE ↵ to plot the slice in the TV window.

Note: several slices may be put on one TV plot. Use TVASLICE ↵ for the additional ones. Multiple colors may be achieved by using different graphics channels (GRCHAN).

Slice files may be converted into plot files by:

> GO SL2PL ↵

The resulting plot files may then be output by:

> GO LWPLA ↵ to display the plot file on a PostScript printer.

> GO TKPL ↵ to display the plot file in the Tek window.

> GO TVPL ↵ to display the plot file on a TV graphics plane.

The task SLFIT fits Gaussian components to one-dimensional data in slice files. Assuming that the usual GETNAME step has been done, a typical session would go like:

> INEXT 'SL'; EXTL ↵ — to list the parameters of the slice files.
> INVERS m ↵ — to select the $m^{\rm th}$ file for analysis.
> TVSLICE ↵ — to plot the slice in the TV window.
> EDROP 840 ; BDROP 700 ↵ — to select a subsection to fit.
> TVSLICE ↵ — to re-plot just the subsection.
> ORDER 1 ↵ — to fit a linear baseline, no baseline up to quadratic are allowed.
> NGAUSS 2 ↵ — to fit 2 Gaussians.
> TVSET ↵

This verb will prompt you to POSITION CURSOR AT CENTER & HEIGHT OF GAUSSIAN COMP 1. Move the cursor to the requested position and hit any button. Then you are asked to POSITION CURSOR AT HALFWIDTH OF GAUSSIAN COMP 1. Move the cursor to the half-intensity point of the component and click any button. Continue until all components have been entered. (Note: these operations are also available on the Tek device with verbs beginning with TK. We recommend the TV versions since cursor reading in X-Windows emulations of Tek devices appears to be unreliable.) Then type:

> TVAGUESS ↵ — to plot the guess on top of the slice plot.

If everything looks ok, then:

> GO SLFIT ↵ — to run the task.

When the task gets an answer, the solution will be displayed as *AIPS* messages, recorded in the message file, and recorded in the slice file itself. To get a hard copy of the results:

> PRTASK 'SLFIT' ; PRTMSG ↵ — to print the message file.

and, to display the results in the TV window, enter:

> TVSLICE ↵ — to re-plot the slice.
> TVAMODEL ↵ — to add the model results to the plot.
> TVACOMPS ↵ — to add the individual components of the model to the plot.
> TVARESID ↵ — to add the residuals (data – model) to the plot.

The TV plot may be easier to examine if you use different GRCHAN for the different types of plot. To get a higher quality plot of the results, an example of which is shown in § 6.3.2.1, type:

> DORES TRUE ; DOMOD 3 ↵ — to request the residuals, the model, and the model components.
> DOSLICE FALSE ↵ — to leave the slice data out of the plot.
> TASK 'SL2PL' ; GO ; WAIT ↵ — to make a plot file and wait for it to be complete.

7.5.5 Spectral parameter fitting (TVSPC, XGAUS, ZEMAN, AGAUS, ZAMAN, RMFIT, FARS)

Task TVSPC assists users in getting to know their data cubes. It uses a 2-dimensional image plane of some parameter that is useful in visualizing the full field. A moment-zero image is one obvious choice. The user selects a pixel on this image using the TV display and a spectrum at that coordinate is displayed. The task is highly interactive allowing a quick choice of an interesting pixel. The spectrum, or a portion of the spectrum, at that point may be fit by up to 4 Gaussians plus baseline and may be saved as a SLice file. The corresponding spectrum from a second transposed cube may also be displayed, allowing for example the I and V or the Q and U polarizations to be displayed simultaneously. Optionally, a third spectral cube may be displayed with the spectral plane selected interactively from the spectral plots. See the associated *AIPS* Memo[3]. In earlier releases, use XPLOT with DOTV 1 to familiarize yourself with your image cube.

[3]Greisen, E. W. 2025, "Exploring Image Cubes in *AIPS*," AIPS Memo 120 revised, https://www.aips.nrao.edu/aipsdoc.html

A task called TVIEW offers interactive TV displays of an image cube in all 3 transpositions plus a reference image or spectrum. The task is similar to TVSPC but is designed solely to help the user become better acquainted with the data cube. The task is described in an *AIPS* Memo[4].

XGAUS is an interactive task which can fit up to 32 Gaussians and a linear baseline to each row of an image. In 31DEC13, it was redesigned so that it begins by setting up a table with a row for each pixel on the second and thirs axes of the cube. It populates the table with the peak image intensities and fit results, so that it may be re-started multiple times with the same table, and so that it provides multiple means to edit the results before it finally writes its results as a set of $n-1$ dimensional image files. Although XGAUS was designed for use primarily on transposed spectral-line cubes (see § 8.5.2), it has a wide variety of other applications. The interaction is optional and uses the TV window on your workstation. The data, initial guess, model fit (total and components), and the residual for each row may be plotted on the TV screen. If the number of Gaussians being fit is larger than one, you may choose for each row to enter a revised initial guess using the cursor in the TV window. This process is similar to that of TVSET described above (§ 7.5.4). If things go well, you may turn off the interactive model setting; the task will turn it back on if there is a bad fit at some pixel. After all selected pixels have been fit, the task enters an image and edit mode. Images are computed for the residual rms and for all Gaussian parameters and their uncertainties. A menu is shown offering displays of the images and several methods to revisit the solutions for selected pixels. This includes swapping the solutions between components and re-doing a list of individual pixels or all pixels for which the solutions or uncertainties exceed specified ranges. When you are satisfied with the result, you may instruct the task to write out images of the residual rms and the Gaussian parameters and their uncertainties. The use of XGAUS as well as ZEMAN, RMFIT, and the plotting and modeling tasks (below) is discussed in detail in *AIPS* Memo 118.[5] See the memo for recent (2025) changes.

Task ZEMAN was written to fit for Zeeman splitting. The basic function fit to the V Stokes spectrum is a constant times the total intensity plus another constant times the derivative of the total intensity with frequency. The task is similar to XGAUS in that it first sets up a table of image intensities and fit results, then fits all pixels above specified intensities, and then offers the image display and edit modes. Unlike previous Zeeman programs (in other packages), ZEMAN also offers the option to use the results of XGAUS on the total intensity cube to fit the V Stokes parameter with a constant times the total intensity plus a separate constant for the derivative of each Gaussian component with frequency. This allows for multiple magnetic fields at the same pixel, each corresponding to a separate Gaussian spectral component.

Tasks AGAUS and ZAMAN have appeared to perform the same functions but on absorption-line spectral cubes. The fit is done to the observations, but the parameter that is Gaussian is the optical depth. These tasks and the essential mathematics are described in *AIPS* Memo 122[6] which was updated in 2025.

Task XG2PL plots the solution for individual spectra from XGAUS. The corresponding ZEMAN fit may also be plotted in a separate panel. Absorption line and optical depth models may also be plotted. Task XG2XG will copy an existing XG or ZE table to a new one with a different upper limit to the number of Gaussians.

Polarization spectra are often fit for rotation measure. The old task RM reads a transposed cube of the polarization angle made with multiple COMBs, then FQUBE or MCUBE followed by TRANS. It fits the rotation measure and the intrinsic magnetic field direction using one of two possible methods to resolve lobe ambiguities.

The task FARS reads two similar cubes of Q and U polarization images and performs a "rotation-measure synthesis." The output of FARS is an image cube of the real part and another image cube of the imaginary part of the Fourier transform on the λ^2 axis, with or without one-dimensional Cleaning. Slice files of the

[4]Greisen, E. W. 2025, "Further Exploration of Image Cubes in *AIPS*," AIPS Memo 124 revised, https://www.aips.nrao.edu/aipsdoc.html
[5]Greisen, E. W. 2025, "Modeling Spectral Cubes in *AIPS*," AIPS Memo 118 revised, https://www.aips.nrao.edu/aipsdoc.html
[6]Greisen, E. W. 2025, "Modeling Absorption-line Cubes in *AIPS*," AIPS Memo 122 revised, https://www.aips.nrao.edu/aipsdoc.html

complex RM transfer function ("dirty beam") are also written. Task AFARS reads these cubes to produce images of the maximum rotation measure and of the amplitude or phase at the maximum. Task RFARS applies the rotation measure image to the FARS input cubes to produce de-rotated cubes of Q and U. These may be SQASH'ed to make an image of the average un-rotated polarization. The RUN file/procedure DOFARS assists with the whole process, doing matrix transpositions, finding channel weights, running FARS, transposing the outputs, and cleaning up.

Rotation-measure synthesis remains an experimental procedure at present. To test the handling of spectra in Q and U, the task TARS was written. It reads a text file giving the spectrum of Q and U at some single coordinate and performs the RM synthesis using methods from FARS plus an additional experimental form of complex Clean. Up to 20 model components may be added to the Q/U spectral data at specified rotation measure, amplitude, and phase. Task QUXTR was written to extract a spectrum from and pair of Q and U polarization cubes for input to TARS. The text-file output of TARS may then be plotted with TARPL, which allows multiple RM spectra from the same TARS output file to plotted on top of each other. In 31DEC20, new task MTARS prepares model Q/U spectra with up to 9999 model components and FTARS performs an interactive fitting of the input to TARS using the TARS output as a guide. This is RMFIT (see below) for a single spectrum. TARPL can now also lot the input to TARS.

It has been found that rotation-measure synthesis is severely limited in its ability to separate multiple RM components. Therefore, another XGAUS-like task called RMFIT was written. It uses the FARS rotation-measure output cubes to provide initial guesses for fits to the Q and U polarization spectral cubes. Like XGAUS, it first builds a table of image intensities to hold the fit results and then displays the FARS spectrum at each pixel to take an initial guess with which to fit the Q and U spectra. They are displayed and, if the fit is accepted, the task goes on to the next pixel. RMFIT is restartable and has the same sort of image/edit stage in which images are displayed and the fits may be revisited by means similar to those in XGAUS. An option to fit a spectral index in Q and U is offered, but, in general, the input cubes should have been corrected by the total intensity spectral index and the channel-dependent primary beam before being used in RMFIT. An additional spectral index in Q and U may then indicate "thick" rotation measure screens rather than a true source spectral index. Other models of thickness may be fit instead. The task may be re-started repeatedly and output images written only when desired. Despite rather simple fitting methods, RMFIT has shown the ability to separate closely spaced components not separable by rotation measure synthesis. Task RM2PL makes plot files from the solutions for individual Q/U or Polarization/angle spectra.

7.6 Image analysis

Image analysis is a very broad subject covering essentially all that *AIPS* does or would like to do plus specialized programs designed to analyze a user's particular image in the light of his favorite astrophysical theories. *AIPS* provides some general programs to perform geometric conversions, image filtering or enhancement, and model fitting and subtraction. These are the subjects of the following sections. Specialized programs for spectral-line, VLBI, and single-dish data reduction are described in Chapter 8, Chapter 9, and Chapter 10, respectively. Chapter 11 of *Synthesis Imaging in Radio Astronomy*[7] covers the topic of image analysis in more detail.

7.6.1 Geometric conversions

The units of the geometry of an image are described in its header by the coordinate reference values, reference pixels, axis increments, axis dimensions, and axis types. The types of coordinates (celestial,

[7] *Synthesis Imaging in Radio Astronomy*, A collection of Lectures from the Third NRAO Synthesis Imaging Summer School, eds. R. A. Perley, F. R. Schwab and A. H. Bridle, Astronomical Society of the Pacific Conference Series Volume 6 (1989)

galactic, etc.) and the type of tangent-plane projection (SIN from the VLA, TAN from optical telescopes, ARC from Schmidt telescopes, NCP from the WSRT) are specified in the *AIPS* headers by character strings. See § 6.1.2 and *AIPS* Memo No. 27 for details of these projections. A "geometric conversion" is an alteration of one or more of these geometry parameters while maintaining the correctness of both the header and the image data. The *AIPS* tasks which do this interpolate the data from the pixel positions in the input image to the desired pixel positions in the output image.

The simplest geometric conversion is a re-gridding of the data with new axis increments and dimensions with no change in the type of projection or coordinates. The task LGEOM performs this basic function and also allows rotation of the image. One use of this task is to obtain smoother displays by re-gridding a sub-image onto a finer grid. To rotate and blow up the inner portion of a 512^2 image, enter:

> TASK 'LGEOM' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to select the image.
> BLC 150 ; TRC 350 ↵	to select only the inner portion of the image area.
> IMSIZE 800 ↵	to get an 800^2 output image. This will allow the sub-image to be blown up by a factor of 3 and rotated without having the corners "falling" off the edges of the output image.
> APARM 0 ↵	to reset all parameters to defaults.
> APARM(3) = 30 ↵	to rotate the image 30° counterclockwise (East from North usually).
> APARM(4) = 3 ↵	to blow up the scale (axis increments) by a factor of 3.
> APARM(6) = 1 ↵	to use cubic polynomial interpolation.
> INP ↵	to check the inputs.
> GO ↵	to run the program.

LGEOM allows shifts of the image center, an additional scaling of the y axis relative to the x axis, and polynomial interpolations of up to 7^{th} order. OGEOM is similar to LGEOM, but handles blanked pixels in a manner that does not increase the blanked area.

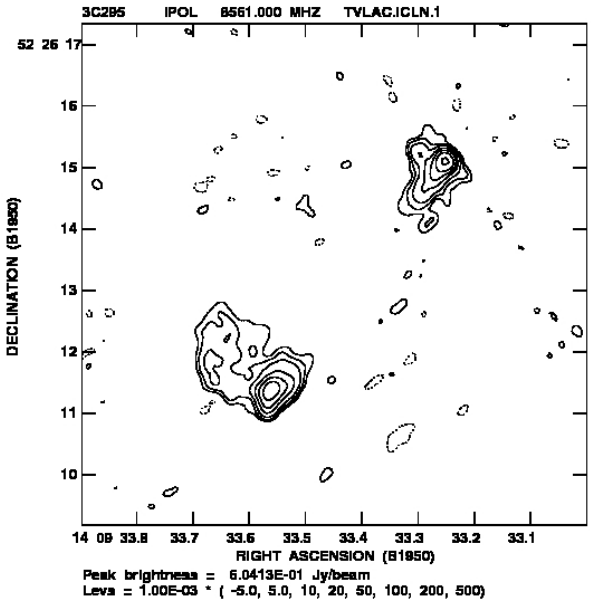
A much more general geometric transformation is performed by OHGEO and HGEOM, which convert one image into the geometry of a second image. The type of projection, the axis increments, the rotation, and the coordinate reference values and locations of one image are converted to those of a second image. One of these tasks should be used before comparing images (with COMB, KNTR, PCNTR, BLANK, TVBLINK, etc.) made with different geometries, *i.e.*, radio and optical images in different types of projection or VLA images taken with different phase reference positions. Use EXPLAIN OHGEO ↵ to obtain the details and useful advice. SKYVE re-grids images from the Digital Sky Survey (optical DSS) into coordinates recognized by *AIPS*.

A potentially very powerful transformation is performed by PGEOM. In its basic mode, it converts between rectangular and polar coordinates. An example of this operation is illustrated in Figure 7.1. However, PGEOM can also "de-project" elliptical objects to correct for their inclination and "unwrap" spiral objects. Type EXPLAIN PGEOM ↵ for information.

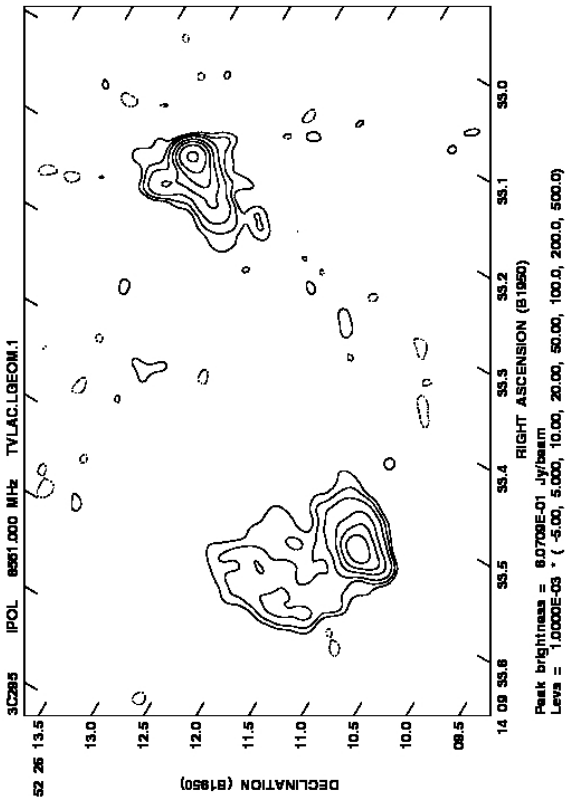
7.6.2 Mathematical operations on a single image

The task MATHS allows the user to do a mathematical operation on a single image on a pixel by pixel basis. Currently supported mathematical operators are: SIN, COS, TAN, ASIN, ACOS, ATAN, LOG, LOGN, ALOG, EXP, POLY, POWR, and MOD. An example of MATHS follows, in which the output image (OUT) is computed in terms of the natural logarithm of the input image (IN) as follows: $\texttt{OUT} = 4 + 2 \times (\log(3 \times \texttt{IN}) - 1)$

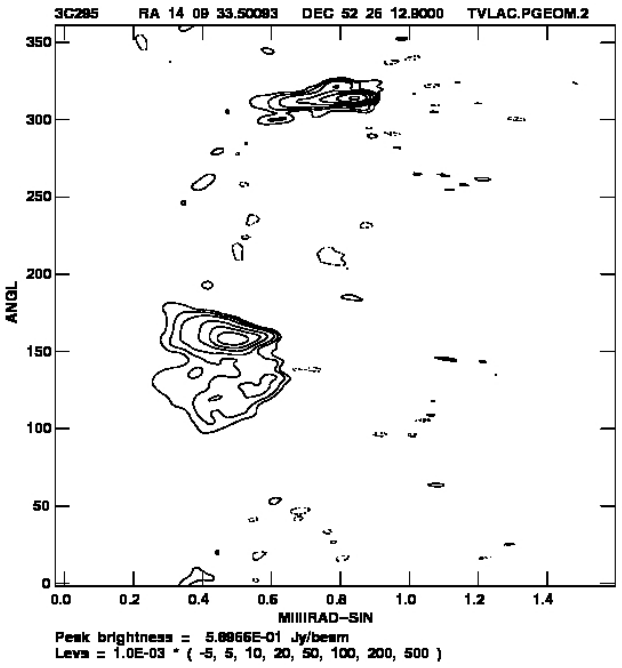
> TASK 'MATHS' ; INP ↵	to review the required inputs.
> INDI 0 ; MCAT ↵	to help you find the catalog number of the input image.



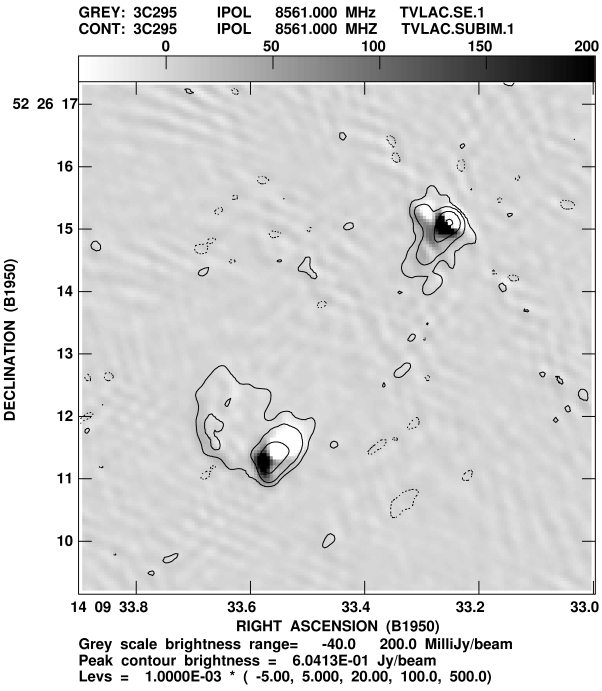
input image



LGEOM with rotation and interpolation



PGEOM



NINER: 'SE ' derivative

Figure 7.1: Geometric and other functions on an image.

> INDI *n* ; GETN *ctn* C_R	to specify the image, on disk *n* catalog slot *ctn* as the input.
> OUTN '*xxxxx*' C_R	to choose *xxxxx* as the name for the output image.
> OUTC '*ccc*' C_R	to choose *ccc* as the class for the output image.
> OPCODE 'LOGN' C_R	to specify the operation to be performed (a natural logarithm).
> CPARM 4 , 2 , 3 , -1 C_R	to specify the coefficients.
> GO C_R	to start MATHS.

Undefined output pixels (in the current example, all pixels in the input image ≤ 0) are either blanked (CPARM(6) ≤ 0) or put to zero (CPARM(6) > 0). Type EXPLAIN MATHS C_R for further information on the available operators and the meaning of CPARM for any particular operator.

7.6.3 Primary beam correction

PBCOR allows correction for the attenuation due to the shape of the primary beam. Its use is straightforward:

> TASK 'PBCOR' ; INP C_R	to review the required inputs.
> INDI 0 ; MCAT C_R	to help you find the catalog number of the input image
> INDI *n* ; GETN *ctn* C_R	to select the input image from disk *n* catalog slot *ctn*.
> OUTN '*xxxxx*' C_R	to specify *xxxxx* for the name of the output image.
> OUTC '*ccc*' C_R	to specify *ccc* for the class of the output image
> PBPARM 0 C_R	to use the VLA, EVLA, ATCA, GMRT, or MeerKAT beam parameters which are known for the particular receiver.
> COORDIN 0 C_R	to use the pointing position from the image header.
> GO C_R	to start PBCOR.

The default behavior requested above uses the position in the header as the pointing position and uses the empirically determined shape of the EVLA, VLA, MeerKAT, GMRT, or ATCA primary beam; PBCOR will scale the primary beam shape according to the frequency provided in the image header and use the parameters associated with the particular antenna feed. These defaults can be overridden by specifying particular values of COORDIN and PBPARM. They beam parameters are frequency dependent within each band, so PBCOR determines the beam value at the two tabulated frequencies nearest to the image frequency and then interpolates. In 31DEC21, Gaussian and polynomial beam patterns have been added as options specified with OPCODE and APARM. Beam parameters for MeerKAT and GMRT were added to 31DEC22 and 31DEC23, respectively, and are known to all tasks which correct for the primary beam.

The task SPCOR may be used to apply corrections both for primary beam and for spectral index. The latter are based on images of spectral index and spectral index curvature such as those used in IMAGR (§ 5.3.4.4). Such corrections may be significant in Faraday-rotation synthesis (§ 7.2, § 7.5.5).

An image of the primary beam may be generated with the task PATGN using OPCODE 'BEAM' C_R with other adverbs to give the antenna type, frequency, cell size, image size, and, optionally, the parameters of the beam shape.

7.6.4 Changing the resolution of an image

CCRES allows you to change the resolution of a Cleaned image by removing any Clean components in the image and then restoring Clean components with your choice of resolution. The task may also be used to remove Clean components to create a residual image or to restore Clean components to an existing residual image. CCRES allows you to smooth or hyper-resolve your image. Unlike RSTOR, CCRES re-scales the residual image to put it into units of Jy/beam for the new beam. This may be a superior way to image with a beam

that does not replicate the central portion of the dirty beam. IMAGR leaves the residual image in units of Jy per dirty beam while restoring the Clean components in units of Jy per the Clean beam given in the header.

Note that CCRES only changes the Clean components, not the residual image resolution. Furthermore, CCRES does not take into account the varying resolutions of the many planes in an image cube. CONVL on the other hand, with OPCODE 'GAUS', will convolve both the Clean components and the residual to the requested resolution, taking into account the change in input resolution as a function of frequency.

7.6.5 Filtering

For our purposes here, we can define "filtering" as applying an operator to an image in order to enhance some aspects of the image. The operators can be linear or nonlinear and do, in general, destroy some of the information content of the output image. As a result, users should be cautious about summing fluxes or fitting models in filtered images. (Technically, these remarks can also be made about Clean and self-calibration.) However, filtered images may bring out important aspects of the data and often make excellent, if unfamiliar-looking, displays of particular aspects.

NINER produces an image by applying an operator to each cell of an image and its 8 nearest cells. The task offers three nonlinear operators which enhance edges (regions of high gradient in any direction). It also offers linear convolutions with a 3×3 kernel which can be provided by the user or chosen from a variety of built-in kernels. Among the latter are kernels to enhance point sources and kernels to measure gradients in any of 8 directions. The 'SOBL' edge-enhancement filter can bring out jets, wisps, and points in the data, while the gradient convolutions produce images which resemble a landscape viewed from above with illumination at some glancing angle (as when viewing the Moon). Both are very effective when displayed on the TV or by the KNTR / LWPLA combination (see Figure 7.1). Enter EXPLAIN NINER ↵ for additional information.

MWFLT, at present, applies any one of six non-linear, low-pass filters to the input image. Each filter is applied in a user-specified window surrounding each input pixel. One of the operators is a "normalization" filter designed to reduce the dynamic range required for the image while bringing out weaker features. Two of the operators are a "min" and "max" within the window. When applied in succession, they produce a useful low-pass filtered image (Rudnick, L. 2002, PASP, 114, 427). Other operators produce, at each pixel, the weighted sum of the input and the median, the "alpha-trimmed" mean, or the alpha-trimmed mode of the data in the window surrounding the pixel. These filters can be turned into high-pass filters by subtracting the output of MWFLT from the input with COMB. Type EXPLAIN MWFLT ↵ for further information.

Histogram equalization provides another form of non-linear filtering. HISEQ converts the intensities of the full input image to make an output image with a nearly flat histogram. This magnifies small differences in the heavily occupied parts of the histogram (usually noise) and diminishes large differences in the less occupied parts (often real signal). TVHLD is an interactive task that loads an image to the TV with histogram equalization and then allows the intensity range and method of computing the histogram to be modified. It can write out at the end an equalized image in arbitrary (non-physical) units. AHIST does an "adaptive" histogram equalization on each pixel using a rectangular window centered on that pixel. This will magnify small differences in a more local sense, bringing out structures in smooth areas of different brightness. SHADW generates a shadowed image as if a landscape having elevation proportional to image value were illuminated by the Sun at a user-controlled angle. Although these tasks magnify noise, they are likely to elucidate real structures in large areas of nearly constant brightness.

7.7 Modeling in the image and uv planes

Special considerations for analysis and modeling of spectral-line image cubes are discussed in § 8.6. Models of galaxy rotation are fit by GAL to images of the predominant velocity (first moment images usually), while the entire data cube may be fit by task CUBIT.

The addition of model data to an image or *uv* data set is often useful either to simplify later processing steps or to study processing steps using a "source" of known structure. For example, the removal of the response to an appropriate uniform disk from the *uv* data for a planet will leave Clean the task of deconvolving only the remaining fine-scale structure to which it is well suited. The removal of a few bright point sources of known position and strength may allow imaging with significant tapers in a numerically smaller field. The tasks MODIM, MODSP, UVMOD, MODAB, and SPMOD will add (or subtract) up to 9999 point, Gaussian, disk, rectangular, spherical, or exponential sources to the (scaled) input image or *uv* data, respectively. These tasks can also add noise and allow the original data to be replaced by the model. UVMOD can include a spectral index for the sources, SPMOD includes spectral lines for the sources, MODIM can include both a spectral index, a rotation measure, and even a rotation measure thickness for each component, and MODSP can include polarized spectral lines with spatially varying spectral structure. Type EXPLAIN MODIM ; EXPLAIN UVMOD ; EXPLAIN SPMOD ; EXPLAIN MODAB ; EXPLAIN MODSP ↵ for details or see *AIPS* Memos 118[8] and 122[9]. For simple cases, the older task IMMOD may be easier to use than MODIM.

The task CCMOD will create a clean-components file representing a Gaussian, disk, or Gaussian times disk model. Clean may then be "restarted" with the model as its initial set of components.

The task UVFIT may be useful for fitting Gaussian or uniform-sphere models to *uv* data sets with a maximum of 2.5 million visibilities. Up to 60 source components may be fit along with up to 30 antenna gains. Provision for getting initial guesses for the specified spectral channel from a text file with guesses for many channels is now available as is a compact text file output option for the solutions. Task OMFIT is a more complicated and difficult to use task which fits source components in a *uv* data set along with antennas gains. It allows multiple different model types and even a full self-calibration. The results from OMFIT are often well worth the extra effort to obtain them.

AIPS contains tasks which create simulated data. PATGN will make images of various patterns. DTSIM generates *uv* data following instructions given primarily in a KEYIN-format text file. You may specify the antennas in detail or as the VLBA or VLA. Source parameters and models, frequency values and structure, data calibration errors of numerous types, scan structure and sequence, and more may be specified. UVCON also generates *uv* data from "scratch" using text files of antenna data and Tsys and efficiency data as well as images of a source model. The two tasks differ in their approach, making both of considerable interest.

7.7.1 Analysis in the uv plane

Several tasks now perform some analysis on *uv* data. The other new task, ELFIT, fits polynomials to table data as functions of elevation, zenith angle, hour angle, parallactic angle, or azimuth. It has been used, for example, to measure antenna spillover, plotting Psys versus zenith angle.

DFTIM is similar to the plot task DFTPL in that it Fourier transforms visibility data to a particular image pixel (celestial coordinate). DFTPL makes a plot versus time for a single group of frequencies. It can do various Stokes including polarization amplitude and position angle. DFTIM instead writes out an image at that coordinate with frequency and time as the axes. The frequencies and times in this "waterfall" image may

[8] Greisen, E. W. 2025, "Modeling Spectral Cubes in *AIPS*," AIPS Memo 118 revised, `https://www.aips.nrao.edu/aipsdoc.html`
[9] Greisen, E. W. 2025, "Modeling Absorption-line Cubes in *AIPS*," AIPS Memo 122 revised, `https://www.aips.nrao.edu/aipsdoc.html`

be averaged as the image is created. ACIMG makes a similar waterfall plot for autocorrelation data, with the third axis being antenna number. These tasks can now display and image other parameters in addition to the correctly computed DFT. New task (DFTQU computes a 3-color image with Stokes Q and U as the two axes with color set by the time.

Statistical analysis of visibility data is performed by several tasks. UVRMS averages the selected data in SOLINT time intervals and then can plot the average values as a function of time and as a histogram. A variety of statistical parameters are computed and printed. SPRMS determines and plots the mean and standard deviation of the selected data as a function of spectral channel. Note that both tasks will combine data from multiple sources, baselines, etc. if so instructed. RIRMS computes statistics over the included visibility data and then prints a matrix over baselines of the real and imaginary parts separately. Histogram plots and plots over time may be generated. In 31DEC21, task VBRFI computes statistics for a data set consisting only of auto-correlation records. It writes out a text file containing the results and can make plots. PLRFI can make plots from one or more of these text files.

7.8 Additional recipes

7.8.1 Golden mousse

1. Combine 1 cup mashed ripe **bananas**, 2 tablespoons **orange juice**, 1/4 cup shredded **coconut**, 3 tablespoons **brown sugar**, a few grains **salt**, and 1/8 teaspoon grated **orange rind**.
2. Whip until stiff 1 cup **heavy cream**.
3. Fold whipped cream into fruit mixture and turn into freezing tray. Freeze rapidly without stirring until firm.

7.8.2 Coriander banana nut bread

1. Blend together in a large bowl $1\frac{2}{3}$ cups sifted all-purpose **flour**, 3/4 cup **sugar**, 1 tablespoon **baking powder**, 1/2 teaspoon **baking soda**, 1/2 teaspoon **salt**, 2 teaspoons ground **coriander**.
2. Mix in 1 cup chopped unblanched **almonds** and set aside.
3. Melt 1/3 cup **shortening** and set aside to cool.
4. Mix until well blended 1 large well-beaten **egg**, 1/4 cup **buttermilk**, and 1 teaspoon **vanilla extract**.
5. Blend in $1\frac{1}{4}$ cups mashed ripe **bananas** and the shortening.
6. Make a well in center of dry ingredients and add banana mixture all at one time. Stir only enough to moisten dry ingredients.
7. Turn into greased $9 \times 5 \times 3$-inch loaf pan and spread to corners.
8. Bake at 350° F about 1 hour or until a wooden pick comes out clean when inserted in center of bread. Immediately remove from pan and set on rack to cool.

8 Spectral-Line Software

This chapter deals with the analysis and reduction of spectral-line data after they have had the basic calibrations, described in Chapter 4, applied. Spectral-line software generally involves three-dimensional images, often called "cubes", in which one of the three axes is frequency or velocity. Special programs are available to build, manipulate, and transpose these cubes and to display them properly. Most of the continuum software will work on data cubes or on appropriate two-dimensional subsets from a data cube. Spectral-line *uv* data can be read into $\mathcal{AIPS}$ from "*uv*" FITS tapes. Often, however, the data already exist on disk having been calibrated first within $\mathcal{AIPS}$.

The data reduction process at this point has a number of stages, namely data preparation and assessment, self-calibration, continuum subtraction, imaging, display and manipulation of data cubes, and analysis. This chapter will address each of these areas in varying, but modest, detail on the assumption that the reader is somewhat familiar with the contents of previous chapters in this *CookBook*. Some aspects of the art of spectral-line imaging are discussed in Chapters 17 and 18 of *Synthesis Imaging in Radio Astronomy*[1]. A brief outline of the basic calibration process is given in Appendix A of this *CookBook* and an outline of spectral-line analysis and calibration is given in Appendix B.

8.1 Data preparation and assessment

In the following sections, it is assumed that the *uv* data are on disk and that the basic calibrations have been applied to them. If your data are not currently on disk, you will need to read them in from magnetic tape or FITS disk files. For tape, mount the tape in hardware and software (§ 3.9) and then position the tape to the desired data files:

> NFILES *n* ; AVFILE ↵	to advance *n* files.
> TPHEAD ↵	to check that the tape is correctly positioned.

Use FITLD to read in the data:

> TASK 'FITLD' ; INP ↵	to review the inputs.
> OUTCL ' ' ; OUTN ' ' ↵	to use the file names recorded on tape (the default).
> NFILES 0 ↵	to read the current file on tape.
> NCOUNT 2 ↵	to load two files (if desired).
> GO ↵	to run FITLD.

For FITS disk files, set DATAIN appropriately and the run FITLD. Examine the *uv* data set header with IMHEAD after FITLD finishes. It should show multiple pixels on the FREQ axis, like the *uv* header illustrated in § 3.3.4.

If your data are not yet calibrated, consult Chapter 4, especially § O.1.7. When the CL, BP, and other calibration tables are complete, apply them to the line data with SPLIT:

> TASK 'SPLIT' ↵	
> SOURCE '*sou1*' , '*sou2*' , . . . ↵	to select sources, ' ' means all.
> TIMERANG 0 ↵	to keep all times.
> BIF 1 ; EIF 0 ↵	to keep all IFs.
> BCHAN 1 ; ECHAN 0 ↵	to keep all spectral channels.
> FREQID 1 ↵	to set the one FQ value to use.
> DOCALIB TRUE ↵	to apply calibration.

[1] *Synthesis Imaging in Radio Astronomy*, Astronomical Society of the Pacific Conference Series, Volume 6, "A Collection of Lectures from the Third NRAO Synthesis Imaging Summer School" eds. R. A. Perley, F. R. Schwab and A. H. Bridle (1989).

> GAINUSE 0 C_R	to use the highest numbered CL table.
> FLAGVER 1 C_R	to apply the flag table.
> DOPOL TRUE C_R	to correct for feed polarization.
> DOBAND 1 C_R	to correct bandpass.
> BPVER 1 C_R	to select BP table to apply.
> STOKES '' C_R	to write the input Stokes type.
> DOUVCOMP TRUE C_R	to write visibilities in compressed format.
> APARM 0 C_R	to clear VLBA options, including the one to calibrate the data weights.
> INP C_R	to review the inputs.
> GO C_R	to run the program when inputs set correctly.

The *uv* data produced by this process should have applied to them the full calibration and editing determined from the calibration sources. Whether these are adequate or will need to be enhanced from the source data themselves is now to be determined.

Your data may need to be edited further at this point. If your calibrator data were almost free of problems, save for simple matters like dead antennas, then your source data are not likely to need much editing either. If your calibrator data required detailed and erratic data flagging, then your source data will also need attention. All of the considerations of § 4.3.5 apply here too. In either case, the first step is to check for narrow-band interference which is a fairly common problem at some wavelengths. The best task for this is SPFLG (§ 10.2.2), a spectral-line version of TVFLG (§ O.1.6). SPFLG displays spectral channels for all IFs along the horizontal axis and time along the vertical axis, one baseline at a time. This is tedious to use for editing all of the data from a large interferometer. Instead, select a few of the shorter baselines (with the ANTENNAS and BASELINE adverbs) and do a quick check for interference. If your data has sufficient signal-to-noise, then this quick examination should let you check which channels have signal and whether the Galaxy, for example, has contributed unwanted signal to your data. If you find significant terrestrial interference, you may flag the bad data with SPFLG (consider using the "ALL-BL" option to avoid looking at hundreds of baselines) or with UVFLG (entering "by hand" what you find with SPFLG). Tasks UVLIN and UVLSF (below and § 8.3) both offer the option to flag all channels of a particular sample if any one channel in the continuum-fitting portion of the spectrum has excessive flux after the fit. UVLSF also offers the option to flag when the rms of the continuum fit is excessive. The task CLIP (§ 5.5.1) works in a channel-dependent way on line data. It can be instructed to flag a full spectrum if more than a user-specified number of channels are flagged in that spectrum. FLGIT is designed to work on line data sets that have a great deal of interference, generating channel-dependent flagging in the output file or in a flag table. UVMLN may also be used to generate flag tables for multi- and single-source files to flag seriously deviant data. FLAGR uses multi-channel data to estimate rms and flags any times having excessive rms, amplitude, weight, etc. RFLAG will flag data in detailed ways based on excessive variability in time and/or frequency. REFLG will compress the large flag table made by RFLAG. UVCOP will apply even a very large flag table if so instructed. These tasks will eliminate certain classes of problems without bothering you with the details, which may — or may not — be a good thing.

To do serious editing of a non-spectral nature and to do self-calibration, you will need to create a single-channel data set that represents your multi-channel data in some fashion. If your data contains little continuum signal, but does have a strong spectral-line signal, then use UVCOP to copy the data for the channel with the strongest signal to a single-channel data set. Alternatively, the task AVSPC may be used to average a few adjacent channels together to create the single-channel data. If, on the other hand, your data set contains a useful level of continuum emission — and most do — then your single channel should be some estimate of this continuum. If your line signal is weak and you have done no frequency-dependent editing, then you can use the so-called "channel-0" data set created by FILLM when you first loaded your VLA data to disk. This data set is the vector average of the center 3/4 of the selected spectral channels, but remains useful only if you have applied all of the same calibration, editing and SPLITing to it that you applied to the line data. It may be safest to create a new continuum estimate from the line data in any case.

The classical method for doing this is the task AVSPC:

> TASK 'AVSPC' ; INP ↵	to review the inputs needed.
> IND *m* ; GETN *n* ↵	to specify the input *uv* file.
> CLRONAME ↵	to use the default output names and disk.
> AVOPT ' ' ↵	to average channels only within IFs.
> ICHANSEL *b1 , e1 , i1 , if1 , b2 , e2 , i2 , if2 ,...* ↵	to average every *i1* channel from *b1* through *e1* in IF *if1*, plus every *i2* channel from *b2* through *e2* in IF *if2*, etc.
> ICHANSEL 10 , 50 , 1, 0, 81 , 119 , 1 ↵	*e.g.*, average channels 10 through 50 and 81 through 119 in each IF, omitting the band edges and the channels with line signal.
> INP ↵	to review the inputs.
> GO ↵	to run the program when you're satisfied with inputs.

A non-classical method for estimating the continuum is provided by the task UVLSF It fits a linear "spectral baseline" in the real and imaginary parts of each visibility spectrum. By adding a slope to the channel-independent bias found in AVSPC, it allows for small changes in the continuum visibility with frequency and spatial resolution across the observed band, as well as small, time-variable changes in the spectral passband not fully removed by BPASS. The option to fit even higher order baselines was added. These are seldom useful and can run into problems since they are not constrained by data in the channel ranges containing the line signals. The principal output of UVLSF is a multi-channel data set with this baseline subtracted, *i.e.*, a continuum-free line data set. However, the user may also get a single-channel "continuum" data set evaluated at any one channel from the fit baselines. Thus:

> TASK 'UVLSF' ; INP ↵	to review the inputs needed.
> IND *m* ; GETN *n* ↵	to specify the input *uv* file.
> CLRONAME ↵	to use the default output names and disk.
> BCHAN 0 ; ECHAN 0 ↵	to include all channels.
> ICHANSEL *b1 , e1 , i1 , if1 , b2 , e2 , i2 , if2 , ...* ↵	to average every *i1* channel from *b1* through *e1* in IF *if1*, plus every *i2* channel from *b2* through *e2* in IF *if2*, etc.
> DOOUTPUT TRUE ↵	to get the continuum data set.
> CHANNEL 0 ↵	to select the channel at which the baseline is evaluated — in this case, the default which is the reference channel. Choose one near the center of the band.
> INP ↵	to review the inputs.
> GO ↵	to run the program when you're satisfied with inputs.

The continuum data set will have *AIPS* class BASFIT and be on the same disk as the output line data set. You may also use UVLSF to flag data with excessive residual flux and rms in the channels used for the fitting. You may apply the usual data selection and calibration and flag tables to the input data, meaning that SPLIT need not have been run on the data previously. Note that neither AVSPC nor UVLSF should be used if the continuum visibility changes substantially across the passband, due to large bandwidth and complexity in the continuum source. If the phase of the continuum changes substantially across the passband due to the centroid of the continuum being considerably separated from the pointing center, you should instruct UVLSF to shift the phase reference position before fitting the linear function and then to shift the phases back afterwards.

Your spectral-line data may need further modification and preparation. If you did not observe with "Doppler tracking" (and even the VLA changes frequency only on a scan-by-scan basis), then your data channels are not at the same velocity through the data set. Task CVEL may be used to convert the data to full Doppler tracked form, or simply to shift the center velocity of your reference channel to, for example, the velocity used in another data set. CVEL works on multi-source as well as single-source data sets. It applies any flagging and bandpass calibration to the data before shifting the velocity (which it does by a carefully

correct Fourier transform method). Note, the use of Fourier-transforms means that one *must not* use CVEL an data with channel separations comparable to the widths of some of the spectral features.

Three other tasks allow you to change the number of spectral channels in your data set. The simplest, AVSPC, allows you to average every n adjacent channels together, producing a data set one $n^{\rm th}$ as large. The second, called SPECR, uses Fourier transform methods to increase or decrease the number of channels with a corresponding change in channel spacing. SPECR can also use simple interpolation methods in case the Fourier methods produce undesirable results (such as ringing when a feature is very narrow in the input data set). The third task is called UVFRE and it attempts to convolve the data from one frequency structure onto another frequency structure defined in a second data set (the uv equivalent of HGEOM). These tasks, along with CVEL may be required to convert one data set to the same channel velocity and separation as another data set, before the two can be concatenated (task DBCON) and used together in imaging. If simple frequency smoothing is needed, the calibration routines will apply various convolutions to the spectrum under control of the adverb SMOOTH. IMAGR, allows you to "average" channels together by gridding them into the correct (channel-dependent) locations in the uv plane. This is seldom important to narrow-band observations, but is critical in wide "bandwidth synthesis" observations. Nonetheless, it allows you to keep full spectral resolution in the uv data set, but abandon it for some or all of the imaging.

8.2 Editing and self-calibration

Self-calibration can improve the "dynamic range" of your images significantly but **only** if there is sufficiently good signal-to-noise in the uv database. It works by comparing the input uv data with the predicted visibilities from a model of the source; from this a set of complex gain (amplitude, phase) corrections are generated for each antenna in the array as a function of time. Before engaging in a potentially long and useless exercise, it is wise to look at the continuum or single-channel data set you have created to determine whether there is sufficient signal (with respect to the noise) to enable detailed editing and/or self-calibration. Even if the continuum data would profit from these things, you must also decide whether there is sufficient signal-to-noise in the line data to benefit from the improved editing and calibration. Consult the sections on editing (§ 4.3.5 and § 4.3.13) and on self-calibration (§ 5.4) before deciding to continue with this section.

In fact, there is very little to be written here. The editing and self-calibration of the continuum or single-channel data set are precisely those described in Chapter 4 and Chapter 5. The goal is to create and fill an FG table (if flagging is needed) and an SN table (if self-calibration is needed) attached to the single-source, single-channel data set. Note that, for single-source data, we use an SN table containing the accumulated calibration while we use, for multi-source files, a number of SN tables with incremental calibrations and a single CL table with the net corrections. When all editing and self-calibration are done, you use TACOP (or TAPPE if the flag table must be appended to an existing flag table) to copy the two tables to the multi-channel data set and then use SPLIT to apply both tables to the multi-channel data (and to the single-channel data set too if desired).

In 31DEC22, task CLOSE may be used to plot the rms closure phase or amplitude as a function of channel. Use this on your calibration source or your target data after division by the continuum model. Values should be near zero but will probably rise at the edges of spectral windows (IFs). See § 5.4.4.

In general, you may keep the flagging information in a flag table (specified with adverb FLAGVER) and add to and apply the table whenever it is needed. Be aware that flag tables are applied only by those tasks that have the FLAGVER adverb, but a number of tasks have been upgraded to understand calibration and flagging tables. In fact, data that are not in time order may have flag tables applied (if they are small enough) and time-independent calibrations (*e.g.*, DOBAND=1 and DOPOL > 0) may also be done.

8.3 Continuum subtraction

Most spectral-line observations contain a certain amount of frequency-independent, continuum radiation in addition to the frequency-dependent line signals. In most (all?) cases, it is probably best to separate the two signals at this stage of the data processing. In this way, a *single* continuum image can be constructed to apply to all frequencies. It will probably be necessary to apply Clean or other expensive deconvolution techniques to determine the best estimate of this continuum image. To do this for each channel individually would not only be considerably more time consuming, but would result in different models for the continuum in each channel. Since Clean is a data-adaptive non-linear algorithm, minor disturbances in its progress causes it to converge to surprisingly different solutions. Such "disturbances" could be caused by differing noise and line signals in different channels. (It can be caused simply by differing computational order on identical data using different computers.) Clean also increases the noise in an image in the areas at which there are sources. If the continuum is removed from the line data before Cleaning, then this increase in noise is also removed. To determine the continuum from separately Cleaned channel images is to use noisier individual determinations to get the final estimate. If the imaging and deconvolution were a linear process, then this would not matter. But the deconvolution is non-linear, making it better to start with the best possible estimate of the continuum.

In many cases, the continuum signal is stronger by far than the line signal. In such cases, it is best to use as many channels as possible to estimate the continuum and to remove that estimate as early as possible. The remaining line signal may be of fairly low signal-to-noise ratio and, therefore, not need the same processing that the continuum signal requires. In particular, it may not be necessary to Clean the channel images particularly deeply, if at all.

All of the arguments above suggest that the continuum should be estimated and subtracted in the visibility domain. This has the unfortunate attribute that we can use only those channels which are free of line signal over the entire field of view. (This is because the Fourier transform relation mixes signals from all directions in the field into every visibility sample.) In the image domain, the dirty images also have this unfortunate attribute, in this case because the dirty beam mixes signals from all parts of the field into every pixel. The above arguments suggest that the only time one should determine the continuum in the image domain is when the observations have essentially no channels which are free of line signal over the full field of view. In that case, the continuum at each position will have to be determined from deconvolved images using those channels which are free of line signal at that position.

AIPS provides three separate tasks for fitting and subtracting the continuum in the *uv* plane. All three fit a linear "baseline" to a selected group of channels and subtract that from the data. The reasons for a linear baseline are (1) a continuum source offset from the field center will produce a linear phase slope across the passband, (2) minor other changes in the visibility with baseline may be approximately linear over narrow passbands, and (3) minor passband variations with time appear also to be approximately linear. Higher-order fits can be done by UVLSF, but they do not seem to be justified and are often not well constrained. The first reason suggests that one should do the fit in amplitude and phase, an algorithm implemented by the task UVBAS. This should be used *only* if you have good signal-to-noise in *all* of your visibility samples. The reason for this very stringent requirement is that visibility amplitudes have Ricean rather than Gaussian noise statistics. As a result, very biased estimates are produced in moderate to low signal-to-noise cases. For these, the two tasks UVLSF and UVLIN should be used. They do the fits in the real and imaginary parts of the visibilities and, hence, do not produce biased estimates in the absence of signal. Both tasks allow you to shift the visibilities to move a strong source to the center of the field before doing the fits (and then shift the visibilities back to the original phase center). This changes sinusoidal real and imaginary parts to linear for an accurate fit, but only for those cases where a single source dominates the field. Both tasks offer the option to flag discrepant data, but they differ in the details of how they do this. UVLSF offers the option to write out a "best-fit" single-channel data set as well as subtracting it from the line data; see § 8.1 for an sample inputs to UVLSF.

There are also a number of ways to remove the continuum in the image domain, all of which assume that the channels have been imaged similarly and placed into a three- or more dimensional "cube." The "classical" method to subtract the continuum, which is the least useful, is to average those image planes which contain no line signals using SQASH on each set of line-free planes and COMB to average those averages. Finally, COMB is used to subtract the resulting plane from each plane of the initial cube. In this method, the cube is in the "natural" transposition with the frequency axis third. The other two methods require you to use the task TRANS (see § 8.5.2) to transpose the frequency axis to the first (most-rapidly varying) axis. Then IMLIN may be used to fit polynomial baselines (linear is usually all that is justified) to the line-free parts of each spectrum. This task, like all of the other tasks so far mentioned in this section, requires you to use a fixed set of channels for all positions in the field. Many objects (*e.g.*, rotating galaxies) have spatially-dependent spectra in which each pixel has a rather narrow line width compared to the object or objects as a whole. In such cases, the task XBASL may be used. This task requires some endurance on your part to complete, but it allows you to specify the baseline region for each pixel interactively using the *AIPS* TV or Tek graphics windows.

8.4 Imaging

There are several tasks which may be used to convolve spectral-line data to a rectangular grid and then Fourier transform and Clean that grid to make an image. The old tasks, which we no longer recommend, are called UVMAP, and APCLN and may be used if you insist. Today, IMAGR should be used for all normal imaging and Cleaning. Read § 5.2 for basic information about this task and § 5.3 for information about using it to deconvolve your images. It can read either single- or multi-source data, can apply calibration, and writes all channels into an image cube. IMAGR uses much more flexible and correct methods for data weighting (§ 5.2.3) and uses the superior multi-field method of Cleaning in which source components are subtracted from the visibility data and the images recomputed in every "major cycle." IMAGR grids each channel in exactly the proper place in the *uv* plane and can be used to do frequency smoothing ("averaging") in this most correct of ways. It also offers options to correct your data for various effects of importance over wide fields and bandwidths. It even has experimental variations on Clean (§ 5.3.4) to deal with extended sources and the Clean bias.

To produce a cube of spectral line images with IMAGR from channels *n1* through *n2* use:

> TASK 'IMAGR' ; INP ↵	to review the inputs needed.
> IND m ; GETN n ↵	to specify the input *uv* file.
> CLRONAME ↵	to use the default output names and disk.
> BCHAN *n1* ; ECHAN *n2* ↵	to include a range of channels.
> STOKES 'I' ↵	to make total intensity (unpolarized) images.
> NCHAV 1 ↵	to avoid averaging channels.
> CHINC 1 ↵	to do every channel, setting the channel increment.
> CHANNEL 0 ↵	do not restart.
> CELLSIZ Δx , Δy ↵	to set the image cell dimensions in arc seconds. Cells do not have to be square.
> IMSIZE N_x , N_y ↵	to set the image size in pixels (must be powers of two).
> NITER 0 ↵	to do no Cleaning.
> GO ↵	to run IMAGR.

This makes "dirty" images and beams of the specified channels. If Cleaning is needed, set the Clean adverbs NITER, NBOXES, CLBOX, GAIN, BOXFILE, etc. Note that you must image one IF at a time in order to avoid combining multiple frequencies. The task NOIFS might be useful depending on your data.

Because of the upgrade of the VLA in 2010, IMAGR is more careful with image units. Unless you force

the Clean beam size with BMAJ, BMIN and BPA, IMAGR will image each channel at its "natural" (frequency-dependent) resolution. It now carefully scales each image plane so that it is actually in units of Jy per the Clean beam listed in the header. The Clean beam for each frequency is recorded in a CG table and used by MCUBE and CONVL among other tasks.

At this point, it is a very good idea to determine the noise in your output images and to compare it to the theoretical noise you expect. If your images are significantly noisier than expected, it is a very good idea to stop processing, to think about what may have gone wrong, and then to check and correct that. The noise may be determined using task IMEAN:

> TASK 'IMEAN' ; INP ↵	to review the inputs needed.
> IND *m* ; GETN *n* ↵	to specify the input image file.
> DOHIST TRUE ↵	to plot the histogram.
> NBOXES 200 ↵	to use a significant number of boxes in the plot.
> PIXRANGE = -*x* , *x* ↵	to limit the histogram to the range x, where x should be about 5 times the expected noise.
> DOTV TRUE ↵	to put the plot on the TV rather than in a file.
> BLC *x1, y1, f1* ; TRC *x2, y2, f2* ↵	to select a sub-image of the cube that is free of signal.
> GO ↵	to run the task, plotting on the TV.

This will print two rms's on your message screen one computed using all the data and one done by fitting a histogram to the noise portion of the signal. Especially if the latter fails, the plot will allow you to estimate the true rms, ignoring those pixels significantly above and below the Gaussian noise part of the histogram. Using this plot, it may be better to include all of the data (BLC 0 ; TRC 0 ↵) rather than to limit the number of pixels contributing to the histogram. The AIPS verb IMSTAT does the all-pixel rms computation without the useful plot, while the verb TVSTAT allows you to mark one or more non-rectangular regions on an image on the TV over which the rms is computed. Task RSPEC will plot robust rmses computed on a per plane basis.

8.5 Display and manipulation of data cubes

8.5.1 Building and dismantling data cubes

Many spectral-line display and analysis functions make sense only on 3-dimensional (or more) images. In particular, image creation functions may create one image per spectral channel while spectral analysis requires the images to be ordered with the spectral axis first (most rapidly varying). To combine a number of n-dimensional images into an n- or $n+1$-dimensional image, use the task MCUBE. This task requires the images to have some one physical axis whose value varies between the input images in a manner consistent with a regular axis and it is that axis which is extended to make the "cube." If the output image specified already exists, then the input maps are inserted in the output image in the appropriate places. This allows the n- or $n+1$-dimensional output image to be built up a bit at a time and allows replacement of portions of the image with a corrected n-dimensional image. Type EXPLAIN MCUBE ↵ to receive a variety of hints and suggestions for using this rather general program. As an example, to put 31 frequency channel images into one cube, type:

> TASK 'MCUBE' ; INP ↵	to review the inputs.
> INNA 'N315' ↵	to select the source.
> INCL 'IIM001' ↵	to select the input image class for *all* images.
> INSEQ 1 ; IN2SE 31 ; IN3SE 1 ↵	to set the first and last image and the step in the task's loop over input sequence number. The task handles any missing images gently.

> AXREF 1 ; AX2REF 31 ↵ — to set the pixel coordinate of the first and last image on the third axis in the cube.
> NPOINTS 31 ↵ — to set the total number of points on the third axis.
> OUTN ' ' ↵ — to use the default OUTNAME.
> OUTCL 'LMFCUB' ↵ — to specify, in OUTCLASS, the order of axes.
> GO ↵ — to run MCUBE.

MCUBE will scale each image plane so that it is on the same brightness scale as represented by the Clean beam in the header.

Later on, you might want to replace some of these images by Cleaned images. *E.g.*, assume that you have Cleaned channels 10 through 20, one at a time. These images got the class ICL001. It is a good idea to give them sequence numbers that are the same as the channel numbers, thus 10 to 20. Putting them in the existing cube (make a backup copy with SUBIM if you are nervous) can then be done as follows:

> TASK 'MCUBE' ; INP ↵ — to review the inputs.
> INNA 'N315' ; INCLA 'ICL001' ↵ — to select the clean images.
> INSE 10 ; IN2SE 20 ; IN3SE 1 ↵ — to set the first and last image and the step in the loop.
> OUTN 'N315' ; OUTCL 'LMFCUB' ↵ — to select the existing cube.
> OUTSE 1 ↵
> GO ↵ — to run MCUBE.

IMAGR with its DOTV option allows you to Clean each channel with its own numbers of iterations and its own interactively set Cleaning boxes. There are new options to create boxes automatically on each image plane and an option to control whether the boxes for channel n are carried over to channel $n + 1$.

MCUBE can build a cube out of a set of images that do not have a suitable axis for image building. This can be a set of images that have all the same axis parameters (*e.g.*, identical images except for date) or they can be images at, for example, an arbitrary set of frequencies. Set DOCONCAT = 2 to force MCUBE to make a SEQ.NUM. axis or, if there are different frequencies, an FQID axis with accompanying FQ table. The original coordinate values on an axis that differs from image to image are recorded in the history file and, for frequencies, also in the FQ table. Task FQUBE does this operation more naturally and can combine two cubes each of which already has an FQID axis.

Most programs work on cubes. However, you may find it convenient, on occasion, to work with single image planes. To separate the channels from the cube, use task SUBIM typing, for example:

> TASK 'SUBIM' ; INP ↵ — to review the inputs.
> INNA 'N315' ; INCL 'LMFCUB' ; INSE 1 ↵ — to select the cube.
> OUTN 'N315' ; OUTCL 'IMAGE' ↵ — to give it an output name and class.
> BLC 0 ; TRC 0 ↵ — to select full planes.
> DOWAIT TRUE ↵ — to run in synchronous mode.
> FOR J=10:20 ; BLC(3)=J ; TRC(3)=J ; PRIN J ; OUTSE J ; GO ; END

This runs the program 11 times, taking planes 10 through 20 and creating separate images for them. To put the images back into the cube after they have been modified, use MCUBE.

8.5.2 Transposing the cube

The task TRANS will transpose the cube, ordering the axes in any way you specify. Typically, one transposes images of spectral channels into spectra at image pixels for both display and analysis purposes. Thus:

> TASK 'TRANS' ; INP ↵	to review the inputs.
> INCLASS 'LMFCUB' ↵	to select the untransposed cube.
> TRANSC '-312' ↵	to make new axis order -3, 1, 2 in terms of the old axis order (*e.g.*, RA, Dec, Freq becomes Vel (opposite sign of Freq), RA, Dec).
> OUTCL 'VLMCUB' ↵	to give it an outclass reflecting the axis order.
> BLC 0 ; TRC 0 ↵	to transpose the whole cube.
> GO ↵	to run the program.

Numerous tasks such as IMLIN, XSUM, PLCUB, and XMOM act on the first axis and, typically, make sense only when that axis is frequency/velocity. Note that such x-axis analysis may also be useful in other cases, including angular and time x coordinates.

8.5.3 Modifying the image header

On occasion you may feel the need to modify or add to the information in the image header. For example, to add an alternate velocity description for the frequency axis of a cube, type:

> INDISK *n* ; GETN *ctn* ↵	to select the image.
> AXTYPE 'OPTHEL' ↵	to specify optical-convention velocities relative to the Sun.
> AXREF 16 ↵	to specify the velocity reference pixel (channel 16 is the center of the band in our example).
> AXVAL 5.E6 ↵	the velocity at the reference pixel in m/s.
> RESTF 1420.4E6, 5752 ↵	to specify the line rest frequency in Hz (1420405752 Hz).
> ALTDEF ↵	to add the information to the header.
> ALTSW ↵	to switch between frequency and velocity information in axis labeling, IMHEADER, etc.

The VLA now provides the velocity information to FILLM, making this operation unnecessary in many cases.

Observers may find the Galactic coordinates of their sources to be of interest. To switch the header between Celestial and Galactic coordinates, type:

> CELGAL ↵	to go to galactic coordinates.
> CELGAL ↵	to go back to celestial coordinates.

You may change many of the parameters in the image header and may fetch for use by the *POPS* language all header parameters. The verb PUTHEAD changes a header parameter and GETHEAD gets that parameter. Type EXPLAIN GETHEAD ↵ for the details, including a list of recognized keywords, and EXPLAIN PUTHEAD ↵ for the more restricted list of values which you may change in the header. For example, to move a header parameter from one image to another:

> KEYWORD *'keyword'* ↵	to specify one of the fully allowed keywords.
> GETN *n* ; GETHEAD ↵	to get the header value from image *n*.
> GETN *m* ; PUTHEAD ↵	to put it in the header of image *m*.

8.5.4 Displaying the cube

The easiest way to look at the cube is by using TVMOVIE. This verb loads portions of planes of a cube into the TV memory, and displays them in sequence at a variable frame rate. Type:

> INDISK *n* ; GETN *ctn* ↵ to select the cube.
> INP TVMOVIE ↵ to review the inputs; use defaults for a start.
> TVMOVIE ↵ to load the images and start the movie.

Now follow the instructions on your screen on how to change the transfer function, change speed, look at single frames, etc. Having terminated the movie (button D), you may restart it with

> REMOVIE ↵ to resume the movie.

With the default adverbs, TVMOVIE will zoom the display so that a single frame fills the display area. If you are using a workstation, this may seriously limit the maximum possible frame rate. To use a smaller zoom factor and smaller display area, enter:

> DOCENTER TRUE ↵ to enable the option.
> FACTOR *m* ↵ to use a zoom factor of *m*.
> REMOVIE ↵ to rerun the movie.

Note that you may set these adverbs before the initial TVMOVIE command as well. If your cube has more than 3 axes with multiple pixels, TVMOVIE can display all planes if DOALL = 1.

TVMOVIE uses an image ordering on the screen designed to improve the performance of movies on "real" TV devices. To use a more pleasing ordering (left to right, top to bottom) that is nearly as efficient on workstations, use the TVCUBE verb instead. The verb REMOVIE works on these displays as well. One of the reasons to use TVCUBE is to capture a full TV plane's worth of channels and print them on a PostScript printer. The first step is to get TVCUBE to put the desired channels in one TV image plane. Set TBLC(3) and TTRC(3) to select those planes you want and set TVCHAN = 2 (or the highest TV plane allowed on your system) to force all the images into one TV image. Stop the move with button D, display the full TV memory (use the F2 button), adjust the enhancement (and color) of the display, and then run task TVCPS. If your images are too large for one channel, use TVCHAN = 1 and run TVCPS once for each channel.

Another way to see multiple image planes on a single page is with the task KNTR, which plots a page of contour and/or grey-scale drawings for each plane. This task allows you to include "star" positions and even an outline of the clean beam. A KNTR plot of neutral hydrogen emission from NGC6503 is shown in Figure 8.1.

It can be very informative to transpose the cube to look at position-velocity planes instead of position-position planes. A TVCUBE display of the NGC6503 data cube transposed to 132 (position-velocity-position) order and then captured with TVCPS is shown in Figure 8.2.

Finally, it is useful to look at the cube as a set of spectra. Use TRANS to transpose to 312 order (or -312, 321, and the like). The task XPLOT displays the spectra on the TV or graphics display selecting only those with a sufficiently strong line signal (set by adverb FLUX) and ignoring those with really strong fluxes (controlled with PIXVAL). This can get rather tedious and it is easy to get lost in your cube. The task PLCUB, illustrated in Figure 8.3, allows you to see numerous spectra at one time plotted in correct orientation with respect to their position coordinate. It is necessary to experiment with the adverbs for PLCUB, especially YINC, ZINC, PIXRANGE, and the like, to get a good plot. Use the TV with DOTV TRUE ↵ to save trees.

In general, other display programs work on one plane at a time. Therefore, you must specify which plane in the cube you want to see. For example, if you want to do TVALL on channel 16, which is on pixel 16 of the third axis, you type:

> TBLC 0 0 16 ; TVALL ↵

Note that adverbs TBLC and TTRC are used for TV displays while BLC and TRC are used for image operations.

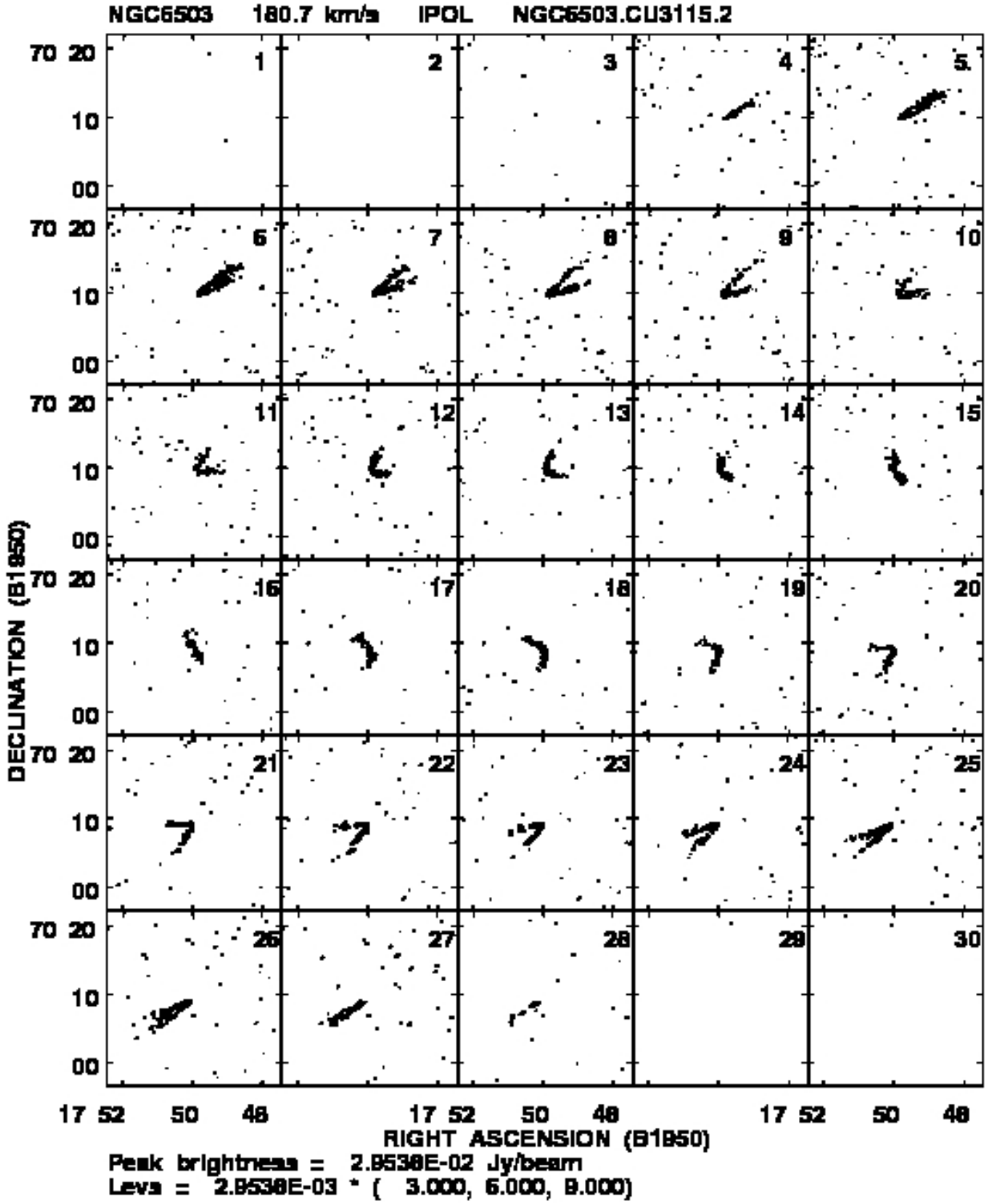


Figure 8.1: Contour images of spectral channels from a cube generated by *AIPS* task KNTR. Data on NGC6503 were provided by Don Wells from observations made with the VLA on 12 April 1983.

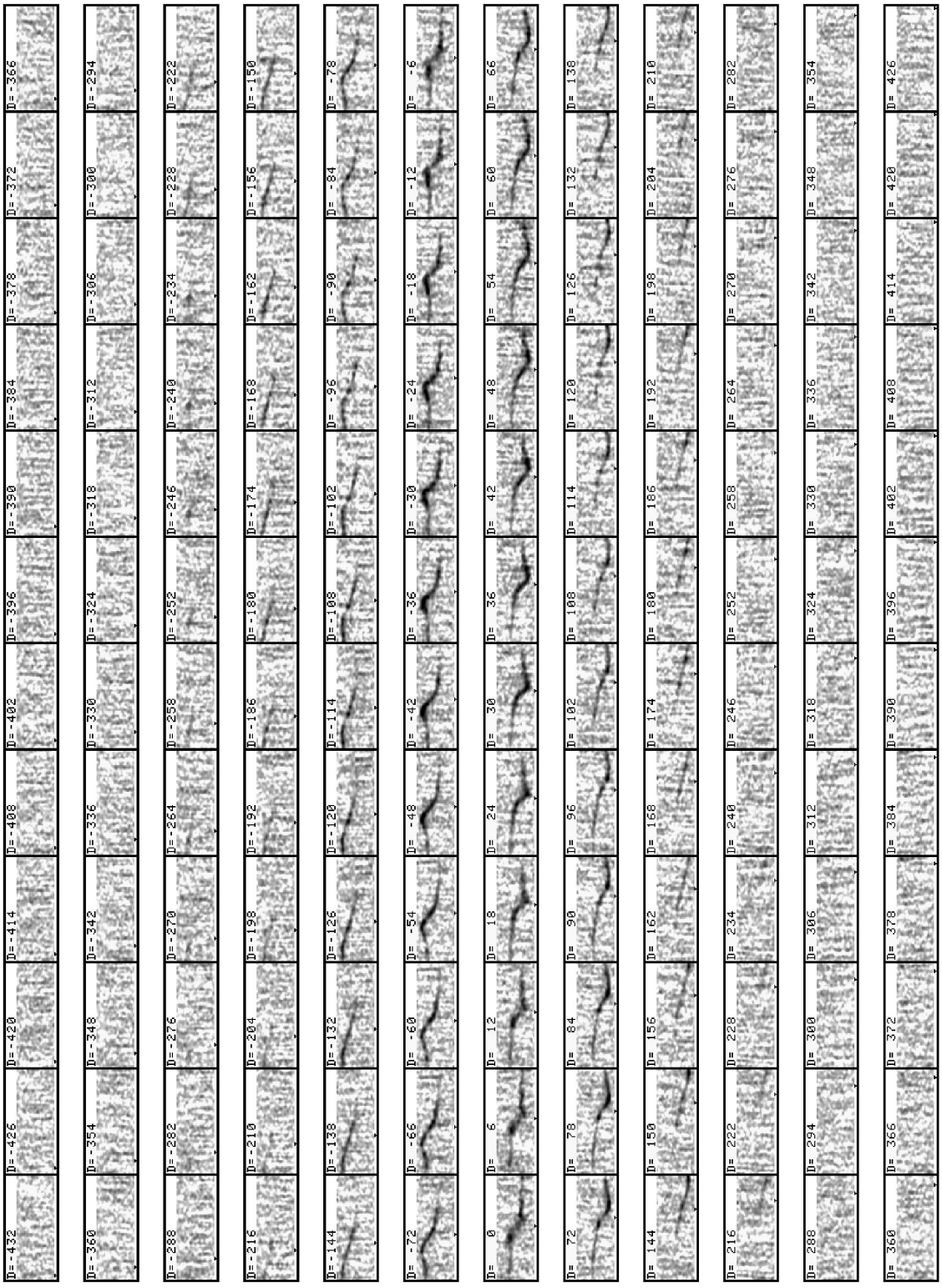


Figure 8.2: Display generated by transposing a data cube into 132 (right ascension, velocity, declination) order and then displaying the central declination position-velocity planes with AIPS verb TVCUBE. The TV display was then captured by *AIPS* task TVCPS for printing on PostScript printers. Data on NGC6503 were provided by Don Wells from observations made with the VLA on 12 April 1983.

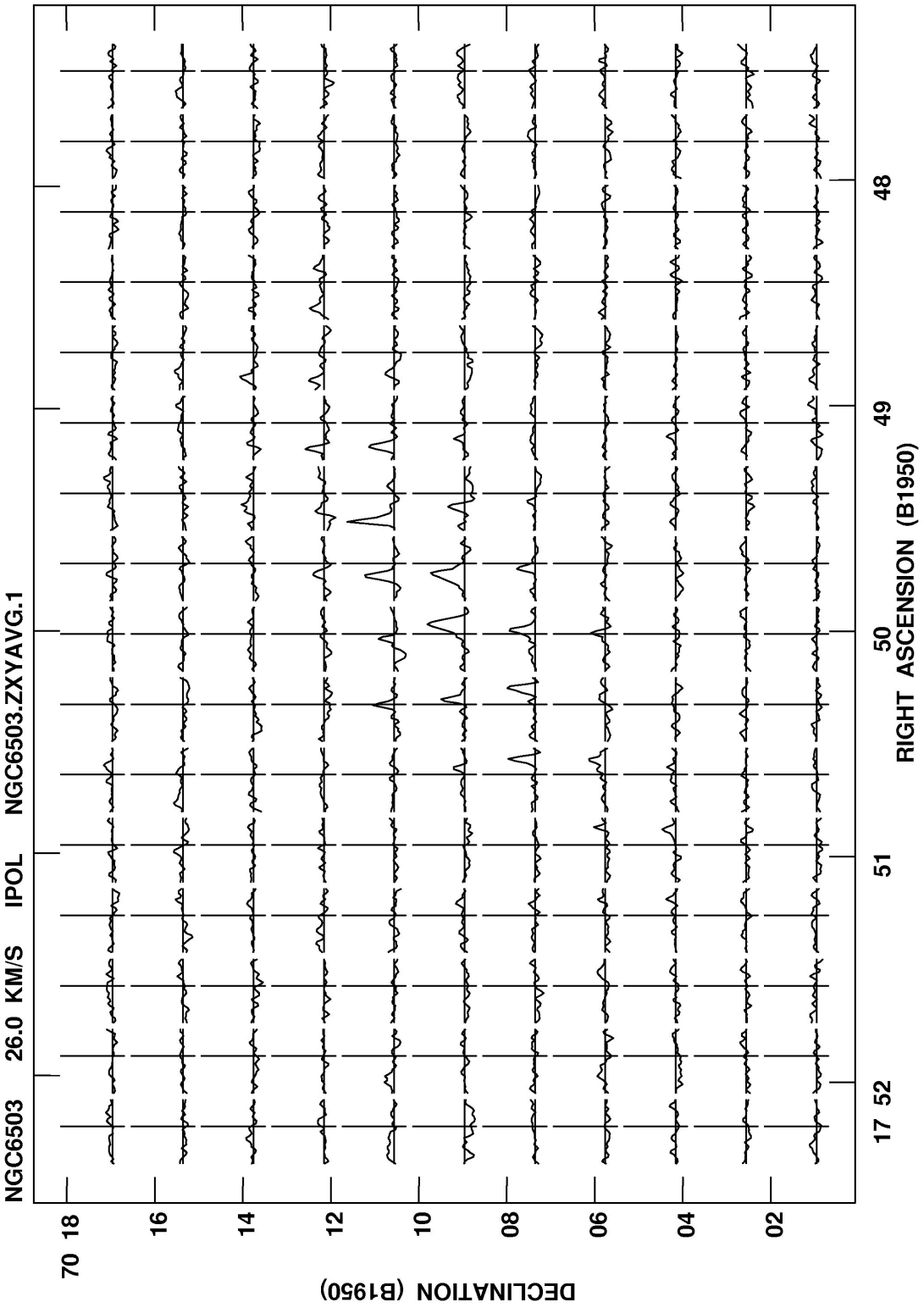


Figure 8.3: Display generated by transposing a data cube into 312 order (velocity, right ascension, declination) and then displaying selected spectra on a regular grid in the angular coordinates with *AIPS* task PLCUB. Data on NGC6503 were provided by Don Wells from observations made with the VLA on 12 April 1983.

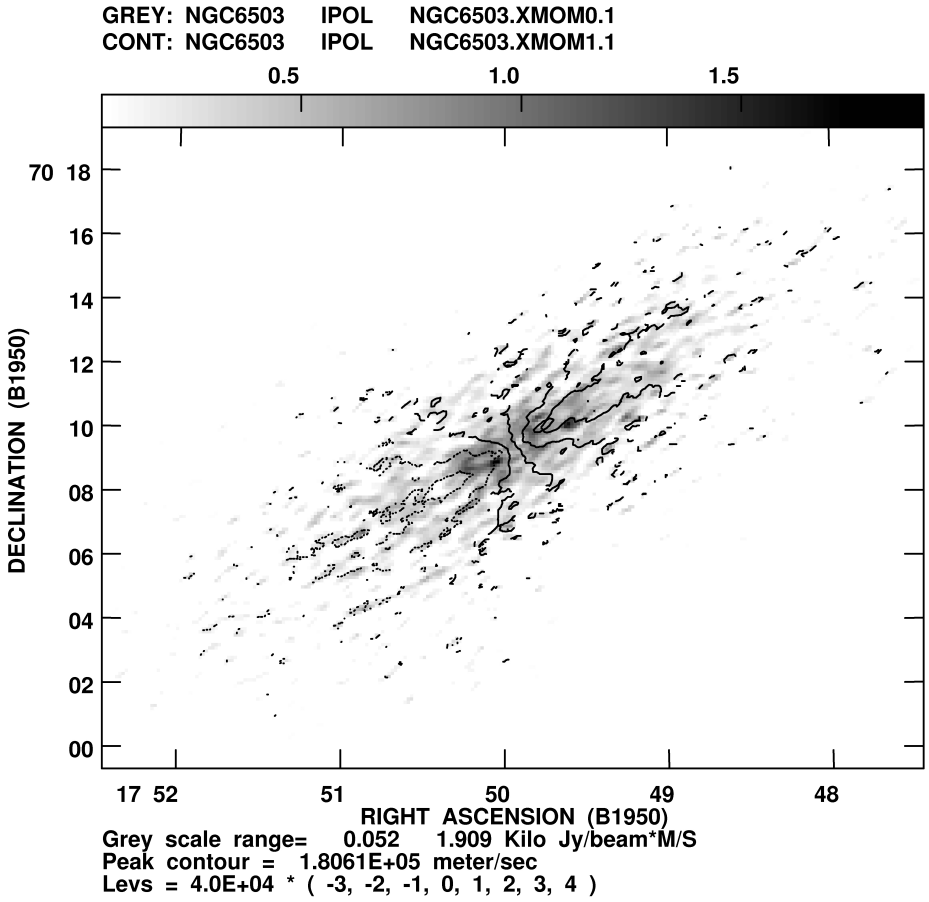


Figure 8.4: Display generated by GREYS and LWPLA using the moment 0 image for grey levels and the first moment image for contours. Data on NGC6503 were provided by Don Wells from observations made with the VLA on 12 April 1983.

8.6 Analysis

A wide variety of programs is available to do further analysis of the data. Exactly which ones you will need depends on the nature of your observations. To see the latest list of symbols related to spectroscopy, enter

> ABOUT SPECTRAL C_R to list them all on your terminal.

or consult Chapter 13 of the *CookBook* for a less current, but paper, copy of the list.

The subject of finding and removing the continuum (aka spectral baseline) was discussed in § 8.3 above. In the image domain, task IMLIN may be used to remove polynomial baselines using the same spectral channels throughout the cube. The task XBASL can be used to remove baselines in an interactive (hence position-dependent) fashion. Be aware, however, that if you have made an error in the calibration, this has most likely caused slopes in amplitude *and* phase. Therefore, it is generally better to track down the error and correct it than to decide (arbitrarily) to take out slopes in (Cleaned ?) image amplitude. Task SQASH may also be used to sum or average planes in a cube, which provides, among other things, a simple but crude way to determine a continuum image which can then be subtracted from the cube by COMB.

Smoothing and blanking are important for almost all analysis programs. CONVL works on cubes and does a spatial smoothing (on position-position-velocity cubes). CONVL will use the CG table to smooth all image

planes to the same spatial resolution. This is preferable to forcing a resolution with BMAJ since it smooths the residuals as well as the components. Using all the defaults in XSMTH performs a Hanning smoothing in velocity (on transposed velocity-position-position cubes) and can be used to do other kinds of smoothing as well. This is not just useful for bringing out weak extended signals. Smoothed images can also assist in determining the boundaries of sources to set windows for subsequent spectral analysis. For example, the smoothed cube could be used to set the CLIP limits in task COMB to be applied to the unsmoothed cube.

In fact, finding all regions of significant line signal may be difficult in large image cubes. The task SERCH offers an algorithm by Juan Uson to find line signals which match specified line widths and exceed a specified signal-to-noise ratio. Histograms may be plotted or printed and a S/N hyper-cube may be written.

The task BLANK offers a variety of algorithms for "blanking" out regions of bad data or source-free regions in spectral-line cubes. It has an interactive mode, which allows you to indicate with the cursor on the TV what are "good" regions. Set everything but the image name to default values, use OPCODE 'TVCU' ↵ and type GO BLANK ↵. Then just follow the instructions, pushing button A to lay out the polygon and button D followed by ↵ to go to the next image. If marking "bad" regions is easier, set DOINV TRUE ↵ before running BLANK. A less subjective way is to use BLANK with OPCODE = IN2C. Blanking is done based on the pixel values in a second cube, usually a smoothed version of the original cube. OPCODE='BOX' simply blanks inside or outside a list of circular or rectangular boxes. Task RMSD can be used to blank an image based on its rms in a window surrounding each pixel. It can also be used to compute images of rms using robust, histogram, or median-absolute-deviation ("MAD") methods.

The blanked cubes can be used to calculate integral profiles with BLSUM and to calculate moments 0 to 3 of the profiles with XMOM. Thus, the 0-moment image will be the integral under the profile (*e.g.*, total HI), the first moment is the velocity field, etc. OPTYPE='MAX' in XMOM will select the peak brightness on each row and make images of it and its coordinate. A comparison of the two types of output can be very illuminating. Task MOMNT does the smoothing, blanking and calculating of moments all in one run. This is very easy to use, but can be dangerous since you don't see what is going on. A display of the $0^{\rm th}$ and $1^{\rm st}$ moment images computed by XMOM is shown in Figure 8.4.

Moment images (or any other 2 or 3 images) may be combined in a display in which one image controls intensity, a second image controls hue (color), and a third optional image controls saturation (color richness or purity, done by varying the ratio of white to pure color). In the days of powerful television display devices, this display could be done at truly interactive speeds in the display hardware. Today, most workstations are capable of displaying full color images and the *AIPS* display program supports them. The verb TVHUEINT will display the two specified TV channels using one as the intensity and the other as hue; see § 6.4.7 *AIPS* task HUINT may also be used with either full-color TV displays. This task offers a small menu of interactive options to enhance the images and otherwise control the display in manners essentially the same as the verb. It however offers the option to write out an image cube having RGB as its third axis. It can also write an image of the 3-color step wedge. These images may then be processed with the usual display tasks such as KNTR and rendered in PostScript by LWPLA. *AIPS* task TVHUI may also be used with either full or pseudo color TV displays. This task also offers a small menu of interactive options to enhance the images and otherwise control the display. It also offers the option to write out an image cube having RGB as its third axis and using much greater mathematical accuracy than is allowed in the TV display. Procedure TV3LOD loads an RGB cube to the TV display with appropriate colors.

RGBMP computes "integral" images another way — as three weighted sums representing the low, center, and high velocity parts of the cube. Like TVHUI, RGBMP writes its results as a cube with RGB as the third axis. Task TVRGB can display these outputs (or any other RGB cube or any three image planes), using one image plane to control the red image, one to control green, and one to control blue. It works on real TV displays and full-color workstations and, using an algorithm to minimize the loss of color information, on pseudo-color workstations. Like TVHUI, it offers a small menu of interactive enhancement options. TVRGB can write 24-bit color PostScript files beginning with the 15JAN96 release. TVCPS is another way to capture the displays generated by TVHUI and TVRGB to send to color PostScript printers.

If you prefer to fit Gaussians instead of calculating moments, the program XGAUS can be used. First, it is a good idea to become acquainted with your data cube. A highly interactive task called TVSPC[2] can let you examine the spectra from one or two transposed cubes at positions selected from an image plane, all on the TV display. The image plane should be some image representing the full field, such as moment-zero or, in polarization, the I polarization image. This task can also display another, non-transposed cube with the displayed plane selected from the displayed spectra. Another highly interactive task called TVIEW[3] lets you examine cubes simultaneously in all 3 transpositions. In earlier releases, use XPLOT first to look at (a sample of) the profiles, before you do any Gaussian fitting. XGAUSS offers a non-interactive mode, but it is frequently unstable and depends on the fit channels and the initial guesses being virtually independent of position. Therefore, in most cases, it is preferable to use the interactive mode, so that you can see what is happening, but be aware that it might be rather time-consuming. XGAUS prepares a table of solutions which can be edited in various ways after all pixels have been fit once. Images of the fit parameters and uncertainties and of the rms of the residual spectra are examined during the editing process and are written out only after you are happy with the results. The task may be re-started with the same table as often as needed. This allows for processing limited regions of the celestial coordinates that have similar spectra, for changing the number of Gaussians fit, and for fitting weaker signals later in the process.

RMFIT is used to fit rotation measures to Q and U spectral cubes, using the cube of Faraday rotation output by FARS to provide initial guesses. It is far more capable than the Faraday rotation synthesis in distinguishing multiple components in the rotation measure. ZEMAN fits Zeeman splitting to spectral cubes of I and V polarization. It can fit the line directly or it can fit the individual Gaussians found by XGAUS individually. The latter allows multiple magnetic-field values to be found in the same direction. See *AIPS* Memo 118[4] for a detailed description of these tasks.

Two more similar tasks have appeared to handle the different mathematics required by absorption-line image cubes. When optical depths become significant, the noise in optical depth becomes a strong function of the optical depth, so the fitting is done in the observed absorption spectra rather than in optical depth even though it is the optical depth which is treated as Gaussian. These tasks, AGAUS and ZAMAN function very much like the tasks for emission-line cubes except that initial guesses are set on a plot of the optical depth spectrum in AGAUS and for details of the mathematics. See *AIPS* Memo 122[5].

Three tasks support the XG, ZE, and RM tables produced by the above family of tasks. XG2PL plots spectra from XG and/or ZE tables, while RM2PL performs a similar function for tables from RMFIT. Task XG2XG copies an XG or ZE table to a new table with a different maximum number of Gaussians.

The cube can be rotated with OGEOM (if the $\alpha - \delta$ pixels are square), *e.g.*, to align one of the axes with the major axis of a galaxy. If the pixels are not square, use OHGEO to re-grid the image instead. A single profile can be produced from these images with SLICE, then plotted using TKSLICE,TVSLICE, or SL2PL (see § 6.5.2, § 6.4.8, § 6.3.2.2, and Figure 6.6). PLCUB, PLROW and XPLOT are convenient programs for displaying multiple individual spectral profiles after the cubes have been transposed.

To compute and display spectral profiles summed over regions in the two angular coordinates, use ISPEC for circular and/or rectangular regions and BLSUM for irregular regions set interactively using the TV display and cursor. Both tasks print their results; both BLSUM and ISPEC make standard *AIPS* plot files, and BLSUM can also make a printer plot. Note that both tasks sum over areas in the first two axes and plot that as a function of position on the third axis, no matter what the three axes actually represent. This suggests a variety of interesting possibilities, such as time functions or line integrals versus position in the source. Both tasks can write the results as slice extension files which may then be printed, plotted and Gaussian

[2]Greisen, E. W. 2025, "Exploring Image Cubes in *AIPS*," AIPS Memo 120 revised, https://www.aips.nrao.edu/aipsdoc.html
[3]Greisen, E. W. 2025, "Further Exploration of Image Cubes in *AIPS*," AIPS Memo 124 revised, https://www.aips.nrao.edu/aipsdoc.html
[4]Greisen, E. W. 2025, "Modeling Spectral Cubes in *AIPS*," AIPS Memo 118 revised, https://www.aips.nrao.edu/aipsdoc.html
[5]Greisen, E. W. 2025, "Modeling Absorption-line Cubes in *AIPS*," AIPS Memo 122 revised, https://www.aips.nrao.edu/aipsdoc.html

fit in a variety of ways. ISPEC can plot the spectral derivative of the profile it has computed, which is of interest in Zeeman splitting experiments. Task RSPEC is similar to ISPEC except that it plots the rms rather than the data. RSPEC has options to write out a signal-to-noise image and/or a text file of channel weights as well.

In 31DEC25, task IM2TX writes out a text file containing a one-dimensional array from the sum, average, or rms from an $N-1$ dimensional region of the image cube. Task TX2IM can read such text files back into *AIPS*. Such one-dimensional images play well with SLICE and SLFIT.

Note that if all you require is a single spectral profile, it may be possible to use POSSM which works on the *uv* data directly. If the spectral profile is a function of position within a well-resolved source, then you will have to go to the image plane.

GAL fits models of galaxy rotation to images of the predominant velocity (*e.g.*, the first moment images written by XMOM, XGAUS, or MOMNT). It accepts a second input image to be used as weights for the first input image; it is common practice to use a the zero'th moment image for this. Plots may be produced. The task MODVF creates a model velocity field based on a user provided model for the rotation curve, orientation, and warping of the plane.

A much more powerful, but tricky to use, task named CUBIT was developed by Judith Irwin and contributed to *AIPS* [6]. This task fits a galaxy rotation and emitter (usually HI) distribution to the full data cube. It uses dynamic memory and so can work on any reasonable size cube. It is much more likely to work if the cube has been very carefully BLANKed and if you approach it with patience. Make the best initial guesses for all the parameters that you can. Then fit for one parameter at a time until you begin to get a reasonable result. Then allow the task to fit all of the parameters at once. And finally, check the fits going through the galaxy in halves or quadrants. But first, study EXPLAIN CUBIT ↵ in detail. The results from this task are worth the trouble to get it to work on your data.

8.7 Additional recipes

8.7.1 Hot banana soufflé

1. Preheat oven to 375° F.
2. Select a 6-cup soufflé dish or other mold and grease it liberally with 1 tablespoon **butter**.
3. Place 6 **eggs**, 1/2 cup **cream**, juice of 1/2 **lemon**, 1 tablespoon **kirsch**, and 1/4 cup **sugar** in blender. Blend until the batter is smooth.
4. Peel 2 large **bananas**, removing any fibers and break into chunks. With blender running, add the chunks one at a time.
5. Break 11 ounces **cream cheese** into chunks and add them to the blender.
6. When all the ingredients are thoroughly mixed, run the blender at high speed for a few seconds.
7. Pour batter into prepared dish and place it in the hot oven. Bake 45–50 minutes until the top is lightly browned and puffy. You may quit when the center is still a bit soft or continue baking until the center is firm.
8. Serve at once. A whipped cream flavored with Grand Marnier makes a nice topping.

[6] Irwin, Judith A. 1994, "Arcs and bridges in the interacting Galaxies NGC 5775/NGC 5774," ApJ, 429, 618-633.

8.7.2 Banana crunch cake

1. Heat oven to 350° F. Grease and flour 10-inch tube (Bundt) pan.
2. In medium bowl, combine 1/2 cup **flour**, 1 cup **cocnut**, 1 cup **rolled oats**, 3/4 cup firmly packed **brown sugar**, and 1/2 cup chopped `pecans`. Mix well.
3. Using fork or pastry blender, cut in 1/2 cup **margarine** until mixture is crumbly. Set aside.
4. In a large bowl, combine 1 1/2 cups sliced very ripe **bananas**, 1/2 cup **sour cream**, and 4 **eggs**; blend until smooth.
5. Add 1 package **yellow cake mix**, Pillsbury Most Supreme is recommended. Beat 2 minutes at high speed.
6. Spread 1/3 of batter in tube pan, sprinkle with 1/3 of coconut mixture. Repeat layers twice more using remaining batter and coconut mixture, ending with coconut mixture.
7. Bake at 350° F for 50 to 60 minutes or until toothpick inserted near center comes out clean. Cool upright in pan 15 minutes; remove from pan. Place on serving plate, coconut side up. Cool completely.
8. HIGH ALTITUDE — above 3500 Feet: Add 3 tablespoons flour to dry cake mix. Bake at 375° F for 45 to 55 minutes.

8.7.3 Panecillos de Plátano

1. Sift together 2 cups **flour**, 1 teaspoon **salt**, and 3 teaspoon **baking powder**.
2. Add 4 tablespoons softened **butter**, mix well, add 3/4 cup **milk**, and stir only until dampened.
3. Roll to 1/2 inch thickness, cut into cookies about 2 inches in diameter, and place on greased cookie sheet.
4. Slice 2 **bananas** in 1/2 inch thicknesses and dip pieces in 2 tablespoons **lemon juice** and then in 2 tablespoons **sugar**. Place a slice on each cookie, pressing it down.
5. Bake in a 425° F oven for 12 minutes or until golden brown.

Thanks to Ruth Mulvey and Luisa Alvarez *Good Food from Mexico*.

8.7.4 Banana-Rhubarb Crisp

1. Slice 2 large `bananas` into 1/4-inch rounds. Combine with $2\frac{1}{2}$ cups diced `rhubarb`, 2 tablespoon `sugar`, 1/4 teaspoon `cinnamon`, and a generous dash `nutmeg`. Spoon the mixture into a well-greased 9-inch pie plate or shallow baking dish (preferably glass or ceramic).
2. In a medium bowl, combine 1/2 cup white or whole-wheat pastry `flour`, 1/2 cup `graham cracker` crumbs, $1\frac{1}{2}$ teaspoons `baking powder`. With a pastry blender or two knives worked in a crisscross fashion, cut in 1/4 cup `butter` until the mixture is crumbly.
3. Combine 1 `egg` lightly beaten with 1/4 cup `milk` and stir into the flour mixture. Spoon the batter as evenly as possible over the fruit mixture. Sprinkle with 2 tablespoons `sugar`.
4. Bake in a pre-heated 400° F oven for 25–30 minutes.

Thanks to *Jane Brody's Good Food Book*.

9 Reducing VLBI Data in $\mathcal{AIPS}$

This chapter describes the reduction of VLBI data in $\mathcal{AIPS}$. A Step-by-step recipe, covering both simple and more difficult situations, is presented. See Appendix C for simpler and shorter recipes suitable for straightforward observations. Procedures to simplify some of the VLBI reduction steps are mentioned and become available to you after you enter the command RUN VLBAUTIL C_R. There is also a VLBA pipeline, VLBARUN, which is useful for simple datasets, see § 9.2. If you have very old VLBI data, in Mk III or Mk IV format, see § O.2. We also include here some background information concerning the structure of VLBI data sets, the data reduction philosophy and a description of some of the effects for which corrections must be determined and applied. It is important to understand these aspects if you wish to reduce your data reliably. For more background information on VLBI data reduction consult *VLBI and the VLBA*, Astronomical Society of the Pacific (ASP) Conference Series No. 82, 1995.

Programs of particular interest for VLBI may be found in Chapter 13 or displayed from inside $\mathcal{AIPS}$ by typing ABOUT VLBI C_R or APROPOS VLBI C_R. Remember, the best and most complete information available on all $\mathcal{AIPS}$ verbs and tasks may be found in their EXPLAIN files. A 15APR97 or later version of $\mathcal{AIPS}$ is required to support full Space VLBI data reduction.

Most types of VLBI data, once read into $\mathcal{AIPS}$, appear very similar in structure as far as the user is concerned. We shall concentrate on describing the reduction path for data produced by the VLBA correlator, but most operations also apply to MkII, MkIII, and MkIV data. See § O.2 where we draw the reader's attention to any differences. The few extra steps necessary for calibrating phase-referencing observations are described in § 9.5.7.4. Note that successful phase-referencing observations require careful planning **before** the observations. See VLBA Scientific Memo No. 24 by J. Wrobel, C. Walker, J. Benson, and A. Beasley.

Some of the VLBI-related tasks require the ability to read files resident outside $\mathcal{AIPS}$. To communicate to $\mathcal{AIPS}$ the directory in which these files exist it is necessary to define a logical pointer or environment variable. Please refer to § 3.10 to see how this is done.

While the majority of VLBI observations are continuum observations, more sophisticated data reduction techniques are increasingly common. Continuum VLBI observers sometimes also apply spectral-line VLBI techniques to improve the dynamic range of their data sets. For these reasons, this chapter is organized to make the discussion of data reduction techniques more uniform. The overview in this section of the steps involved for several type of VLBI data reduction is meant to guide the user through the rest of the chapter. It is strongly recommended that you read the overview carefully before proceeding.

The expected size of the output *uv* data file can be an important consideration in VLBI data reduction. The disk space required by $\mathcal{AIPS}$ for a compressed dataset is given by the relation:

$$\text{Disk Space} = 4 \text{ x } 10^{-6}\ N_{Stokes}\ N_{chan}\ N_{IF}\ \frac{N_{ant}}{2}(N_{ant}+1)\ \frac{T_{expt}}{\triangle T}\ \text{MBytes}$$

where T_{expt} is the total observing time, $\triangle T$ the correlator integration time, N_{Stokes} the number of polarization correlation pairs (RR, LL, RL, LR), N_{chan} the number of spectral channels per IF, N_{IF} the number of IF's, and N_{ant} the number of antennas in the network. Space VLBI (SVLBI) data can have different integration times for the ground and space baselines of $\triangle T_g$ and $\triangle T_s$, respectively, and therefore the total disk space requirement is larger. If N_{ant} is the number of ground telescopes,

$$\text{Disk Space} = 4 \text{ x } 10^{-6}\ N_{Stokes}\ N_{chan}\ N_{if}\ \left[\frac{1}{2}N_{ant}(N_{ant}-1)\frac{T_{expt}}{\triangle T_g} + N_{ant}\frac{T_{expt}}{\triangle T_s}\right]\ \text{MBytes}$$

In uncompressed format the same data set will require two-three times the disk space. Be forewarned that some tasks attempt to create uncompressed scratch files which may not fit into the available disk space. The

amount of available free disk space can be determined using the AIPS command FREE. The blocks referred to in the FREE output are equal to 1024 bytes.

Note that certain operating systems are still subject to a 2-Gigabyte limit for any individual file, as a result of their 32-bit file systems. Larger *AIPS* files are supported on DEC Alpha, SGI (running XFS), HP, Solaris (revision $\geq$ 2.6), and Linux (kernel $\geq$ 2.4.2). The last three require all *AIPS*' C code to be compiled with an additional option. The size of the output file can be reduced by IF selection or limited concatenation in FITLD or by time- or spectral-averaging later using UVAVG, SPLAT or AVSPC.

It is possible to construct data sets on disk that cannot be written to a single tape using FITTP because FITTP uncompresses the data when writing to tape. The task FITAB is designed to address this problem. FITAB writes data in compressed form to tape and can write data in pieces to multiple tapes. Note that FITAB is only available in 15APR99 and later releases and that versions of FITLD from earlier releases cannot read such data. Packages other than *AIPS* may also be unable to understand these files.

One large point of divergence in the reduction of continuum polarization VLBI data is the question of whether or not to determine separate LL and RR phase solutions. The polarization-specific portions of the recipe given below are based upon the premise that L and R phase solutions should always be determined separately on the grounds that it is safer and should work with data from a wide variety of antennas. If the L-R phase offsets for antennas in your data set are small and constant in time, you may consider modifying the recipe in § 9.1 by determining averaged LL,RR phase solutions everywhere except in step 7.

9.1 VLBI data calibration recipe

See Appendix C for simpler and shorter recipes suitable for straightforward observations.

1. LOAD THE DATA For data from the VLBA correlator, run FITLD (§ 9.3.1.1); if needed, follow up with MSORT, USUBA, INDXR, VBGLU, VBMRG, and MERGECAL (§ 9.3.1.4–§ 9.3.1.7). For data from a MkIII correlator, see § O.2. Data from the Penticton correlator should be loaded using FITLD, sorted (MSORT, § 9.3.1.4), and indexed (INDXR, § 9.3.1.6).

   POLARIZATION: The combination of the VLBA correlator and FITLD incorrectly labels polarizations for dual parallel-hand correlation (RR and LL only), even if RR and LL are in different frequency bands (*e.g.*, LL at 5 GHz and RR at 8.4 GHz). For these types of data, you must run FXPOL (§ 9.3.1.8).

2. EXAMINE THE DATA It is important to familiarize yourself with the data set before proceeding further, especially if you have little experience with VLBI data. There are many *AIPS* tasks for the examination of your data (see § 9.4 for a fuller discussion). Minimally, you should at first run LISTR, IMHEAD, EDITR, POSSM, VPLOT, and PRTAN. At later stages you will probably find SNPLT, PRTAB, DTSUM, and SHOUV useful for examining data and calibration tables.

   SVLBI: Task OBPLT allows you to examine different aspects of the spacecraft orbit.

3. PROCESS THE CALIBRATION FILES You will have either received calibration files, or instructions on where to obtain them. Some calibration files can be automatically processed into a form suitable for use within *AIPS* using VLOG (§ 9.5.2). ANTAB is now the primary *AIPS* task for loading calibration information from log files.

   VLBA CORRELATOR: The VLBA correlator will usually attach calibration information directly to your data for all VLBA and some other antennas. This obviates the need to run VLOG, ANTAB, PCLOD, and UVFLG to process your *a priori* calibration information for VLBA antennas. Some information for non-VLBA antennas must usually still be loaded from text files.

   POLARIZATION: Be careful to make sure that the polarization labeling of the IFs in the calibration text files is the same as the labeling in the data.

4. CORRECT FOR THE IONOSPHERE For low frequency experiments (< 3GHz), TECOR should be run to remove at least part of the ionospheric contribution to the phase offsets. This should also be considered for higher frequencies (*e.g.*, 8 GHz) for abnormal ionospheric conditions.

5. CORRECT FOR THE EARTH ORIENTATION PARAMETERS For phase referencing experiments correlated at the VLBA correlator, particularly between 5-May-2003 and 2-August-2005, CLCOR (OPCODE='EOPS') should be run. This will correct the possibly inaccurate Earth Orientation Parameters used by the VLBA correlator. This is particularly important for astrometry experiments but can effect any phase referencing experiment including those correlated outside the above range of dates.

6. EDIT THE DATA Identifying and editing bad data now can save you time later. Data should first be edited using UVFLG to apply editing information supplied with your calibration files (§ 9.5.3). Some useful tasks for examining and editing data are EDITR, UVFLG, TVFLG, SPFLG, EDITA, BPEDT, FLAGR, FINDR, VPLOT, and QUACK.

   POLARIZATION: You may want to edit the data consistently in all polarizations (select STOKES = 'IQUV' within EDITR, TVFLG or SPFLG) — this can greatly simplify the imaging stage (see step 15).

7. POLARIZATION: ADD PARALLACTIC ANGLE CORRECTION For alt-az mounted antennas, a parallactic angle correction for the rotating orientation of the antenna feeds with respect to the observed source must be performed as the first step in the phase calibration using CLCOR (§ 9.5.4.1). This step should be performed no later than immediately after the T_{sys} calibration.

   PHASE REFERENCING: You *will* want to perform the parallactic angle correction described above for phase referencing observations even if you only correlated the parallel hands (RR, LL).

8. APPLY SAMPLER CORRECTION **We now advocate a new amplitude calibration strategy based on VLBA Scientific Memo #37 (Walker 2015). This strategy interleaves the classic *a priori* calibration with instrumental delay and bandpass calibration to improve the calibration of data from the new Roach Digital Backend (RDBE) on the VLBA (see VLBA Observational Status Summary for a description of the RDBE and VLBA Scientific Memo #37 for a discussion of amplitude problems when using the RDBE).** With data from before the RDBE you can use either the old or new strategy. First corrections for sampler biases should be applied using ACCOR (§ 9.5.4.2) for data from the VLBA and some other correlators. Some correlators apply this correction to the data before writing them out — notably the EVN JIVE correlator and correlators used for the Australian LBA. The VLBA hardware and software (DiFX) correlators do *not* apply this correction to the data. Therefore, ACCOR is required for the VLBA correlators and any others that do not apply the correction. ACCOR should be benign (do nothing) for those correlators that do apply the correction prior to reading the data into *AIPS*. Note that you can always run ACCOR and look at the SN table produced with PRTAB or SNPLT to see if it was benign or not.

9. CALIBRATE THE INSTRUMENTAL DELAYS *Phase-cal*s, or measured single-band and multi-band instrumental phase errors, should be applied using PCCOR (§ 9.5.4.3). You can manually perform a phase-cal by running FRING on a limited subset of your data to account for missing phase-cal information or to refine the reported phase-cal measurements (§ 9.5.4.4).

   SPECTRAL-LINE: Delay calibration should be carried out only on the continuum sources at this stage. Since there should be no pulse-cals, the "manual" phase-cal method should be used.

   POLARIZATION: If running FRING to determine the instrumental delays, be certain to solve for independent left- and right-polarization delay solutions APARM(3) = 0. Run RLDLY after calibrating the instrumental delays, to determine a single delay offset between left and right polarization (§ 9.5.7.11).

10. CALIBRATE THE COMPLEX BANDPASS RESPONSE FUNCTION Run BPASS or CPASS to determine the bandpass response function using the *cross-power* spectra (§ 9.5.4.5). The normalization

should be over the full bandwidth, be careful because the default channel selection is the inner 75% of the band. If you use the inner 75%, it can lead to amplitude errors of up to 15%.

SPECTRAL-LINE: The bandpass response function should be determined using only the continuum calibrator sources.

11. APPLY ONE MORE AUTOCORRELATION CORRECTION AND THEN FINISH *A PRIORI* CALIBRATION After determining the bandpass calibration the autocorrelations are probably different from unity by a few percent. To correct this run the task ACSCL, applying the previous calibration and bandpass correction. Then finally use APCAL to complete the *a priori* amplitude calibration (§ 9.5.4.6) — this is called the T_{sys} method of amplitude calibration. APCAL can also be used to perform opacity corrections.

    SPECTRAL-LINE: Unless the line emission is very weak, you may wish to defer amplitude calibration of your *line sources only* until step 16 below. The template method described there is much more accurate than the T_{sys} method.

12. FRINGE FIT THE DATA Estimate and remove residual delays, rates and phases using FRING or BLING and CLCAL (§ 9.5.7.7–§ 9.5.7.8).

    SPECTRAL-LINE: Only fringe-fit the calibrator source at this stage. Check the coherence of the target source using the resulting solutions to decide whether or not to zero the rate solutions using the 'ZRAT' option in SNCOR (§ 9.5.7.10). It is probably better to force the rates to be zero in FRING rather than zeroing them after the fitting.

    PHASE REFERENCING: You should **not** fringe-fit on the *target*, or phase-referenced source. Rather, you should fringe-fit on the *cal*, or phase-reference calibrator. When you apply the solution, be sure to set the CALSOUR and SOURCES adverbs in CLCAL appropriately to interpolate the solutions for the cal source onto the target source (see § 9.5.1.2). If you are not interested in astrometric calibration and your target source is strong enough, you may wish to consider fringe-fitting on it to further refine the phase calibration (§ 9.5.7.4).

13. POLARIZATION: ESTIMATE THE INSTRUMENTAL POLARIZATION . Correct for the instrumental polarization terms, commonly known as 'D-terms' using PCAL or LPCAL on the polarization calibrator (§ 9.5.7.13). This polarization calibrator should first be fully calibrated and imaged before this step can be performed.

14. POLARIZATION: CALIBRATE THE POLARIZATION POSITION ANGLE If a calibration source with known polarization orientation is available, use CLCOR to make a final correction to adjust the polarization angles of the target source data (§ 9.5.7.11).

    SPECTRAL-LINE or POLARIZATION: The bandpass response function should be determined using only the calibrator source. Unlike step 7, this step cannot be skipped.

15. APPLY THE DOPPLER CORRECTION (spectral-line data only). Run CVEL to compensate for the changing Doppler shifts of the antennas with respect to the source during the observation and between the different observations (§ 9.5.5).

16. SPECTRAL-LINE: REFINE THE AMPLITUDE CALIBRATION Run ACFIT to amplitude calibrate the program source using the template spectra method (§ 9.5.6). Note that the traditional T_{sys} method (§ 9.5.4.6) can also be used if the line emission is too weak for the template method to work successfully.

17. SPECTRAL-LINE:DETERMINE RESIDUAL RATES Now estimate the residual rates ONLY by running FRING or BLING on one or a few spectral points on the target source (§ 9.5.7.10).

18. APPLY CALIBRATION, AVERAGE, AND INSPECT THE FINAL DATA Run SPLIT or SPLAT to apply the calibration solutions and to average the data in frequency if appropriate (§ 9.6.1), and UVAVG to average the data in time (§ 9.6.2). You can also run SPLAT to combine these three operations into a single step. It is recommended that you take the time to inspect the calibrated data to see if more editing is needed, and to check that no gross calibration errors remain in the data (§ 9.6.3).

19. SELF-CALIBRATE/IMAGE OR SELF-CALIBRATE/MODEL-FIT THE DATA The final complex gain corrections are determined by iterating self-calibration with imaging of the resultant data set. This is called hybrid-mapping. Alternatively, self-calibration can be iterated while fitting models directly to the data — the goal is to self-calibrate using the best model possible. The options are outlined in § 9.7.

    SPECTRAL-LINE: One final distinction remains between continuum and spectral-line data. Only one or a few spectral points are used to determine final complex gain corrections which are then applied to all spectral points in the line data. After applying these gains, the line source data can be imaged to form an image cube.

    POLARIZATION: While the Stokes I and Stokes V images formed using the RR and LL visibilities will be real-valued, the Stokes Q and Stokes U images formed using LR and RL visibilities can, in principle, be complex-valued. You must use a fully complex imaging and deconvolution technique (see the HELP files for CXPOL, CXCLN, and CXIMAGR) or you can simply edit the LR and RL visibilities to enforce the condition that whenever you have a RL visibility on a baseline, you also have the LR visibility on the same baseline; this ensures that the Stokes Q and U images are real-valued and allows you to use the standard imaging tasks.

9.2 Pipeline for the VLBA

VLBAPIPE and VLBARUN are two similar pipeline procedures which use the VLBA calibration procedures (from VLBAUTIL) and some logic to calibrate and image VLBA data. They both attempt to make intelligent decisions on defaults, so they can be run fairly automatically, if the names of the sources are known. VLBAPIPE and VLBARUN have been given options to flag data on elevation, to flag excessive fluxes with CLIP, and to apply auto-flagging with RFLAG. The option to clip out bad system temperatures, replacing them with good ones, with TYSMO was also added. The two pipelines differ primarily in the order in which they carry out their operations. Most importantly, VLBAPIPE runs FRING on all calibration scans before computing the bandpass. If the primary calibrator was onserved with multiple scans, this order should help to make the scans phase coherent. If desired, VLBAPIPE and VLBARUN will produce diagnostic plot files and write them to disk creating an HTML file to ease examination of these files. Images will be produced, optionally including self-calibration of the calibration sources. Since only one self-cal, phase only iteration is run, the images should be considered diagnostic in nature.

The two piplines use essentially the same input adverbs, except for the meaning of the DOFLAG parameter. Sample inputs for procedure VLBAPIPE are:

> RUN VLBAUTIL $^{C}_{R}$	to acquire the procedures used by VLBAPIPE.
> RUN VLBAPIPE $^{C}_{R}$	to acquire VLBAPIPE.
> DATAIN */dirname/data.fits* $^{C}_{R}$	to set data file to load from disk.
> OUTDISK *n* $^{C}_{R}$	to set disk for output.
> OPTYPE 'CONT' $^{C}_{R}$	to say this is a continuum data set.
> CLINT 0 $^{C}_{R}$	to use default.
> CHREFANT 'FD' $^{C}_{R}$	to set reference antenna to Fort Davis.
> TIMERANG 0 $^{C}_{R}$	to have VLBARUN determine a good instrumental delay calibration scan.

> INVER 0 ↵ — PC table to use, -1 if you want to force manual phase cal.
> CALSOUR 'CAL-BAND', 'CAL-PHASE', 'CAL-AMP' ↵ — to list calibrators, bandpass calibrator *must* be first.
> SOURCES 'CAL-PHASE', 'TARGET', ... ↵ — to list phase referencing and target pairs.
> CHINC -1 ↵ — to determine channel increment based on the data channel width.
> INFILE '' ↵ — set if you want to apply a zenith delay file produced by DELZN.
> SOLINT 0 ↵ — to use default.
> IMSIZE 512 ↵ — to make images and specify size of target images.
> FACTOR 0 ↵ — to make calibrator images 128x128.
> DOCLIP fp, fc ↵ — to run CLIP to flag a sample when the parallel-hand flux is above $|fp|$ or the cross-hand flux is above $|fc|$. If fp is positive, the clip is done after amplitude calibration; if negative, it is done before the bandpass and amplitude calibration. The values for fp and fc are different in these two cases.
> DOFLAG 1 ↵ — to run RFLAG on each source individually.
> TYSMOLEV 100 ↵ — to clip system temperatures above 100 K with median-window filtering at 10 K and replacing the clipped values with nearby good ones.
> ELEVLIM 15 ↵ — to flag out all data for which the elevation is < 15 degrees.
> GET2N *ctn2* ; NMAPS=1 ↵ — to use a model for the bandpass calibration.
> SELFCAL 1 ↵ — to do one pass of self-cal, phase only on each calibrator prior to the final SPLIT¿
> BPASS5 -2 ↵ — to normalize bandpass scans before the fit. Default is +1 to normalize the bandpass functions themselves without normalization of the data.
> BPSOLINT 0 ↵ — to make a solution for each calibration scan. BPSOLINT=-1 averages all calibration scans.
> DOBAND 3 ↵ — to interpolate the bandpass over time. DOBAND=1 (the default) averages all bandpass records.
> CALTASK 'CALIBS' ↵ — to use FRING for the initial time series calibration but use CALIB for any self-cal. Other choices are 'CALIB' and 'FRING' (the default) to use the specified task for all time series calibrations.
> DOPLOT 1 ↵ — to make some diagnostic plots; level 2 makes an awful lot of plots.
> OUTFILE '*/directory/name* ↵ — to specify directory for output plots. If this is set, plots are written out from AIPS and organized in an HTML file for easy viewing. The plots, HTML file, and message file contents are put in a dated subdirectory below the specified directory.
> OUTTEXT '*email@somewhere.edu* ↵ — to specify an e-mail address if the user wants to be notified when the pipeline is done.
> BADDISK 0 ↵ — to specify which disks to avoid for scratch.
> VLBAPIPE ↵ — to run the procedure.

VLBAPIPE or VLBARUN will then run and produce the requested number of diagnostic plots. For details on the plots produced for each level of DOPLOT, see EXPLAIN VLBAPIPE. If an e-mail address is specified then a VLBAPIPE DONE or VLBAPIPE FAILED message will be sent. However, the VLBAPIPE FAILED message will only be sent if the pipeline failed because of problems with the inputs, if it failed because a task it was running failed this message is not sent. It is highly recommended that the user read the explain file for

VLBAPIPE before use.

The only adverb that differs between the two pipeline scripts is DOFLAG. In VLBAPIPE, the pipeline will shut down after running RFLAG if $0 <$ DOFLAG < 10 to allow you to rerun the pipeline using the RFLAG flag tables. Presumably, the calibration in the second pass should be better. If DOFLAG ≥ 10 in VLBAPIPE or DOFLAG> 0 in VLBARUN the pipeline invokes RFLAG and then continues to the optional self-cal step followed by SPLIT and IMAGR. VLBARUN uses the value of DOFLAG as an initial clip level in RFLAG but only if it is greater than 100. VLBAPIPE does not use the initial clip option in RFLAG, leaving that to the DOCLIP adverb.

Note that editing with CLIP is always a good idea. RFLAG appears to work well at low frequencies but to be overly enthusiastic at C-band and above. Note that the flagging is done after all calibration steps, so you may examine your data with and without the new flags to determine their correctness. RFLAG is run using baseline-dependent clip levels and so should work with arrays containing high-sensitivity telescopes as well as the VLBA. You may wish to use VLBAPIPE without RFLAG (or to discard the flags produced by RFLAG) and then run SPFLG to flag the visibility data interactively. Then use VLBAPIPE over again applying the manually produced flags. Procedure R_RESTART (P_RESTART in VLBARUN) will clear away all plot and calibration tables from your data set, leaving any flag tables. Do not delete version one of the CL and FG tables since they contain the initial calibration and on-line flagging information.

A similar RUN file named VLBAMPHC is available to calibrate, flag, and image data sets for secondary phase stopping centers when DiFX has been used to correlate multiple centers from a single pointing.

In 31DEC21, a similar pipeline was constructed to inspect autocorrelation data for RFI. VLBARFI reduces the data set using many of the steps of VLBARUN. After the amplitude calibration, the pipeline writes out the autocorrelation data with SPLAT and then runs VBRFI to analyze the statistics of the autocorrelations and to write then into a text file and optionally to plot them. Task PLRFI may be used later to plot one or more of these output text files. In 31DEC22, VLBARFI may be run with no calibration and statistics determined with medians using VLBRF rather than VBRFI.

9.3 Loading, fixing and inspecting data

In theory, *AIPS* can process data from multiple frequency bands (FQ numbers in *AIPS* parlance) coexisting within the same data set. However, it is recommended that the data be separated into different frequency bands as soon as possible after loading the data and process each FQ number separately. If you wish to do this, you should do it immediately after performing the relevant steps in §9.3. The VLBAUTIL procedures VLBAFQS and VLBAFIX do this automatically (VLBAFIX is recommended and does other "fixing" tasks like fixing subarrays etc.). If you want to do this by hand use the task UVCOP.

9.3.1 Loading data from the VLBA correlator

9.3.1.1 Running FITLD

The information below applies to data from the VLBA Correlator in what the VLBA archive calls "raw" format, more formally known as FITS-IDI format. The archive now also contains data called "calibrated" which have been run through a few of the simplest VLBA procedures but which are nowhere near calibrated. These files are recommended under normal circumstances. They may also be loaded with FITLD, but the DOCONCAT option does not work with the FITS table format used in the archive.

Data generated by the VLBA correlator are loaded from DAT (or Exabyte) tape (or from disk files) into *AIPS* using FITLD. First, physically load your tape and MOUNT it (§3.9), then run FITLD. Often the data on

your tape will be divided into a number of separate files (corresponding to separate "correlator jobs"). In this case, run FITLD with NCOUNT set equal to the number of files on the tape (or a suitably large number), as listed on the paper index which comes with the tape. The adverb ANTNAME allows the user to control the antenna numbering if desired. Also set DOCONCAT = 1 ↵ to ensure that all tape files with the same structure are concatenated into a single *AIPS* file. Note that standard tape handling tasks (*e.g.*, PRTTP and TPHEAD) can be used to inspect the tape contents.

Note that antennas, sources, frequency IDs, and other things may be numbered differently in different correlator jobs. FITLD fixes all this for you, but only if you set DOCONCAT = 1 and, better still, load as many files as possible in each execution of FITLD. FITLD can load VLBA correlator data from multiple disk files so long as they have the same name plus a consecutive post-pended number beginning with 1. If you forget to put all the related data together with FITLD you can use MATCH to align the antenna numbers followed by DBCON later.

Typical inputs to FITLD would be:

> TASK 'FITLD' ; INP ↵	to review the inputs.
> INTAPE *n* ↵	to specify the input tape number.
> NFILES 0 ↵	to skip no files on tape.
> DATAIN ' ' ↵	to load from tape, not from disk.
> OUTNAME 'TEST' ; OUTCL 'FITLD ' ↵	to specify the name of the output file.
> OUTSEQ 0; OUTDI 1 ↵	to specify the sequence number and disk of the output.
> OPTY ' ' ↵	to load any type of file found.
> NCOUNT 20 ↵	to load 20 tape files.
> DOUVCOMP 1 ↵	to save disk space by writing compressed data.
> DOCONCAT 1 ↵	to concatenate files with same data structure into one disk file.
> CLINT $\triangle t$ ↵	set CL table interval to $\triangle t$ minutes (see discussion below).
> DIGICOR 1 ↵	to request digital corrections (usually VLBA correlator only).
> DELCORR 1 ↵	to request delay decorrelation corrections (VLBA correlator only).
> WTTHRESH 0.65 ↵	flag incoming visibilities with correlator weights less than 0.65.
> DOWEIGHT 1 ↵	to have the data weights include channel bandwidth and integration time as well as correlator weights.
> SOURCES ' '; QUAL 0 ↵	to accept all sources found.
> TIMERANG 0 ↵	to accept data from all times.
> BCHAN 0; ECHAN 0; BIF 1; EIF 0 ↵	to accept all channels in all IFs.
> SELBAND 0 ↵	bandwidth to select (kHz).
> SELFREQ 0; FQTOL 0 ↵	frequency to select with tolerance of 10 kHz.
> OPCODE ' ' ↵	to not copy the tape statistics table ('VT' table).
> GO ↵	to run the program.

This may seem a bit formidable. For straightforward VLBI observations, there is a collection of procedures to simplify matters including the loading of data. Enter

> RUN VLBAUTIL ↵	to acquire the procedures; this need be done only once since they will be remembered.
> INTAPE *n* ↵	to specify the input tape number.
> NCOUNT 20 ↵	to load 20 tape files.
> OUTNAME 'TEST' ; OUTDI 1 ↵	to specify the name and disk of the output file.
> DOUVCOMP 1 ↵	to save disk space by writing compressed data.

> CLINT $\triangle t$ C_R to set the CL table interval to $\triangle t$ minutes (see discussion below).

> INP VLBALOAD C_R to review the inputs.

> VLBALOAD C_R to run the procedure.

Because the data files tend to be very large, you will usually write compressed data (DOUVCOMP=1). These files take about 1/3 of the space of 'uncompressed' data sets, but cause information about the weights of individual polarizations, spectral channels, and IFs to be lost. There is some loss in dynamic range and sensitivity when the weight information is (partially) compromised. (See Appendix F for an expanded discussion of when to and when not to write 'compressed' data sets.) If your observation has more than one DAT or Exabyte tape, simply run FITLD for each tape. Setting DOCONCAT 1 and setting the output file name completely will ensure that the data from separate tapes with compatible observing band/data structure will be appended to existing *AIPS* files. Generally, after loading all of your data, you will have one file for each such observing band and/or observing mode. However, observations which require multiple passes through the correlator will have one file per observing mode *per correlation pass*. Data from separate correlator passes can be concatenated using task VBGLU and/or merged with task VBMRG.

Adverb CLINT, which specifies the CL table time sampling interval, must be short compared to the anticipated coherence time. CLINT should be set such that the shortest anticipated fringe-fit interval is spanned by a few CL entries. Time sampling in the CL table that is too coarse can lead to calibration interpolation errors when applying the fringe-fit solutions at later stages of the data reduction. If the interval is made unnecessarily short the CL table may become unmanageably large.

It is recommended that corrections for digital representation of the correlated signals be performed in FITLD under control of adverb DIGICOR, but only for data from the VLBA correlator. DIGICOR should be set to one for all continuum and nearly all spectral line experiments. Set DIGICOR to 3 or 4 if the digital corrections are desired for a non-VLBA correlator, *e.g.*, some versions of the DiFX correlator. In the special case of spectra with very strong narrow features, the absence of correlator zero-padding may limit the accuracy of the quantization corrections. See the FITLD help file for further information. The details of digital correction for FX correlators can be found in *Radio Science* **33**, 5, 1289–1296, "Correction functions for digital correlators with two and four quantization levels", by L. Kogan.

Adverb DELCORR enables amplitude corrections for known delay decorrelation losses in the VLBA correlator, as described in *AIPS* Memo 90 (1995, "Delay decorrelation corrections for VLBA data within *AIPS*" by A. J. Kemball). Setting DELCORR=1 will create a correlator parameter frequency (CQ) table for each file written by FITLD. Do this for the VLBA correlator only. The presence of this table enables the delay decorrelation correction once the residual delays have been determined in fringe-fitting. These corrections will not be applied if the data were not correlated at the VLBA correlator or if the CQ table is missing. For older FITLD files the CQ table can be generated using task FXVLB and this must be done before any changes in the frequency structure of the file are made. The CQ table is used for rate and delay amplitude decorrelation corrections after residual delay and rate errors have been determined by fringe-fitting, and are being applied to the data. The CQ table has no immediate effect on the data written by FITLD but is essential for later processing.

The WTTHRESH adverb can be applied to drop incoming data with playback weights less than the specified limit. Note that data flagged in this way are *unrecoverable* except by re-running FITLD. The data weights are normalized to unity so good data usually have correlator weights close to 1.0. If DOWEIGHT > 0, the actual data weights will be scaled by twice the product of the channel bandwidth in Hz and the integration time in seconds. This should make the weights approximately one over the rms squared. You should examine your data carefully if you use WTTHRESH to make sure that you have not discarded too much data at this stage. Typically 0.8 or higher is good for the VLBA, but for non-VLBA stations a lower value such as 0.6 or 0.7 may be appropriate.

Calibration data have been transferred from the correlator with your data if your data include VLBA antennas and were correlated after 1 April 1999 and before late 2009, when the DiFX correlator

came on line and your IMHEADER listing shows the presence of GC, TY, WX, PC and FG tables, as in the example below. If you loaded more than one tape file, you must merge the calibration tables. VLBALOAD does the merging for you. See §9.3.1.2 for additional details. Note that, as this example shows, it is possible your data have calibration transfer tables even though they were correlated before 1 April 1999. If your IMHEADER does not show GC and TY tables, you do not have calibration transfer and must manually load calibration information in from text files. Also, even if you have calibration transfer, you may still have to manually load calibration information for some non-VLBA antennas (see https://science.nrao.edu/facilities/vlba/calibration-and-tools/caliblogs for some information in this regard).

The output files produced by FITLD are in standard multi-source format (as described in §4.1) and contain data from all the target and calibrator observations in your observation. FITLD also writes a large number of extension tables including an index (NX) table, and many tables containing calibration information. A description of the VLBA correlator table types is given in §9.8. If you are missing the CORR-ID random axis, your *AIPS* release is stale (pre-15APR97) and you are strongly encouraged to upgrade to the latest release; much of the information presented in this chapter will not be usable with pre15APR97 releases of *AIPS*. Your catalog header should be similar to the one, obtained using verb IMHEADER, given below. If you have GC, TY, FG, WX, and PC tables as in this example data header, your data were processed with calibration transfer - see §9.3.1.2 for more details.

```
Image=MULTI     (UV)         Filename=329         .OVLB  .   1
Telescope=VLBA               Receiver=VLBA
Observer=TM008               User #=   44
Observ. date=23-SEP-1998     Map date=06-JAN-1999
# visibilities      6567     Sort order  **
Rand axes: UU-L  VV-L  WW-L  TIME1  BASELINE  SOURCE  FREQSEL
           INTTIM  CORR-ID  WEIGHT  SCALE
-----------------------------------------------------------------
Type    Pixels   Coord value     at Pixel     Coord incr   Rotat
COMPLEX      1   1.0000000E+00       1.00  1.0000000E+00     .00
STOKES       1  -2.0000000E+00       1.00 -1.0000000E+00     .00
FREQ        16   4.9714900E+09        .53  5.0000000E+05     .00
IF           8   1.0000000E+00       1.00  1.0000000E+00     .00
RA           1    00 00 0 .000       1.00        .000000     .00
DEC          1    00 00 0 .000       1.00        .000000     .00
-----------------------------------------------------------------
Coordinate equinox 2000.00
Maximum version number of extension files of type HI is   1
Maximum version number of extension files of type CQ is   1
Maximum version number of extension files of type AT is   1
Maximum version number of extension files of type IM is   1
Maximum version number of extension files of type CT is   1
Maximum version number of extension files of type GC is   1
Maximum version number of extension files of type TY is   1
Maximum version number of extension files of type FG is   1
Maximum version number of extension files of type PC is   1
Maximum version number of extension files of type MC is   1
Maximum version number of extension files of type OB is   1
Maximum version number of extension files of type AN is   1
Maximum version number of extension files of type WX is   1
Maximum version number of extension files of type FQ is   1
Maximum version number of extension files of type SU is   1
Keyword = 'OLDRFQ  '  value =  4.97149000D+09
```

Note that the sort order of the output data set is listed as ** rather than TB and that there are no attached

CL and NX tables. This happens when FITLD detects what might be a sub-array condition (two frequency IDs or two sources observed at the same time) on reading the data. In clear cases, the actual simultaneous frequency IDs and sources will be reported. In this case, FITLD detected the use of multiple integration times on different baselines in the data set; this is common for SVLBI data. The message reported by FITLD in this case takes the form:

```
*****************************************
FITLD5:  Subarray or multiple dump-rate condition found.
FITLD5:  NX/CL tables deleted.
FITLD5:  Use USUBA to set up subarrays.
FITLD5:  Rerun INDXR using CPARM(3) and (4)
FITLD5: *****************************************
```

Unless any of the following criteria are met, the data written by FITLD are immediately ready for further processing.

- If the sort code has been blanked as in this example, you must sort the data (use UVSRT or MSORT).
- If the source subarray condition is encountered, you may need to run USUBA *after* doing any ANTAB that may be needed..
- If the frequency ID subarray condition is encountered, you must separate the frequency IDs into separate data sets; procedure VLBAFQS will do this for you.
- If FITLD does not leave behind CL and NX tables, you must run INDXR to create them. Procedure VLBAFIX will do all of the above for you.
- If you wish to join together data processed in multiple correlator passes, you must run VBGLU and/or VBMRG.

FITLD can also be used to load archived *AIPS* data previously written to tape using either FITTP or FITAB, as described in § 5.1.2. In this case the VLBA correlator-specific adverbs, such as those enabling digital and delay corrections, are not active.

9.3.1.2 Calibration transfer

Beginning on 1 April 1999, the VLBA correlator attaches calibration information for VLBA and some non-VLBA antennas directly to the output FITS files. If your IMHEADER listing shows GC, TY, WX, FG, and PC tables, then the correlator has provided calibration information; this service is called *calibration transfer*. Note that projects correlated at slightly earlier dates may also have calibration transfer information. You must have 15APR99 or later version of *AIPS* to take advantage of calibration transfer. Not all antennas provide all the information needed for calibration transfer to the VLBA correlator, see

https://library.nrao.edu/public/memos/vlba/ops/VLBAO_34.pdf

for the latest information on this subject. For those antennas for which calibration information was not transferred by the VLBA correlator, you must process the log files in the traditional way as outlined in § 9.5.2. Calibration for the VLA and the GBT began to be transferred with the FITS files in November 2003.

Between April 1999 and late 2009 the information processed by the correlator was somewhat redundant so that the calibration tables, the GC table in particular, must be merged using TAMRG, a very general and hence complicated task. There are a couple procedures to do this for you in the VLBAUTIL package, VLBAFIX and VLBAMCAL, if you have previously run VLBAFIX your tables have been merged:

> RUN VLBAUTIL ↵ to acquire the procedures; this should be done only once since they will be remembered.

> INDISK n ; GETN ctn ↵ to specify the input file.
> INP VLBAFIX ↵ to review the inputs.
> VLBAFIX ↵ to run the procedure.

X
You should use VLBAFIX after you have finished loading the data from tape, but before you either change the polarization structure of the data with FXPOL, load any calibration data for non-VLBA telescopes, or apply the calibration data.

At this point it is a good idea to save the tables that were loaded with your data with TASAV. This protects you from having to reload the uv data from scratch if one of the original tables is damaged in some way. If you need to copy a table from the TASAV'ed file use TACOP.

9.3.1.3 Repairing VLBA data after FITLD

As listed above, there are a variety of reasons why VLBA data may need some repair after FITLD has been run. They may need to be sorted into strict time order, to have the subarray nomenclature corrected, to be split into different frequencies, to have the polarization structure fixed, and/or to have the original index (NX) table and calibration ((CL) recreated. These repairs can all be done by the procedure VLBAFIX, which will examine the data and perform any of the necessary fixes. If the data contain subarrays then the procedure must be told to split the data into multiple subarrays (SUBARRAY=2), otherwise it will assume no subarrays and force all the data into one subarray.VLBAFIX is intended to replace VLBASUBS, VLBAFQS, VLBAMCAL and VLBAFPOL, all of which can be run individually instead. Also we have recommended in the last few sections that VLBAFIX be run, if you have already run it, it does not need to be run again.

> RUN VLBAUTIL ↵ to acquire the procedures; this should be done only once since they will be remembered.
> INDISK n ; GETN ctn ↵ to specify the input file.
> CLINT $\triangle t$ ↵ to set the CL table interval to $\triangle t$ minutes (see discussion above in § 9.3.1.1).
> OUTDISK m ↵ to specify the output disk when needed.
> INP VLBAFIX ↵ to review the inputs.
> VLBAFIX ↵ to run the procedure.

Remember that all of the VLBAUTIL procedures have HELP files with good discussions about when to use the simple procedures and when to use the tasks directly.

9.3.1.4 Sorting and indexing VLBA correlator data

If multiple integration times are used on different baselines, the VLBA correlator will write data that are not in strict time-baseline (TB) sort order. VLBAFIX (§ 9.3.1.3) will sort your data if needed, if you want to do the sorting by hand do the following. In general, task UVSRT can be used to sort randomly ordered *uv* data files in *AIPS*, but has significant disk space requirements through the use of intermediate scratch files. A special task, MSORT, has been written which uses a direct memory sort with sufficiently large buffers to accommodate the scale over which the data deviate from true time-baseline sort order. No intermediate scratch files are used and it can be significantly faster than UVSRT for this special case. MSORT competes with UVSRT in performance even in other cases, particularly when the individual visibility records are large due to many spectral channels and/or IFs. The inputs to MSORT are similar to those required by UVSRT and take the form:

> TASK 'MSORT' ; INP ↵ to review the inputs.
> INDISK *n* ; GETN *ctn* ↵ to specify the input file.
> OUTDISK *n* ; OUTNAM ' ', OUTCLA ' ' to specify the output file.
> SORT ' ' ↵ to select default sort order ('TB' or time-baseline).
> GO ↵ to run the program.

Note that if the input and output file names are identical, the input file is sorted in place. In-place sorting is dangerous, but may be necessary if there is insufficient disk space for a second copy of the data set or for the intermediate scratch files required by UVSRT. *Never abort an in-place sort in progress because you will destroy the integrity of your data set.*

9.3.1.5 Subarraying VLBA correlator data

If the project was observed without using subarrays (defined as times at which separate antennas are simultaneously observing different sources or at different frequencies), *this step involving* USUBA *is not necessary and should be skipped*.

If the observations have been scheduled in separate subarrays, defined either by source or frequency selection, the subarrays should be labeled in *AIPS* before proceeding any further. The VLBA correlator does not conserve subarray information, which in any event often has no unique characterization. This is specified in *AIPS* using task USUBA which allows subarrays to be defined through either the input adverbs, an external KEYIN text file, or through the use of an automatic algorithm to identify and label subarrays found in the data. The automatic algorithm is recommended, *but its results should be checked closely*. Note that ANTAB tables do not know about subarrays that will be assigned by USUBA, so you must run all ANTABs before running USUBA.

If you have subarrays, they need to be sorted, have the subarray nomenclature corrected, and/or have the index (NX) table and calibration (CL) version 1 table rebuilt. In this case, there is a simplified procedure to combine the three repair operation, VLBASUBS. Only use this procedure if you know you have subarrays.

> RUN VLBAUTIL ↵ to acquire the procedures; this should be done only once since they will be remembered.
> INDISK *n* ; GETN *ctn* ↵ to specify the input file.
> CLINT $\triangle t$ ↵ to set the CL table interval to $\triangle t$ minutes.
> INP VLBASUBS ↵ to review the inputs.
> VLBASUBS ↵ to run the procedure.

The only user-controllable input is the CL table interval; see discussion above. VLBAFIX will perform this operation if requested (§ 9.3.1.3).

For automatic subarray labeling by USUBA, representative input parameters would be:

> TASK 'USUBA' ; INP ↵ to review the inputs.
> INDISK *n* ; GETN *ctn* ↵ to specify the input file.
> OPCODE 'AUTO' ↵ to identify subarrays automatically.
> TIMERANG 0 ↵ to include all times.
> ANTENNAS 0 ; SOURCES ' ' ↵ to include all antennas and sources.
> FREQID -1 ; SUBA 0 ↵ to include all frequency IDs and subarrays.
> INFILE ' ' ↵ to use no external file for subarray identifications.
> GO ↵ to run the program.

Sometimes FITLD erroneously identifies a subarray condition, usually because of spurious total-power data points. In such cases, you can set OPCODE = ' ' ; SUBARRAY = 1 to force all data into the first subarray.

In circumstances requiring USUBA, one often wants the calibration in one subarray to apply to other subarrays. USUBA will make the subarray column have value 0 which means all. Other tasks may not be so obliging so you may need to use TABED or verb TABPUT to change tables from subarray-specific to subarray-general.

9.3.1.6 Indexing VLBA correlator data

If FITLD had not written NX or CL tables or it was necessary to sort the data as described in § 9.3.1.4, you must run task INDXR. If VLBAFIX (§ 9.3.1.3) was run this is done automatically. INDXR will generate an NX table and, if need be, a CL table. Typical parameters for INDXR are:

> TASK 'INDXR' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> PRTLEV 0 ↵	to print minimal details of progress.
> CPARM 0, 0 , $\triangle t$, 1 ↵	to set the CL interval to $\triangle t$ and recalculate the model.
> GO ↵	to run the program.

Note that CPARM(4) can be set to zero unless the correlator model is required in later reduction (*e.g.*, in astrometry or geodesy observations) The CL table sampling interval $\triangle t$ should be chosen subject to the same considerations given regarding adverb CLINT in the discussion of FITLD in § 9.3.1.1. VLBAFIX will perform this operation if needed (§ 9.3.1.3).

9.3.1.7 Concatenating VLBA correlator data

Sometimes an observation is correlated using multiple passes through the VLBA correlator. In this context, multiple pass means different IFs/pass; this is due to data rate limitations in the correlator. Be careful to have FITLD load each pass into a separate disk file; otherwise a very confused data set will be produced. If it is desired to join together the IFs correlated on each pass, the task VBGLU should be used. VBGLU can only handle data sets that have the same polarization and spectral channel parameters. In 31DEC21, it can join data sets from different days while previously it required the oservation date to be the same. If the IFs are not observed at the same time, the file becomes rather larger due to the many flagged IFs. But, this option does allow rather different frequencies to be imaged together correctly. Task MATCH may be used to make the antenna, source, and frequency ID numbers in one data set the same as those in another data set so that they may be used as inputs to VBGLU. Note that MATCH is very fussy about details. You may use RESEQ to renumber antennas and RESOU to renumber sources in order to bring data sets into agreement.

The inputs to VBGLU are rather simple. Each of the input files to be glued together is specified via INNAME–IN4NAME, and an output file is specified via OUTNAME. The choice of input file 1 is no longer important. No data are lost in the revised version of this task.

> TASK 'VBGLU' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> IN2DISK *n* ; GET2N *ctn* ↵	to specify the $2^{\rm nd}$ input file.
> IN3DISK *n* ; GET3N *ctn* ↵	to specify the $3^{\rm rd}$ input file.
> IN4DISK *n* ; GET4N *ctn* ↵	to specify the $4^{\rm th}$ input file.
> OUTDISK *n* ↵	to specify the output disk.
> GO ↵	to run the program.

With the changes in recording technology, it is also possible that a correlator will not have enough playback units for all antennas in an experiment. In this case, multiple correlations will also have to be done in order to correlate every possible baseline. But, inevitably, certain baselines will appear more than once in these

correlations. FITLD will load all passes into a single data set (if DOCONCAT=1) or separate disk files which may be concatenated, after MATCH, with DBCON. Note that MATCH is very fussy about details. You may use RESEQ to renumber antennas and RESOU to renumber sources in order to bring data sets into agreement. Sort the data set into 'BT' order with UVSRT. Then task VBMRG may be used to discard any duplicate data. Task DBAPP may be used to avoid the 2^n proliferation of files, but only if the files are fairly similar in antennas, subarrays, and frequency IDs.

9.3.1.8 Labeling VLBA correlator polarization data

The VLBA correlator does not preserve polarization information unless it is operating in full polarization mode. This results in polarizations not being labeled correctly when both RR and LL polarizations are observed without RL and LR. Each VLBA correlator band is loaded into *AIPS* as a separate IF and is assigned the same polarization. FXPOL takes a data set from the VLBA correlator and produces a new data set that has the correct IF and polarization assignments. Unfortunately, there is no reliable way to determine the polarization of each IF from the input data set and you must specify the polarization assignments using the BANDPOL adverb.

Most VLBA setups assign odd-numbered bands to RCP and even-numbered bands to LCP. In this case BANDPOL should be set to '*(RL) ' (the default) and FXPOL will generate a new data set that is of equal size to the input data set, but has two polarizations and half the number of IFs. This case normally applies if LISTR shows pairs of IFs with the same frequency and QHEADER shows one pixel on the STOKES axis with coordinate value RR, but there may be exceptions to this rule when non-VLBA antennas are used.

There is a procedure for use with VLBA-only data that attempts to determine which of the above cases applies and then runs FXPOL for you, if you ran VLBAFIX (§ 9.3.1.3) this has already been done:

> RUN VLBAUTIL ⏎	to acquire the procedures; this should be done only once since they will be remembered.
> INDISK *n* ; GETN *ctn* ⏎	to specify the input file.
> INP VLBAFPOL ⏎	to review the inputs.
> VLBAFPOL ⏎	to run the procedure.

Use VLBAFPOL to check whether you need to relabel the polarizations in your data after loading the data, looking for subarrays, and merging redundant calibration data, but before reading any calibration data from non-VLBA stations. VLBAFPOL assumes that all of your FREQIDs have similar polarization setups. For this reason, you should normally run VLBAFPOL after copying each frequency ID to a separate file using VLBAFQS (§ 9.5). This strategy also reduces the amount of disk space needed for VLBAFPOL.

To use FXPOL directly, typical inputs are:

> TASK 'FXPOL' ; INP ⏎	to review the inputs.
> INDISK *n* ; GETN *ctn* ⏎	to specify the input file.
> BANDPOL '*(RL)' ⏎	to specify the normal VLBA polarization structure.
> GO ⏎	to run the program.

Consult HELP FXPOL for further information about more complicated cases. Note that FXPOL has to write a new output file since the structure of the data is being changed. All standard extension files are also converted, but it is still a good idea to run FXPOL before running the calibration tasks.

In single-polarization observations, LL data may simply be mis-labeled as RR or vice-versa. This does not need to be corrected within *AIPS* but the user needs to take this into account when selecting or calibrating the data, particularly in specifying the polarization in the amplitude calibration text file (§ 9.5.2). The Stokes axis can however be modified. Before running PUTHEAD, you should run IMHEAD to check which axis is the Stokes axis in the catalog header.

> INP PUTHEAD ↵ to review the inputs.
> INDISK *n* ; GETN *ctn* ↵ to specify the input file.
> KEYWORD 'CRVAL*m*' ↵ to select the Stokes or $m^{\rm th}$ axis in the header.
> KEYVALUE = -2 ↵ to set the Stokes value to 'LL' (or -1 for 'RR').
> PUTHEAD ↵ to set the coordinate value.

9.3.1.9 Ionospheric corrections

At low frequencies (2 GHz and lower) the ionosphere can cause large unmodeled dispersive delays, seen as rapid phase wrapping. This can be of particular importance in phase referencing observations, where phases must be interpolated over weak sources. Even at high frequencies (*e.g.*, 8 GHz) the ionosphere can be important, depending on the experiment and the condition of the atmosphere during the observation. One way to remove at least some of the ionospheric phase offsets is by applying a global ionospheric model derived from GPS measurements. The *AIPS* task TECOR processes such ionospheric models that are in standard format known as the IONEX format. These models are available from the Crustal Dynamics Data Information System (CDDIS) archive. There is a procedure which is part of VLBAUTIL, called VLBATECR that automatically downloads the needed IONEX files from CDDIS and runs TECOR. It will examine the header and the NX table and figure out which dates need to be downloaded, so the observation date in the header must be correct and an NX table must exist. See EXPLAIN VLBATECR for other requirements. TECOR also determines the ionospheric Faraday rotation which is applied in the calibration of the cross-hand visibilities.

> RUN VLBAUTIL ↵ to acquire the procedures; this should be done only once since they will be remembered.
> INDISK *n* ; GETN *ctn* ↵ to specify the input file.
> INP VLBATECR ↵ to review the inputs.
> VLBATECR ↵ to run the procedure.

You can also download the files manually from the CDDIS archive through anonymous ftp and run VLBATECR or TECOR, see EXPLAIN TECOR for detailed instructions on how to retrieve the models. TECOR interpolates between the maps of electron content in the ionosphere; therefore IONEX files must be retrieved to cover the entire experiment. Presently, each IONEX file contains maps every 2 hours from hours 00:00 to 24:00. Before November 2002, they contained maps every 2 hours from hours 1:00 through 23:00. Therefore, for example, if an experiment prior to November 2002 started at 0:00 then files must be retrieved for the day of the experiment and the previous day so the times between 0:00 and 0:59 can be interpolated. More recent experiments require two or more files only if they occurred in two or more days.

Typical inputs to TECOR are:

> TASK 'TECOR' ; INP ↵ to review the inputs.
> INDISK *n* ; GETN *ctn* ↵ to specify the input file.
> INFILE 'FITS:JPLG1230.01I' ↵ to set the name of the IONEX file. If there is more than one file, this name *must* be a standard format and be the first file. See EXPLAIN TECOR for more details.
> NFILES *n* ↵ to set number of IONEX files to be read.
> SUBARRAY 0 ↵ to process all subarrays. This option allows you to process subarrays used on different dates.
> ANTENNAS 1 2 3 4 6 7 8 9 10 ↵ to find corrections for all antennas except antenna 5 in a ten antenna experiment. This is important because if the IONEX models do not cover an antenna and it is not excluded here then all the solutions for that antenna will be undefined and the data flagged when the CL table is applied.

> GAINVER 1 ↵	to apply corrections to the first CL table.
> GAINUSE 2 ↵	to create CL table 2 with the corrections.
> APARM 1 0 ↵	to correct for dispersive delay; otherwise only the ionospheric Faraday rotation will be corrected.
> GO ↵	to run TECOR, correcting for the ionosphere in a new CL table.

The dispersive delays should be checked using EDITA (option DODELAY=3), SNPLT (options INEXT 'CL'; INVERS 2; OPTY 'DDLY') and VPLOT (options BPARM 0; APARM 0; DOCAL 1; GAINUSE 2). TEPLT plots parameters from the TE table written by TECOR, in particular the Faraday rotation, total electron content, and dispersive delay. It can also plot the difference of two TE tables in multiple parameters. MFIMG can be used to make an image cube of the IONEX data after which TVMOVIE and other image display tools will let you examine the data given to TECOR. TECOR is only as good as the models, which at this time are quite rough. Therefore, it is a very good idea to compare the corrected and uncorrected phases using VPLOT.

Extensive studies have revealed that TECOR usually over-corrects data. It removes the time-dependent IFRM, but leaves behind an extra rotation measure that changes some with obsering date and very slightly with source. A package of software called ALBUS has been found to give somewhat better solutions for the IFRM. In 31DEC24 a task also named ALBUS was written to invoke this package to correct uv data including dispersive delay which is important to VLBI data. Unfortunately, the complications of this package require the Linux operating system and an outside program called apptainer. See EVLA Memo 235 for details.[1]

CLCOR has a OPCODE = 'IONO' to make delay corrections for the ionosphere, similar to the 'ATMO' operation which is for the atmosphere (§ 9.5.7.5).

9.3.1.10 Corrections to the Earth Orientation Parameters

This correction is only useful for experiments correlated at the VLBA correlator. VLBI correlators must use measurements of the Earth Orientation Parameters (EOPs) to take them out of the observations. These change slowly with time and therefore the EOPs used by the correlator must be continually updated. From 5-May-2003 to 9-Aug-2005 the VLBA correlator used old predicted EOPs which could be significantly wrong and will effect all phase referencing experiments. Incorrect EOPs can both move the position and possibly smear the target of a phase referencing experiment. Self-calibration can improve the smearing. Even outside the above quoted period of particularly bad EOPs, the EOPs can be off so it is recommended that all phase-referencing experiments, particularly astrometry experiments should have their EOPs corrected. CLCOR (OPCODE='EOPS') can do this correction. It uses the CT table which is only produced by the VLBA correlator, so at the moment CLCOR can only correct experiments processed at the VLBA correlator. CLCOR also uses a file of measured EOPs, which can be downloaded from NASA (see EXPLAIN CLCOR for details). There is a procedure which is part of VLBAUTIL, called VLBAEOPS, which downloads the file automatically and runs CLCOR. To run the procedure:

> RUN VLBAUTIL ↵	to acquire the procedures; this should be done only once since they will be remembered.
> INDISK n ; GETN ctn ↵	to specify the input file.
> INFILE ' ' ↵	to automatically download file.
> INP VLBAEOPS ↵	to review the inputs.
> VLBAEOPS ↵	to run the procedure.

The procedure will correct the highest CL version while copying it to a version one higher.

If multiple days have been concatenated, you should run CLCOR manually. Download the file using the instructions in CLCOR's explain file. Sample inputs are as follows:

[1] "Testing Ionospheric Faraday Rotation Measure Models", Perley R., Greisen, E., Tremou, L., Butler, B., Willis, T., October 2024

> TASK 'CLCOR' ; INP ↵	to review the inputs.
> INDISK *n1* ; GETN *ctn* ↵	to specify the input file.
> OPCODE 'EOPS' ↵	to select correction of the EOPs.
> GAINVER *clin* ↵	CL table to read, new default is the current highest version.
> GAINUSE 0 ↵	CL table to write; version highest+1 is written unless GAINUSE = GAINVER.
> INFILE 'FITS:usno_finals.erp ↵	to specify file with correct EOPs — note missing close quote to retain lower case letters.
> CLCORPRM(2) *dd* ↵	to use *dd* days worth of data following the start date from the USNO file. Set dd to about 4 days more than your last day of data.
> GO ↵	to run the program.

In case the automatic download fails, VLBAEOPS can use the file which you have downloaded manually.

9.3.1.11 Preparing the OB table for SVLBI data

The spacecraft orbit table (OB) as produced by FITLD contains the spacecraft position (x, y, z) and velocity (v_x, v_y, v_z) as calculated to high accuracy from the JPL reconstructed orbit using the SPICE package (developed at JPL). These quantities are calculated by the correlator on-line software and are passed directly through to *AIPS* via FITLD by the VLBA correlator. The orbit table is indexed on time and can include information such as the angle between the spacecraft pointing direction and the Sun, the time since the start and end of the last eclipse, and the spacecraft parallactic angle. The latter quantities are not available to the correlator on-line software and, if desired, need to be computed separately for later use in *AIPS* by task OBTAB. Additionally OBTAB stores orbital elements in the AN table; these are essential for later use in plotting or inspecting spacecraft orbit information. Sample inputs for OBTAB, are as follows:

> TASK 'OBTAB' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> INVERS 1 ↵	to process OB table 1.
> SUBARRAY 1 ↵	to select subarray number.
> APARM 1, 0 ↵	to update orbital elements in AN table
> GO ↵	to run the program.

Note that APARM(8) can be used to directly specify which antenna is the orbiting antenna (this task assumes there is only one orbiting antenna).

Task OBTAB determines mean orbital elements from the OB table using the spacecraft positions and velocities and updates the AN table under control of APARM(1). The orbital elements can be examined by using PRTAB to review the updated AN table. The mean elements are used to compute *uv* coordinates for the spacecraft so that model amplitudes AND closure phases and amplitudes can be plotted (by tasks VPLOT, CLPLT, and CAPLT). The orbit table can be plotted using task OBPLT. Alternatively, the task TAPLT can be used to display individual columns. Use PRTAB to determine the names of the columns you wish plotted.

9.3.1.12 Loading the time corrections file for SVLBI data

The round-trip residual delay measurements determined by the tracking stations are supplied to the correlator in FITS format. These so-called delta-T tables are not passed to *AIPS* by the correlator but can be loaded indirectly. This table might be used, for example, to plot the time correction as a function of time. Such information could be useful if a user suspects a loss of coherence due to a poor predicted orbit

or a clock jump at the tracking station. The table contains no internal time stamps, so the row number must be used to determine the approximate time of a given entry (there are typically 10 rows per second).

The delta-T tables can be loaded at present using FITLD and attached to a null *uv* data file, using input parameters as follows:

> TASK 'FITLD' ; INP ↵ to review the inputs.
> OUTDISK *n* ↵ to specify a separate output file.
> OUTNAM 'DUMMY', OUTCLA 'DT'
> DATAIN 'FITS:3551708.kct.a ↵ to specify the external FITS file.
> GO ↵ to run the program.

The other FITLD adverbs are not relevant in this instance. Note that lower case letters can be used in the DATAIN adverb if the trailing quotation mark is omitted. FITLD will load the external FITS file successfully but will print an error message complaining that no array geometry table was found. This message can be ignored in this case. The delta-T table will appear as an unknown table of type UK, and can be plotted using task TAPLT

9.4 Tools for data examination

Before proceeding further it is important to examine the data, to make sure they are all loaded, and (especially if this is the first time you have reduced VLBI data) to familiarize yourself with the data structure. As processing continues it is also important to inspect the data periodically to check on the progress of the calibration. Use the verb IMHEAD regularly to check the *uv*-data header, particularly the list of tables (as seen in § 9.3.1.1).

Some tasks that can be used to examine the data and the associated tables are LISTR, DTSUM, POSSM, VPLOT, CLPLT, CAPLT, EDITR, TVFLG, SPFLG, SNEDT, SNPLT, PRTAB, FRPLT, PRTAN, COHER, OBPLT, and SHOUV. Some of these tasks are described in the next few pages.

9.4.1 Textual displays

As a first step, use the procedure VLBASUMM to print out the essential contents of your data set:

> RUN VLBAUTIL ↵ to acquire the procedures; this should be done only once since they will be remembered.
> INDISK *n* ; GETN *ctn* ↵ to specify the input file.
> DOCRT -1 ↵ to direct the output to the line printer.
> INP VLBASUMM ↵ to review the inputs.
> VLBASUMM ↵ to run the procedure.

This will make a listing of the scans, sources, frequency structure, and antennas found in your data set. You should run this procedure after "fixing" the data with VLBAMCAL, VLBAFQS, VLBASUBS, and VLBAFPOL, but you may also find it useful on the initial dataset.

VLBASUMM runs the task LISTR to give a listing of the scans, with source names, time ranges, frequency ID's and total number of visibilities per scan for each of your output files. It is often useful to print out a paper copy of this to facilitate later data plotting/editing. If you did not do VLBASUMM, use:

> TASK 'LISTR' ; INP ↵ to review the inputs.
> INDISK *n* ; GETN *ctn* ↵ to specify the input file.
> OPTYP 'SCAN' ↵ to request printing of scan summaries.

> DOCRT -1 ↵ to direct the output to a printer.
> OUTPRINT ''; ↵ to have the output printed immediately.
> GO ↵ to run the program.

Note that at the end of the above LISTR output is useful information about the frequency structure of your data set. In 31DEC23, LISUN produces a similar listing plus the angle of each scan to the Sun with fewer adverbs to consider. LISUN also prints a summary of the antennas used. LISTR with OPTYP = 'LIST' and DPARM(1) = 1 is also a good way to look for phase coherence. If LISTR fails at this point, you may have forgotten to run INDXR and/or MSORT (see § 9.3.1.6 and § 9.3.1.4).

Other verbs/tasks for inspecting your data include;

1. IMHEAD lists the file header including information on the number of frequency channels and Stokes parameters in the data and gives a list of all the extension tables.

2. PRTAB can be used to print the contents of any of these extension tables, for instance the SU or Source table contains information about each of the sources observed. Some tables have very many columns. You can use input parameter BOX to select the list of the columns you want to print.

3. PRTAN, run by VLBASUMM, provides a listing of the antenna names and their associated antenna numbers. This is useful because antennas are generally specified by their antenna numbers. (Procedure ANTNUM in the VLBAUTIL set allows you to translate station names into numbers.)

4. DTSUM produces a matrix showing the number of visibilities on each baseline for each scan allowing a check to be made that all baselines have been loaded. It also tells you the data integration time. DTSUM also has a mode (triggered by setting APARM(1) = 1) that will produce a useful matrix summary of your whole data set. DTSUM does not properly report the integration times when there are multiple integration times in the data set. Verb GETITIME returns the integration time as INTTIME but has the same problem with multiple integration times.

9.4.2 Spectral displays: POSSM

Your data file will probably contain a number of IFs, observed at different frequencies, corresponding to the separate "IF channels" used during the observations. Use IMHEAD to find the number of IF channels and the number of spectral channels per IF channel, or examine the LISTR or LISUN output. The reason that the data must be stored in narrow spectral channels, even for continuum applications, is that, in VLBI, the geometrical and propagation errors affecting the data can be large enough to cause significant phase changes across an IF channel bandwidth, preventing a coherent integration over the full bandwidth.

The frequency structure of the data can be inspected using POSSM, which provides a plot of visibility data as a function of frequency as integrated over a specified time interval. Optionally, data from up to nine baselines can be plotted on a single plot page. Initially it may be interesting to view the frequency structure of data on a bright calibrator source, as in the example below. Because, prior to calibration, the phases in each IF channel are likely to vary rapidly with time, it is important to average data coherently only over a short time interval. In general, you will see phase slopes and offsets affecting the data; these phase errors must be determined and removed before the data can be averaged in frequency and/or time. See § 9.5.7 for more information and a sample plot. There is a procedure simplifying the use of POSSM:

> RUN VLBAUTIL ↵ to acquire the procedures; this should be done only once since they will be remembered.
> INDISK *n*; GETN *ctn* ↵ to specify the input file.
> SOURCES '' ↵ to plot all sources.
> TIMERANG 0 ↵ to plot all times.

> SUBARRAY 0 ↵	to plot all subarrays.
> REFANT *n* ↵	to plot the cross-power spectrum for baselines with antenna *n*.
> STOKES 'I' ↵	to plot Stokes I.
> GAINUSE *CLin* ↵	to apply CL table *CLin* to the data before plotting.
> DOTV 1 ↵	to plot the data on the TV; -1 to make a plot file.
> VLBACRPL ↵	to plot the data.

To use POSSM directly to display the visibility spectrum of a source on the TV, use:

> TASK 'POSSM' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> SOURCES 'OQ208', ' ' ↵	to specify a single source name.
> TIMER 1 2 15 0 1 2 15 30 ↵	to define a time range.
> ANTENNAS 8 ; BASELINE 0 ↵	to plot all baselines to antenna 8.
> DOCAL -1 ↵	to plot the data without calibration.
> BIF 0 ; EIF 0 ; BCHAN 0 ; ECHAN 0 ↵	to include all IFs and spectral channels.
> STOKES 'HALF' ↵	To plot RR and LL separately.
> CODETYPE ' ' ; POLPLOT ' ' ↵	to plot amplitude and phase.
> SOLINT 0 ↵	to average over the full time range.
> APARM 1 , 1 , 0 , 0 , -180 , 180 , 0 , 0 , 3 , 0 ↵	to control the plot: APARM(1)=1 to use vector averaging, APARM(2)=1 to use fixed scale plots, APARM(5) and APARM(6) to set phase range, APARM(9)=3 to plot all IFs and polarizations together in one diagram.
> NPLOTS 9 ; BPARM 0 ; OUTTEXT ' ' ↵	to have 9 plots per page without division by "channel 0" and without writing the spectrum to a file.
> DOTV 1 ↵	to plot on the TV, else create plot extension.
> BADDISK 0 ↵	to use all disks for scratch.
> GO ↵	to run the program.

Note that the amplitudes are totally uncalibrated at this stage and are in units of "correlation coefficients"; these will generally appear on plots mislabeled as mJy (representing multiples of 10^{-3} in correlation coefficient). POSSM can produce text output into the file given by OUTTEXT.

Sample POSSM displays are given in Figure 9.1, Figure 9.2, and Figure 10.2.

Task SHOUV with OPTYPE 'SPEC' will display the data from a number of channels on the printer with optional time averaging.

9.4.3 Time displays: VPLOT, CLPLT, and CAPLT

The task VPLOT can be used to view the visibility data as a function of time (or other variables). Again, data from several baselines can appear on one plot page. Plots of amplitudes and phases and several other quantities can be made, although, to view closure phase and amplitude, you must use tasks CLPLT and CAPLT. Note that VPLOT can average spectral channels or plot them individually under control of AVGCHAN and can plot spectral channels and IFs in separate panels or all together under control of CROWDED. Data points are plotted with a user-selected SYMBOL and may be connected by lines under control of FACTOR. Calibration can be applied to the data before they are plotted. The data can be averaged in time and the

max/min within the interval plotted along with the average. Also, if desired, a model can be plotted against the data. The model can either be displayed at the times of the data samples or, with somewhat less accuracy, continuously, even at times for which there are no associated data or recorded *uv* coordinate values.

The following parameters will display uncalibrated amplitudes and phases from a single spectral channel of a single IF channel for a short scan on a bright calibrator:

> TASK 'VPLOT' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> CLR2NAME ↵	to ensure no model is plotted.
> SOURCES 'OQ208' ' ↵	to specify the source name.
> BIF 4 ↵	to give first included IF channel.
> EIF 4 ↵	to give last included IF channel;, if EIF > BIF then IFs are averaged.
> BCHAN 8 ↵	to set the lowest spectral channel to include in average prior to plotting.
> ECHAN 8 ↵	to set the highest spectral channel to include in average prior to plotting; no averaging in this case.
> TIMER 1 2 15 0 1 2 25 00 ↵	to define a 10-minute time range.
> DOCAL -1 ; DOBAND -1 ↵	to apply no calibration tables or bandpass tables to data.
> OPTYP ' '	to display cross-correlations; 'AUTO' to get auto-correlations.
> SOLINT 0	to do no time averaging of the data before plotting.
> XINC 1 ↵	to plot every record.
> BPARM 0 , -1 ↵	to set x-axis type (BPARM(1) = 0 plots time (Hrs, min, sec)), y-axis type (BPARM(2)= -1 plots both amplitude and phase), and to use self-scaling. See EXPLAIN VPLOT ↵ for other options.
> NPLOT 4 ↵	to plot 4 baselines/page.
> GO ↵	to run the program.

An example VPLOT output appears as Figure 9.1. It may be useful to make a plot of data "weight" versus time on the autocorrelation data from each antenna (set BPARM(2)=16 and OPTYP = 'AUTO'). The weight depends on the number of valid bits correlated and is a good indication of tape playback quality.

9.4.4 EDITR

Another task which can display data as a function of time is EDITR. This program is used primarily to edit data interactively (see § 5.5.2), but its interactive aspects (*i.e.*, allowing the user to "zoom" in on certain time periods) make it useful for pure data inspection. EDITR now offers options reminiscent of difmap, the VLBI data reduction package from CalTech. In particular, the CROWDED adverb allows displays of all IFs and/or all polarizations at the same time (and color may be used to separate them visually) and the FLAG QUICKLY run-time option allows fast sample deletion with only quick mouse clicks. The new CLIP EVERYTHING option is paricularly efficient in deleting high values. The new coherence parameter (scalar average divided by vector average minus one) offers interesting possibilities as well. It is well worth exploring the abilities of this powerful program. Tasks TVFLG (§ O.1.6), SPFLG (§ 10.2.2), and WIPER are also useful in this way.

9.4.5 SNPLT

Supplied with your VLBI data will be a number of important tables used for calibration, and many more are generated as calibration proceeds. §9.8 summarizes the contents of each of these tables. Two of the most important tables are the calibration or CL tables and the solution or SN tables.

The task SNPLT should be used periodically to inspect the contents of the latest CL and SN tables. There is a simplified procedure for making these plots:

> RUN VLBAUTIL C$_R$	to acquire the procedures; this should be done only once since they will be remembered.
> INDISK *n* ; GETN *ctn* C$_R$	to specify the input file.
> INEXT 'CL' C$_R$	to plot a CL table.
> INVERS 0 C$_R$	to plot the highest version.
> SOURCES ' ' C$_R$	to plot all sources.
> TIMERANG 0 C$_R$	to plot all times.
> STOKES ' ' C$_R$	to plot both R and L solutions.
> SUBARRAY 0 C$_R$	to plot all subarrays.
> OPTYPE 'AMP' C$_R$	to look at amplitudes; 'PHAS', 'DELA', and 'RATE' are other useful choices.
> DOTV 1 C$_R$	to plot the data on the TV; -1 to make a plot file.
> VLBASNPL C$_R$	to plot the data.

Using SNPLT directly, the example below plots the antenna-based amplitude corrections stored in CL table *m*.

> TASK 'SNPLT' ; INP C$_R$	to review the inputs.
> INDISK *n* ; GETN *ctn* C$_R$	to specify the input file.
> OPTYPE 'AMP' C$_R$	to plot amplitudes.
> OPCODE 'ALST' C$_R$	to plot both polarizations in each plot; selected IFs are plotted separately.
> INEXT 'CL' ; INVER *m* C$_R$	to choose CL table version.
> DOBLANK 1 ; DOTV 1 C$_R$	to plot on the TV screen including failed solutions; otherwise, create plot extension files.
> NPLOTS 5 C$_R$	to plot 5 antennas/IFs per page.
> DO3COL 2 C$_R$	to separate the calibration sources by color.
> GO C$_R$	to run the program.

SNPLT can also be used to plot quantities from other tables generated by the calibration process including the contents of TY ("system temperature"), and PC ("phase-cal") tables. OPTYPE = 'MULT' allows you to compare values of two or more parameters, plotting them at the same time in separate panels. Task SNIFS is similar and may be used to compare values across IFs. SNBPL is similar to SNPLT, but plots calibartion parameters on a baseline basis. Unstable phases may be stable on some baselines, but not others. The antenna-based plots of SNPLT may not differentiate these adequately.

9.4.6 COHER

The task COHER can be used to determine the coherence time in a *uv* data set broken down both in time and by antenna and baseline. The coherence time is estimated by comparing vector and scalar averaged amplitudes over increasing time averaging intervals. Averaging is not performed over source

scan boundaries. The coherence time is defined as the averaging interval over which the ratio of vector and scalar amplitudes falls below a pre-assigned level. This can be set under user control. In addition, data that fall below a specified signal-to-noise ratio can be excluded from the coherence time estimates. Provision is made for selection by source name, time range, IF channel, antenna and frequency ID, and the ability to average over a subset of individual frequency channels. The input parameters to task COHER take the form:

> TASK 'COHER' ; INP C_R	to review the inputs.
> INDISK *n* ; GETN *ctn* C_R	to specify the *uv* input file.
> APARM 0 C_R	SNR cutoff 5; vector to scalar cutoff 0.8.
> TIMERANG 0, 10, 5, 0, 0, 10, 15, 0 C_R	time range selection.
> BIF 5 ; BCHAN 1 ; ECHAN 8 C_R	IF and channel selection.
> SOURCES 'DA193' , ' ' C_R	source selection.
> FREQID 1 C_R	Frequency ID selection.
> GO C_R	to run the program.

Be warned that COHER can take quite a long time to run. A simpler, though less rigorously correct method for determining coherence intervals is to examine the data on different baselines using EDITR.

9.4.7 FRPLT

A preliminary examination of the coherence of individual scans or time segments in the data can also be performed using task FRPLT, which allows time series or fringe-rate spectra to be plotted for one or more baselines and time intervals. This task allows data to be selected by the usual criteria, including time range, source name, and frequency parameters, amongst others, and will then plot the individual time series in amplitude and phase or the associated fringe-rate spectrum. Provision is made for averaging over frequency channels within each IF, for varying degrees of padding in the FFT, and for division by the pseudo-continuum average "channel zero" before plotting. Typical input parameters to FRPLT are:

> TASK 'FRPLT' ; INP C_R	to review the inputs.
> INDISK *n* ; GETN *ctn* C_R	to specify the input file.
> SRCNAME 'DA193' C_R	to select a source.
> TIMERANG 0, 10, 5, 0, 0, 10, 15, 0 C_R	to select a time range, strongly recommended.
> SOLINT 2 C_R	to plot fringe-rate spectra for every 2-minute interval in TIMERANG.
> NPLOTS 6 C_R	to do 6 plots per page.
> STOKES 'LL' C_R	to select a single Stokes.
> BIF 1 ; EIF 0 C_R	to select the IFs included.
> BCHAN 3 ; ECHAN 12 C_R	to select the range of frequency channels averaged within each IF.
> ANTENNAS 3, 4 C_R	to select baseline 3-4, 3-5, and 4-5.
> BASELINE 4, 5 C_R	to select baseline(s) plotted in the familiar way.
> DOCAL -1 ; DOPOL -1 C_R	to apply no continuum calibration.
> DOBAND -1 C_R	to do no bandpass calibration.
> APARM(1) 1 ; APARM(2) 0 C_R	integration time 1 sec; self-scale the plots.
> APARM(7) 0 C_R	plot fringe-rate spectra with no padding.
> BPARM 0 C_R	to do no division by "channel zero."
> DOTV 1 C_R	to plot directly on the TV device.
> GO C_R	to run the program.

The underlying time series in amplitude and phase can be plotted by setting APARM(7)=1; otherwise the fringe-rate spectrum is plotted. Note that the baseline(s) shown are selected with ANTENNAS and BASELINE

in the usual way and a time range must be selected with TIMERANG. APARM(1) sets the integration time to be used before doing the FFT over the selected time range. SOLINT may be used to break the time range into intervals. Separate plots are produced for each IF, baseline, and time interval.

9.5 Calibration strategy

If you have multiple frequency IDs in your data, you may want to separate the data for different FREQID before performing any calibration. Use UVCOP to do this and take advantage of the opportunity to delete data flagged by the correlator with FLAGVER=1. You no longer need to re-run INDXR on the output files. Again there is a procedure to do this for you:

> RUN VLBAUTIL ↵	to acquire the procedures; this should be done only once since they will be remembered.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> CLINT $\triangle t$ ↵	to set the CL table interval to $\triangle t$ minutes (see discussion above in § 9.3.1.1).
> INP VLBAFIX ↵	to review the inputs.
> VLBAFIX ↵	to run the procedure.

VLBAFIX will normally be run after loading the data.

We can now begin the process of calibrating VLBI data. As the calibration process proceeds, both amplitude and phase corrections are incorporated into the CL tables. VLBI correlator output is in terms of dimensionless "correlation coefficients." To convert to Janskys, large amplitude correction factors have to be entered into the CL tables. In addition, phase correction factors must be entered into the CL table to correct for phase offsets and ramps as functions of frequency and time. These corrections must be made so that the data can be averaged over frequency and time without loss of coherence. The process of determining the phase corrections is known as "*fringe-fitting*" in VLBI; see § 9.5.7.

In order to calibrate the absolute phase of their data, VLBI users use either "phase-referencing" or "self-calibration". If the self-calibration route is chosen, then the phase derivatives with respect to time and frequency are calibrated and the absolute phases are normally left uncalibrated. Then self-calibration methods are then used to generate images (see § 9.7). For phase-referencing, an absolute-phase calibration is done using an external calibrator. Phase-referencing in VLBI is similar to, but slightly more complicated than, phase calibration on the VLA. In general terms, phase-referencing VLBI data is accomplished by similar methods as used for VLA data in *AIPS*; be sure to read § 9.5.1.2 and § 9.5.7.4 for details on how to calibrate phase-referenced observations.

For astrometric data reduction methods in *AIPS* the reader is referred to the guide to *AIPS* astrometric data reduction available from within *AIPS* by typing HELP ASTROMET.

Optimum fringe-fitting results are obtained if amplitudes are calibrated first, since, in this case, the data will be weighted appropriately (the *AIPS* task FIXWT may be used to adjust the weights in the data to reflect the scatter of the actual data). We therefore describe the process of amplitude calibration first in § 9.5.4.6. Then, in § 9.5.7, we describe the calibration of residual phase using "fringe-fitting" techniques. Note however, that for observations of strong sources or observations using only VLBA antennas (where, particularly at centimeter wavelengths, the sensitivities on all baselines are roughly the same), the order of the two calibration steps may be reversed.

9.5.1 Incremental calibration philosophy

The general strategy adopted by *AIPS* for calibration is, starting with the lowest version of the CL table, to incorporate step-by-step amplitude and phase corrections for a number of different effects. At each stage either an existing CL table is modified or a new version is created from a lower version by a task which applies a certain type of calibration. Note that the actual visibilities are not changed until you are satisfied that you have the best possible calibration file; at this point the task SPLIT can be used to apply the calibration information of the best CL table to the data. However at each point along the way the effect of a particular CL table on the data can be viewed using POSSM or VPLOT by setting DOCAL=1 and GAINUSE equal to the chosen CL table. Since many CL tables may be produced in the course of calibrating a VLBI data set it is important to keep a note of which effects are included in each one. Ideally one should delete CL tables which are judged incorrect and ensure that the accumulated corrections lie in the highest numbered CL table. It is suggested that version 1 of the CL table, as produced by FITLD, be copied to CL version 2 using TACOP before any calibration is begun, and that CL version 2 be used as the starting point in the calibration sequence. An effort is made within *AIPS* to insure that CL version 1 is not deleted inadvertently. If this does occur however it can be re-generated using task INDXR (described in § 9.3.1.6). (Note that INDXR cannot re-generate some types of information, *e.g.*, the phase-cals inserted by the MKIII correlator so that it is important to try to preserve the first CL table.) The task TASAV can be used to back up your tables by copying all of them to a dummy file containing no data. This can be used to save a "snapshot" of the tables at various points in your data processing for insurance purposes. The tables can be copied back into your data file if necessary using TACOP.

You can also use the verb HINOTE to add comments into the history file. This can be very useful for check-pointing progress during calibration.

When you are ready to apply the calibrations, you run either SPLIT or SPLAT. Both tasks can average data in the spectral domain if appropriate. SPLAT can also time-average the data and produces multi-source data sets on output if requested.

9.5.1.1 Smoothing and applying corrections in SN and CL tables

The various stages of calibration described below produce SN tables which are then used to create CL tables using CLCAL. The ancillary tasks SNCOR, SNSMO, SNEDT, and CLCOR can be used to modify the SN and CL tables directly. It is important to choose the proper methods of interpolation in these tasks.

SN tables can be smoothed using tasks SNEDT and SNSMO before being used to update CL tables using CLCAL. SNSMO uses superior smoothing methods to those available in CLCAL and should always be used to do any smoothing of VLBI data, *i.e.*, data with non-zero delays and rates. The adverb DOBLANK now controls which data are actually altered by the smoothing; use it carefully.

Typical inputs for SNSMO would be:

> TASK 'SNSMO ; INP ⏎	to review the inputs.
> INDISK *n* ; GETN *ctn* ⏎	to specify the input file.
> SOURCES ' ' ⏎	to modify the solutions for all sources.
> SELBAND -1; SELFREQ -1; FREQID 0 ⏎	to do all frequency IDs.
> BIF 0; EIF 0 ⏎	to include all IFs.
> TIMERANG 0; ANTENNAS 0 ⏎	to include all times and antennas.
> SUBARRAY 0 ⏎	to select the first subarray; NB, SNSMO works only on one subarray at a time.
> SAMPTYPE 'MWF' ⏎	to use the median window filter method.

> SMOTYPE 'AMPL' ↵	to smooth amplitudes only.
> BPARM 0.5,0 ↵	to use a 30-minute filter time for amplitude.
> DOBLANK 1 ↵	to replace blanked values with interpolated values *only*.
> CPARM 0.5, 0, 0, 0, 0, 0.02, 0 ↵	to set ranges of allowed values.
> INVER *snin* ; OUTVER *snout* ↵	to read in the SN table version *snin* and to write SN table version *snout* (which should be a new table).
> REFANT 0 ; BADDISK 0 ↵	to keep the current reference antenna and to allow all disks to be used for scratch files.
> INP ↵	to check the inputs.
> GO ↵	to run the task.

Typical inputs for CLCAL would be:

> TASK 'CLCAL' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> CALSOUR ' ' ↵	to use all corrections in the SN table.
> SOURCE ' ' ↵	to apply corrections to all sources.
> OPCODE 'CALP' ↵	to apply SN tables to a CL table.
> INTERPOL ' ' ↵	to use linear vector interpolation ('2PT').
> SAMPTYPE ' ' ↵	to do no further smoothing of the merged SN tables..
> SNVER *snin* ; INVERS 0 ↵	to select the one SN table containing solutions to be interpolated.
> GAINVER *clin* ↵	to select the CL version to which solutions are to be applied.
> GAINUSE *clout* ↵	to select the output CL version, containing updated calibration information.
> REFANT 0 ; BADDISK 0 ↵	to try to use use a single reference antenna if possible during all steps of calibration.
> INP ↵	to check the inputs.
> GO ↵	to run the task.

Note well that SNVER=0 means here to combine the solutions from all SN tables, GAINVER=0 means to apply the solutions to the highest numbered CL table and GAINUSE=0 means to write a new CL table. If two SN tables contain two similar attempts at finding corrections and SNVER=0 then, effectively, CLCAL will be inconsistent in applying the solutions from these two tables. Basically, CLCAL simply concatenates all SN tables and then merges apparently identical records (same time and antenna) to eliminate blanked solutions and to complain about otherwise non-identical solutions.

The parameters INTERPOL and SAMPTYPE allow the user to choose between several different methods of smoothing the SN files followed by interpolation to the times in the CL table. Use EXPLAIN CLCAL ↵ to view all the options. The default interpolation option is INTERPOL = '2PT', in which the SN table is linearly interpolated between the measurements in the SN table. Using SAMPTYPE = 'BOX' causes the SN table to be smoothed with a boxcar function before being interpolated onto the CL table. The smoothing times for delay, rate etc. are specified in parameter BPARM. DOBLANK controls how both failed and good solutions are handled when smoothing. DOBLANK ≥ 0 replaces failed solutions with smoothed ones, while DOBLANK ≤ 0 replaces good solutions with smoothed ones. However it is recommended that SN smoothing be done prior to CLCAL using task SNSMO. When smoothing delays, use the new SMOTYPE = 'VLDE' method whenever the IFs have been grouped to find single solutions (APARM(5)> 0 in FRING) across multiple IFs. Be sure to set NPIECE to indicate how many groups of IFs were used in FRING, *e.g.*, 1 for a multi-band delay, 2 for a delay for IFs 1 — $Nif/2$ amd a delay for IFs $Nif/2 + 1$ — Nif.

With good quality data, the INTERPOL = 'AMBG' option should work well. Note, however, that this option uses the SN solutions immediately before and after a CL entry to make the interpolation and it uses any SN

solution found for any source specified in CALSOUR. Therefore, if CALSOUR is left blank (allowing all sources) and delay and rate solutions were significantly different for different sources, then inappropriate solutions may be applied for a few minutes before or after a source change.

One way of avoiding this problem is to run CLCAL with INTERPOL = 'AMBG' several times, once for each source, setting both SOURCES and CALSOUR to the name of the desired source with all other inputs remaining unchanged. Another way of avoiding the problem is to use INTERPOL = 'SELF'. In this option, only solutions found on a given source are used to calibrate that source and the SN table entries closest in time for that source are used with interpolation. This is not as good as doing multiple runs with the INTERPOL = 'AMBG' option because there can be jumps in phase at points equidistant from two SN table entries.

If there are bad SN solutions, INTERPOL = 'POLY' is used to fit a polynomial to the rate solutions and then integrate this polynomial to determine the phase corrections to be entered into the CL table.

A final note on CLCAL. It is sometimes the case that *a priori* information is not available for all antennas in a single format. For example, you may have system temperature information for VLBA antennas in your SN version 2 table and for non-VLBA antennas in SN version 3. You can merge this information by running CLCAL twice with the same GAINVER and GAINUSE; each time you should explicitly set the SN version number and list the antennas to be processed using the ANTENNAS adverb. If you leave the ANTENNAS adverb blank, the final CL table will contain information only for antennas present in the last SN table processed. Note that you must use OPCODE = 'CALI' when building up a CL table in pieces.

9.5.1.2 Running CLCAL for phase referencing observations

In particular, this example illustrates how to set the inputs for CLCAL for the specific case when phase corrections determined for the cal source 'J1636-16' are to be transferred to the target source 'P1643-12' in a phase referencing experiment:

> TASK 'CLCAL' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> CALSOUR 'J1636-16', ' ' ↵	to use the corrections determined for the cal source.
> SOURCES 'J1636-16', 'P1643-12', ' ' ↵	to apply corrections to both the cal source and the target source.
> OPCODE 'CALP' ↵	to apply SN tables to a CL table passing all data.
> INTERPOL 'AMBG' ↵	to use linear vector interpolation with no SN table smoothing and simple phase ambiguity removal. See above for more discussion of INTERPOL.
> SAMPTYPE ' '; BPARM 0 ↵	to clear smoothing parameters.
> SNVER *snin* ↵	to select the SN table containing solutions to be interpolated.
> GAINVER *clin* ↵	to select the CL version to which solutions are to be applied.
> GAINUSE *clout* ↵	to select the output CL version, containing updated calibration information.
> REFANT 5 ; BADDISK 0 ↵	Use a single reference antenna if at all possible during all steps of calibration.
> INP ↵	to check the inputs.
> GO ↵	to run the task.

It's a good idea to always apply the calibration information to both the cal and target sources when running CLCAL for phase-referencing observations. This allows you to monitor the cal source data to check the progress of the phase calibration procedure.

9.5.2 Processing observing log and calibration information

As of 1 April 1999, the VLBA correlator provides calibration transfer information, as described in § 9.3.1.2 for VLBA antennas. Experiments correlated after November 2003 also have full calibration transfer for the VLA, the GBT, Arecibo and the Bonn 100m. Consequently, **you can skip § 9.5.2 entirely** unless you have data from non-VLBA telescopes (*e.g.*, the VLA before November 2003, Space, other European telescope) or other correlators or you wish to process the log files manually for other reasons.

This section describes the processing of external calibration information, as supplied in ASCII log files. The information that may be used by *AIPS* includes T_{sys} or related total power measurements, edit flags as written by the tracking stations or on-line monitor control system, weather information, and pulse-calibration data. These external data can be read into *AIPS* by tasks ANTAB, UVFLG, APCAL, and PCLOD respectively, as described in § 9.5.2.3, § 9.5.3, § 9.5.4.6 and § 9.5.4.3.

You should have received information about where to obtain your calibration data. VLBA calibration log files may be obtained by ftp as described below. Similar calibration files for other participating antennas and VSOP should be obtained from the appropriate sites.

9.5.2.1 Automatic formatting of VLBA and VLBA-like log files

For VLBA antennas, the external calibration file for a given experiment can be downloaded from

`https://www.vlba.nrao.edu/astro/VOBS/astronomy/`*mmmyy*`/`*xxxxx*`cal.vlba.gz`

where *mmmyy* is the month and year (*e.g.*, aug03) and *xxxxx* is the project code (*e.g.*, bz199). The calibration file is named as above and is in GNU-zipped format. Gain curve information can be obtained from the same web site in the astronomy directory (*i.e.*, two directories up from the project directory). The gain curve should be concatenated to the external calibration file.

The external calibration file, if suitably close to the standard VLBA format, concatenated with the gain curve file, can be automatically subdivided and re-formatted to comply with *AIPS* requirements using task VLOG. Typical inputs would be:

> TASK 'VLOG' ; INP ⏎	to review the inputs.
> INDISK *n* ; GETN *ctn* ⏎	to specify the FITLD output file.
> SUBARRAY 1 ⏎	to select the required subarray.
> CALIN 'FITS:bz199cal.vlba ⏎	to specify the input external calibration file.
> OUTFILE 'FITS:BZ199' ⏎	to define the directory and prefix for the output files.
> FQTOL 1000 ⏎	to set the tolerance for frequency match in kHz; one channel width is recommended.
> PRTLEV 0 ⏎	to limit output; in particular to avoid echoing the calibration file to the screen.
> GO ⏎	to run the program.

A sequence of output text files will be created in the specified ($FITS here) directory named BZ199.* with suffixes:

1. .TSYS: T_{sys} calibration data, including gain curves, suitable for direct use by ANTAB. An INDEX record is constructed for each frequency ID if possible. If no match can be made to the FQ data, a warning message is printed and the INDEX keyword is omitted; the INDEX keyword must then be inserted by hand (see the HELP file for ANTAB). Gain curve entries for the frequency and time range in the *uv* data are copied to this output file. It is recommended to insert T_{sys} values at the end of scans immediately before source changes to avoid interpolation problems for sources of greatly

differing flux density or use INTERPOL = 'SELF' in CLCAL. Occasionally spurious data from previous observing runs or system start-up files will end up in the .TSYS file and must be edited out by hand.

2. .FLAG: Flag data, suitable for direct use by UVFLG. In the older format "antennas=vlba_xx" the appropriate antenna and day numbers are inserted.
3. .WX: Weather data, as used by APCAL if performing an opacity solution. This file is altered to conform with APCAL requirements (*e.g.*, WEATHER keyword) and lines with bad entries (*) are commented out.
4. .PCAL: Pulse-calibration data, for input to PCLOD. No editing is performed.

Files with suffixes .SCAN and .MKIII contain scan summaries and MkIII information and are for information purposes only.

9.5.2.2 Manual formatting of log files

Partitioning of the calibration file can and must be done by hand if the calibration file format is sufficiently distinct from the standard VLBA format or, possibly, if it contains multiple frequency bands. If your observation used non-VLBA antennas, you will need to edit the calibration text file manually to add any log information supplied for these antennas. The necessary steps are as follows:

Extract the flagging and T_{sys} information from the calibration text file. You will also need to append gain curves to the T_{sys} file. Try EXPLAIN ANTAB to see an example file in the proper format. The parameter TIMEOFF should be set to zero for each station since both flag information and data are stored with UTC times. The keyword DTIMRANG is also supported which pads each flagged time interval to insure that even very short flag intervals are applied.

While older VLBA format calibration files were supplied with the ANTENNA keyword, newer VLBA format calibration files are supplied with the ANT_NAME keyword. In the former case, the file should be edited to insert the antenna numbers as listed in the AN table (use PRTAN on your *AIPS* file to find these) and the absolute day numbers must be replaced by relative day numbers with respect to the *AIPS* reference date. In the latter case, no adjustments to either day numbers or antenna numbers are necessary.

9.5.2.3 Loading calibration log information

The calibration information in the external text files such as T_{sys} and gain curve measurements are read into TY and GC tables using ANTAB. These tables are then used by APCAL to generate an amplitude solution (SN) table, allowing an optional solution for atmospheric opacity. The user is advised to read the ANTAB help file closely and check the syntax of the text file carefully.

The INDEX keyword is used to assign the tabulated T_{sys} data to individual *AIPS* IF channels and polarizations. Up-to-date information on the usage of the INDEX keyword may be found by typing EXPLAIN ANTAB. Be careful to match up the proper polarization labels for the tabulated T_{sys} information. The frequency and polarization association for each IF channel in the *AIPS* file can be compared (use LISTR with OPTYPE = 'SCAN' or LISUN) with that at the head of the calibration text file.

The CONTROL group at the head of the calibration file is used only to specify a default index mapping. If the IF channel orders in the calibration file and the *uv* file are identical it is not required.

Source flux densities are not specified in the ANTAB input file. If source flux densities are required by APCAL, the source (SU) table will be searched. Use SETJY to insert flux densities if necessary.

The parameter TIMEOFF in the input file adds a time offset to the all entries. Non-VLBA stations sometimes measure the system temperature between, rather than during, scans causing ANTAB to be unable to match the measurements with the source and frequency ID. The ANTAB input parameter OFFSET serves the same purpose, but is more successful since the scan times are expanded at both ends.

ANTAB permits specification of IF-dependent and tabulated gains; the format description may be found by typing EXPLAIN ANTAB.

ANTAB can be run multiple times to append to the same TY and GC tables. Also, calibration files from separate antennas (*e.g.*, VLA) which have T_{sys} data tabulated in a different format can be concatenated and processed in one run. In this case the INDEX keyword must be specified for each antenna to fix the data format.

Note that ANTAB will (usually) ignore calibration data for which there are no corresponding uv data (see the help file for ANTAB). There is one exception however: calibration data for an antenna that does not appear in the AN table will cause ANTAB to fail. If ANTAB quits under such circumstances, you have two choices. You can edit the calibration text file, removing all reference to the missing antennas; or you can use the input adverb SPARM to specify explicitly the names of antennas for which there are calibration data, but which do not appear in the AN table.

**** The use of SPARM is no substitute for careful inspection of the calibration text files. ****

Having created the input text file, typical inputs for ANTAB would be:

> TASK 'ANTAB' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> CALIN 'MYVLB:BC25CAL.VLBA' ↵	to specify the text file.
> SUBARRAY 1 ↵	to select subarray one.
> TYVER 0 ; GCVER 0 ↵	to create new TY and GC tables.
> BLVER 0 ↵	to create new BL table for any specified baseline factors.
> PRTLEV 1 ↵	to select print level.
> GO ↵	to run the program.

PRTLEV = 2 will echo the calibration file as it is processed, which can be useful in locating format errors.

9.5.2.4 Generating VLA amplitude calibration with VLAMP

Before February 2013, the VLA's *a priori* calibration was done with calibration files, named *xxxxx*cal.y.gz (stored in gzipped format). These were obtained from the same server and disk directory as for VLBA files. This VLA file started with an explanatory preamble, including minor editing instructions. See § C.11 for more detailed instructions.

After February 2013 and the advent of the phased-EVLA, either an ANTAB style file was provided to the observer or the calibration information is in the data file. This calibration information is generated using the EVLA switched power (SY table, see § 4.3.15), the known gain curve (GC table), and the first CL table using the task VLAMP. Sometimes there are problems with the values in the SY, GC, and/or CL tables and the observer might want to recreate the amplitude calibration of the phased-EVLA after editing one or more of them. To do this the phased-EVLA only data (not the VLBI data) must be loaded into *AIPS* with BDF2AIPS, see § 4.1.1. Then the SY and/or GC tables can be edited. For instructions on how to edit or smooth the SY table see § 4.3.15. Then run VLAMP, which will create an output text file that can be loaded into the VLBI data with ANTAB. Typical inputs for VLAMP would be:

> TASK 'VLAMP' ; INP ↵	to review the inputs.

> INDISK *n* ; GETN *ctn* CR	to specify the input file.
> FLAGVER 1 CR	to apply flags to the input tables.
> INVER 1 CR	to specify the SY table.
> GAINVER 1 CR	to specify the CL table.
> IN2VERS 1 CR	to specify the GC table.
> VPARM 5 6 11 12 CR	to select the EVLA spectral windows used by the VLBA.
> OUTTEXT '*filename*' CR	to specify the name of the output text *filename*.
> GO CR	to run the program.

The RUN file and procedure named DOVLAMP can be used to run the full process, reading the data into *AIPS* with BDF2AIPS, smoothing and applying the SY tables with TYSMO and TYAPL, correcting standard calibration source amplitudes with special usage of CALIB and CLCAL, and finally running VLAMP to produce the needed file for ANTAB.

9.5.3 Data editing

Before proceeding to calibrate, you should first flag any obviously bad data. In summary, initial editing is based on the flagging information supplied by the on-line antenna monitor systems, which is applied using UVFLG. This information may be extracted to a .FLAG file as outlined in the previous section or subsequent editing based on the station report logs, or elevation limits can also be performed using UVFLG. Finally, graphical editing tasks such as EDITR and IBLED may be used for interactive baseline-based editing. In both single- and multi-source data formats, flagging information is stored in the FG table instead of being used to discard data directly. One may also undo a flag operation using OPCODEs 'UFLG', 'REAS', and 'WILD' in UVFLG. Note that this operation works *only* when the operation being undone is in the current flag table. (Note that many tasks delete fully flagged data when copying a data set.)

To edit *uv* data by reading a text file listing periods of known errors (*e.g.*, the .FLAG text file created by VLOG) run UVFLG with the following inputs:

> TASK 'UVFLG' ; INP CR	to review the inputs.
> INDISK *n* ; GETN *ctn* CR	to specify the input file.
> SOURCES '' CR	to flag all sources, which is usually desired.
> SUBARRAY 0 CR	to select the required subarray, all in this case.
> FREQID -1 CR	to flag all frequency IDs.
> OUTFGVER 2 CR	to specify the output flag table; use 2 *only* if you copied FLAGVER 1 to 2 as suggested in § 9.3.1.2.
> INTEXT 'MYVLB:BC25CAL.FLAG' CR	to specify the input text file.
> GO CR	to run the program.

This will generate a FG table with entries read from $MYVLB:BC25CAL.FLAG.

To edit *uv* data based on elevation limits, UVFLG can be used with input parameters:

> TASK 'UVFLG' ; INP CR	to review the inputs.
> INDISK *n* ; GETN *ctn* CR	to specify the input file.
> SOURCES 'DA913' , ' ' CR	to select one source.
> SUBARRAY 1 CR	to select subarray one.
> ANTENNAS 3, 4, 5, 8 CR	to flag only certain antennas.
> APARM 0, 0, 0, 0, 10 CR	to flag data between elevations 0 and 10 degrees (APARM(4) to APARM(5)), with no flagging on amplitude or weight.
> OUTFGVER 2 CR	to specify the output flag table.

> GO ↵ to run the program.

This will generate FG table flagging time ranges that fall between the specified elevation limits.

Note that in both of these examples, the flagging information is incorporated into a FG table instead of "irrevocably" deleting your data. In order to apply these flags, you must set FLAGVER to the appropriate FG table version.

Another editing task which may be useful is QUACK, which can edit the selected portion of the scan at its beginning and/or end. This might be needed because (at non-VLBA antennas) the telescopes were still slewing or system temperature measurements were being made. The first 20 seconds or so of a scan, for baselines to the VLA, are often unusable while the VLA correlator phases up at the new source position. The first second or so after a scan may also need to be flagged (OPCODE 'TAIL') since some antennas may not leave the previous source promptly, leaving bad data marked good. QUACK can be used to flag specific antennas for these and other reasons.

Tasks EDITA and EDITR can be used to inspect and edit the data interactively. IBLED is no longer of much use since EDITR now can edit on a coherence parameter. These tasks are similar in many respects to TVFLG, but are more suited to interferometers with small numbers of baselines. It is also possible to use TVFLG to perform your editing although the TV display is somewhat confusing on sparse arrays of data, especially if there is significant source structure. Read § 4.3.5 through § 4.3.11 and § 5.5 for more details on the editing of data. Also VPLOT may be used to edit data which deviates excessively from the mean amplitude over a specified averaging interval; see HELP VPLOT ↵ for details. Task WIPER is a dangerous but powerful editing tool as well.

The task DEFLG generates a flag table to delete data having coherence less than a user-specified limit. It must be run only on sources that should be strongly coherent although it may be used to flag data from other sources in between the strong-source scans. The task SNFLG is also used to flag data whenever the phase jumps in an SN or CL table are excessive on a baseline-by-baseline basis.

The VLBA correlator may produce a lot of samples which it knows to be bad and which are flagged by the transferred flag table. For this reason alone, or if you also flag a noticeable fraction of the data set, you may wish to run UVCOP to discard the flagged data to conserve disk space and processing time.

9.5.4 Amplitude and instrumental delay calibration

We now advocate a new amplitude calibration strategy base on VLBA Scientific memo #37 (Walker 2015). This strategy interleaves classic *a priori* calibration with instrumental delay and bandpass calibration to improve the amplitude calibration of data from the new Roach Digital Backend (RDBE) on the VLBA (see the VLBA Observations Status Summary for a description of the RDBE and VLBA Scientific Memo #37 for a discussion of amplitude problems with the RDBE). With data from before the RDBE you can use either the old or new strategy.

9.5.4.1 Parallactic angle correction

The RCP and LCP feeds on each antenna will rotate in position angle with respect to the source during the course of the observation for alt-az antennas (which probably constitute a majority of the antennas in your observation). Since this rotation is a simple geometric effect, it can be corrected by adjusting the phases without looking at the data. This correction must be performed before any phase calibration which actually examines the data is executed. This correction is important for polarization and phase-referencing observations, so it should probably be applied to all cases.. Task CLCOR is used for this purpose; see § 9.5.4.1.

There is a procedure which assists you in running CLCOR to correct phases for parallactic angle:

> RUN VLBAUTIL ↵	to acquire the procedures; this should be done only once since they will be remembered.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> SUBAR *ss* ↵	to select the subarray number — only one per execution.
> INP VLBAPANG ↵	to review the inputs.
> VLBAPANG ↵	to run the procedure.

VLBAPANG should be run before applying any other phase corrections.

9.5.4.2 Digital sampler bias corrections for VLBA correlator data

The voltage threshold levels in the digital samplers at the antennas may differ from their optimum theoretical values and this may vary from antenna to antenna and from polarization to polarization. This sampler bias, which is usually significant only in two-bit quantization, introduces an antenna/polarization-based amplitude offset. In full polarization observations this appears as an amplitude offset between RR and LL. The cross-correlation amplitudes may be corrected if the auto-correlation spectra have been measured. See VLBA Scientific Memo No. 9 (1995, "Effect of digitizers errors on the cross and auto correlation response of an FX correlator", by L. Kogan). Task ACCOR can be used to remove these digital sampler biases from VLBA correlator data, if FITLD was run with DIGICOR = 0, 1, or 2. ACCOR corrects these offsets by examining the autocorrelation spectra. You can check your autocorrelation data with task ACIMG which makes an image cube with axes of frequency, time, and antenna number. Use the image display options describe in Chapter 6 to inspect the image. There may be data that needs flagging. Since ACCOR computes the necessary correction by examining the total-power spectra, it must be run immediately after FITLD in the sense that nothing else should be run that actually modifies the total-power spectra directly. Although ACCOR ignores any SN or CL tables that are present, it is essential to correct for the sampler biases before performing any *a posteriori* calibration, *e.g.*, fringe-fitting or self-calibration. If the ACCOR gain factors deviate from unity, there may be overall scaling or *b*-factor errors. For 2-bit quantization, however, the amplitude offsets for the VLBA correlator are typically of order $5-10\%$, but values as high as 20% have been observed.

For one-bit quantization, no significant sampler bias correction is expected. Nonetheless, it is recommended that ACCOR be run on one-bit data as a consistency check. VLBACCOR is a procedure which runs ACCOR, smooths the results with SNSMO, and applies the solutions using CLCAL. Typical inputs would be:

> RUN VLBAUTIL ↵	to acquire the procedures; this should be done only once since they will be remembered.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> FREQID *ff* ; SUBAR *ss* ↵	to select the frequency ID and subarray numbers — only one of each per execution.
> INP VLBACCOR ↵	to review the inputs.
> VLBACCOR ↵	to run the procedure.

VLBACCOR will normally be run after "fixing" the the data with VLBAFIX (if needed). Use this procedure only on data from the VLBA correlator.

If you do not want to use VLBACCOR, typical inputs to ACCOR for this correction would be:

> TASK 'ACCOR' ; INP ↵	to review the inputs.
> INDISK *n*; GETN *ctn* ↵	to specify the input file.
> TIMERANG 0 ↵	to select data from all times.
> SOLINT 2 ↵	to set the solution interval for sampler corrections (min).
> GO ↵	to run the program.

The correction factors were expected to be fairly stable over time, but they have been found to vary over times less than an hour. With a solution interval of a few minutes, such as the two minutes indicated here, it is well to examine the solution (SN) table generated by ACCOR using SNPLT for any bad points or inconsistent values. One approach is to inspect the SN table and then run SNSMO with clipping to get rid of discrepant points. Alternatively, the interactive table editing task SNEDT can be used.

9.5.4.3 Instrumental phase corrections

If you run POSSM on a short ($\approx$ 1 minute) section of data on a strong calibrator using the inputs described in §9.4.2 and set DOCAL = -1 ↵, you will see that each individual IF channel has its own independent phase offset and its own phase gradient against frequency. These phase offsets and instrumental "single-band delays" are caused by passage of the signal through the electronics of the VLBA baseband converters (or MkIII/MkIV video converter units). The VLBA system can inject narrow band signals ("phase-cals") into the data recorded at each antenna from which the IF channel phase offsets, and the instrumental single-band delays, can be determined.

For those antennas for which phase-cal measurements are available, task PCCOR can be used to incorporate the phase-cals into an SN table. (see §9.5.4.3). If you have other antennas for which phase-cal measurements are not available, you can run CLCAL using the OPCODE='CALP' option to incorporate the incomplete SN table information loaded by PCCOR without throwing away data for the antennas with missing phase-cal information.

If external phase-calibration data (pulse cals) are not available then directly fringe-fitting a short scan of data to measure the phase and single-band delay offsets may be applicable under limited circumstances (see §9.5.4.4). Even when phase-calibration information is available, performing a manual phase-cal can be a good idea to confirm that the IF-dependent delays and phases have been successfully estimated and removed. One good check is to inspect the data using POSSM at times different from the time used to determine the manual phase calibration. Note that time-dependent delays may still be seen because of low elevation and ionospheric effects.

As of 1 April 1999, the VLBA correlator is capable of transferring phase-cal information directly to a PC table for some antennas. For other antennas, the phase-cal information may be read into the PC table using the task PCLOD. Type EXPLAIN PCLOD for further information. If you are unsure whether your VLBI data has phase-cal information, use IMHEAD to list the extension tables and look for a PC extension.

Phase-calibration data in a PC table can be used by task PCCOR to generate an SN table which corrects for the single-band instrumental phase and delay offsets (note that PCCOR uses two phase-cals in each IF). A short calibrator scan must be specified to be used by PCCOR to resolve any 2π phase ambiguities in the phase-calibration data. The specified time range must include at least one PC table entry for each antenna appearing in the PC table. Having resolved the 2π phase ambiguities, PCCOR uses the whole PC table to calculate entries in an SN table for all times (not just the TIMERANG used to resolve the ambiguities). The procedure VLBAPCOR runs PCCOR, FRING (if needed because of missing pulse-cals for some antennas), and CLCAL for you. Inputs are:

> RUN VLBAUTIL ↵	to acquire the procedures; this should be done only once since they will be remembered.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> TIMERANG *d1 h1 m1 s1 d2 h2 m2 s2* ↵	to specify a short scan on a calibrator. There is no default.
> REFANT *m* ↵	to select a particular reference antenna.
> SUBARRAY 0	to do all subarrays.
> CALSOUR *'cal1'*, ' '	to specify the calibrator source name.
> GAINUSE *CLin* ↵	to indicate the CL table with all calibration up to this point.

> OPCODE 'CALP' ↵	to indicate that there are antennas with no usable pulse cals; use OPCODE ' ' if all antennas have pulse cals.
> ANTENNAS *a1 a2 a3* ↵	to solve for antennas *a1*, *a2*, *a3* "manually" (using FRING).
> VLBAPCOR ↵	to run the procedure.

This should be done after the sampler correction (VLBACCOR) and before the complex bandpass, The CALP option requires the data in the specified TIMERANG to include strong fringes for those antennas lacking phase-cal data.

Running the tasks individually, typical inputs to PCCOR are:

> TASK 'PCCOR' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> TIMERANG 1 2 15 0 1 2 20 0 ↵	to isolate a short scan on a calibrator; *no default*.
> SNVER 0 ↵	to create a new SN table.
> INVER 1 ↵	to specify the input PC table version.
> REFANT 5 ↵	to specify reference antenna.
> SUBARRAY 1 ; FREQID 1 ↵	to set subarray and freqid.
> CALSOUR '3C345',' ' ↵	to set calibrator source name.
> GO ↵	to run the program.

The resulting solution table is applied using CLCAL. If you are missing phase-cal information for some antennas, you must use the 'CALP' mode of CLCAL; this mode allows calibration information for some antennas to be incorporated into the CL table while passing other antennas through without modification. Examine the corrected data using POSSM to determine if the instrumental phase and delay offsets between the IF channels have been removed correctly.

To check that the applied phase-cals are valid, run POSSM on a short section of data containing a strong source setting DOCAL = 1, GAINUSE to the version number of the CL table containing the phase-cal calibration, and APARM(9) = 1 to place all IFs on the same plot. The phase as a function of frequency on each baseline should be smoothly varying, with no sharp jumps between different IF channels. There may be an overall linear gradient with frequency due to residual delay errors. Unless these conditions hold for all baselines, you should proceed to § 9.5.4.4.

The DiFX correlator has the capability of writing a large number (1 per MHz typically) of pulse cals into each IF. *AIPS* can handle this with new tasks PCFLG (flags PC data with a SPFLG-like TV display), PCEDT (flags PC data with a BPEDT-like TV graphical display), PCPLT (plots PC table data all antennas with one time per page or all times with one antenna per page), PCASS (finds *amplitude* bandpass), and PCFIT (fits delays and phases to PC data, writing an SN table of the changes in these as a function of time). PCLOD can now read PCAL files written by DiFX. Other tasks include PCAVG to average pulse-cal tables in time and PCRMS to edit pulse-cal tables automatically from internal statistics. POSSM can also now plot PC table data (APARM(8) = 9. This package of routines remains experimental, but it has been shown do good things with the data. It is now documented in *AIPS* Memo 123.[2] New tasks in 31DEC21 include VLBSY to read VLBA total power files, VLBCD to read noise tube values from a VLBA cal file, PCABL to copy cable cal values from the old PC table to ones created by PCLOD, and PCDIV to scale pulse-cal amplitudes by the system temperatures.

9.5.4.4 "Manual" instrumental phase corrections

If your file does not have phase-cal information, or if these phase-cals do not successfully remove frequency phase offsets, you can use observations of a bright calibrator source and the task FRING to correct for these

[2] Greisen, E. W. 2025 (revised), "New Pulse-cal Capabilities for VLBI in *AIPS*" *AIPS* Memo 123, https://www.aips.nrao.edu/aipsdoc.html

effects. If you attempted phase-cal calibration, it is best to avoid possible confusion by first deleting any partial or erroneous phase-cal information that already exists. Using CLCOR:

> TASK 'CLCOR' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> GAINVER *clin* ; GAINUSE 0 ↵	to specify which CL table to modify after copying it to a new table.
> OPCODE 'PCAL' ; CLCORPRM 0 ↵	to set phase-cals to zero.
> INP ↵	to check the inputs.
> GO ↵	to run the task.

If you have large known delay offsets you may also wish to run CLCOR using OPCODE='SBDL' to shift the center of the fringe search window.

If you have a known clock offset, as may be common for an SVLBI data set, you may wish to run CLCOR with OPCODE = 'CLOC' and CLCORPRM = 0, *offset*, 0, 0, 0, 0, 1, 0.

Now you can determine the manual phase correction for one or two strong calibrator scans for which all the antennas are present using VLBAMPCL. Task BSCAN may be used to select the scan(s) to be used at this stage. VLBAMPCL runs FRING and CLCAL once or twice, depending on whether one or two scans are used. Choose a scan with strong fringes to all antennas; if none exists, find a second scan that has strong fringes to the antennas missing from the first. Note that if you use 2 scans the REFANT must have good fringes in both scans. Typical inputs are:

> RUN VLBAUTIL ↵	to acquire the procedures; this should be done only once since they will be remembered.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> TIMERANG *d1 h1 m1 s1 d2 h2 m2 s2* ↵	to specify a short scan on a calibrator. There is no default.
> REFANT *m* ↵	to select a particular reference antenna.
> SUBARRAY 0	to do all subarrays.
> CALSOUR '*cal1*', ' '	to specify the calibrator source name.
> GAINUSE *CLin* ↵	to indicate the CL table with all calibration up to this point.
> OPCODE 'CALP' ↵	to indicate that there are antennas with no fringes for the scan in TIMERANG; use OPCODE ' ' if all antennas will be corrected by the first scan.
> TIME2 *d1 h1 m1 s1 d2 h2 m2 s2* ↵	to specify a short second scan on a calibrator.
> CALSOUR '*cal2*', ' '	to specify the calibrator source name for the second scan.
> ANTENNAS *a1 a2 a3* ↵	to solve for antennas *a1*, *a2*, *a3*.
> VLBAMPCL ↵	to run the procedure.

If you wish to run FRING separately, you can determine the phase offsets/single band delays by running FRING on a short section of calibrator data where all or most of the antennas are present. Suitable inputs for FRING for this purpose are shown below; for more details of some of the FRING input parameters see § 9.5.7.7. Note that it is simplest to choose a single short section of data within a single scan using TIMERANG and to set SOLINT equal to the scan length so that a single solution is achieved. The interval chosen must be less than than an atmospheric coherence time, but long enough that high signal-to-noise is achieved. At centimeter wavelengths with Jansky-level calibrators, solution intervals of a few minutes will work well. For example:

> TASK 'FRING' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> CALSOUR '0954+658 ' , ' ' ↵	to select a strong calibrator source.
> TIMERANG 0, 16, 0, 0, 0, 16, 2, 0 ↵	to select a single short scan.

> DOCALIB 1 ; GAINUSE 2 CR	to apply the amplitude calibration from CL table 2 to the weights as well as the visibilities.
> FLAGVER 0 CR	to apply the most recent flag table.
> SMODEL 0 CR	to use the null (point-source at the origin) source model.
> REFANT 5 CR	to specify a reference antenna that will give fringes to most other antennas.
> SOLINT 0 CR	to set the solution interval in minutes; do not exceed the atmospheric coherence time.
> APARM 0; DPARM 0 CR	to initialize FRING options to defaults.
> APARM(1)=2 CR	to require at least 2 antennas.
> APARM(6)=1 CR	to solve for the rate, single-band delay and phase of each IF separately.
> DPARM(1)=1 CR	to use only 1 baseline in the initial (coarse) fringe search.
> DPARM(9)=1 CR	to suppress fitting of rates; rates will be 0 in the SN table.
> SNVER 0 CR	to create a new SN table.
> ANTWT 0 CR	to apply no additional weights to the antennas before doing the solutions.
> INP CR	to check the inputs.
> GO CR	to do the fit.

If there was no single scan where all the antennas were present, you can run FRING again for another scan setting REFANT to be one of the antennas found in the first run (the same REFANT would be best) and ANTENNAS to this antenna plus all of the antennas *not* found in the first run. FRING will generate a new SN table each time; be careful to keep track of which SN tables you wish to use.

The phase solutions in the SN table(s) are interpolated onto a calibration or CL table using task CLCAL as described in §9.5.1.1. To apply the calibrator solutions to the other sources in the data file, set CALSOUR to the calibrator source used when running FRING (in the example above, CALSOUR '0954+658') and set SOURCES = ' ' for all sources. Also, set REFANT to whatever was used when running FRING. If multiple runs of FRING were required, you can set SNVER=0 so that all SN tables are combined before being applied ; if you do this, you must first be careful to delete all SN tables except those generated by FRING. Or you can run CLCAL multiple times specifying each SN table in turn, with the specific antenna numbers, while keeping the same CL table versions.

To check the output CL table, run POSSM for the scan used for the FRING solution with DOCAL = 1 and GAINUSE = 3 (*i.e.*, the output CL table from the CLCAL above). The phase should be flat as a function of frequency on all baselines, although it may not be centered on zero. Run POSSM on another scan containing a strong calibrator to check that the assumption of constant IF phase offset holds.

9.5.4.5 Bandpass calibration

Full bandpass response calibration should be performed for all observations. It is suggested that integrating over a variable bandpass function is one of the most significant sources of non-closing errors in continuum VLBI data. By calibrating the bandpass before averaging over frequency, these effects can be avoided. In VLBA test observations, a dynamic range of 28,000:1 was achieved on the source DA193 (Briggs *et al.* 1994, VLBA Memo No 697A, "High Dynamic Range Imaging with the VLBA") after applying bandpass calibration.

Bandpass calibration is carried out using the task BPASS using either the auto- or cross-correlation data. The output is a BP or bandpass table. For this amplitude calibration scheme the cross-correlation (complex)

bandpass must be used. The derived bandpass solutions can be plotted using POSSM by setting APARM(8)=2. The effect of applying these bandpass solutions to your data can also be viewed using POSSM by setting DOBAND=1 and BPVER. An example of the inputs to produce bandpass spectra from the cross-correlation data would be:

> TASK 'BPASS' ; INP ↵	to review the inputs.
> INDISK *n1* ; GETN *ctn1* ↵	to select the multi-source visibility data as the input file.
> CALSOUR 'BLLAC' , 'DA193' ↵	to specify the continuum source(s) which were observed for the purpose of bandpass calibration.
> DOCALIB 1 ↵	to apply calibration.
> SOLINT 0 ↵	to average data over whole scans before determining the bandpass.
> BPASSPRM 0 ↵	to use the cross-spectra.
> BPASSPRM(5) 1 ↵	to not divide by "channel 0."
> BPASSPRM(9) 1 ↵	to interpolate over flagged channels.
> BPASSPRM(10) 6 ↵	to normalize the amplitude solutions using the *power* solutions rather than the actual solutions which are voltages.
> ICHANSEL $chan_{beg}$, $chan_{end}$, 1, IF_{num}, ... ↵	to set channels for entire bandwidth for normalization, if this left 0 then the inner 75% is used and can cause up to 15% error in the amplitude, so unfortunately ICHANSEL must be explicitly set to all the channels in the IF, for every IF.
> GO ↵	to run the program.

Be careful with the adverb SMOOTH. If you smooth, or do not smooth, the data while finding a bandpass solution, then you must apply the same SMOOTH adverb values whenever you apply that bandpass solution to the data. The only exception is that you may smooth the data after applying the bandpass solution with SMOOTH(1) values 5 through 8 when you did no smoothing in BPASS.

This will produce a BP table containing the antenna-based bandpass functions to be applied to the data. You should check your results very carefully. The BP tables can be plotted with POSSM or printed with PRTAB. Note that this task merely creates a BP table. To use this BP table, set BPVER and DOBAND as described in § 4.3.8 when running any later *AIPS* tasks.

Note, if your bandpass calibrator is not stable or strong enough during the observation (your bandpass calibrator data should have a better S/N than the data you're trying to correct with the bandpass), you could consider using the phase reference source (if there is one) and use SOLINT = -1 (include all scans to make one bandpass solution). If the bandpass calibrator is strong enough, but the average phase varies through the scan, then divide each record by "channel 0" by setting BPASSPRM(5) = -1 to adjust phase only. You should select a range of channels that have similar phases to be averaged as channel 0 using adverb ICHANSEL

The normalization of the bandpass correction can be checked by running ACSCL after BPASS. This task is a version of ACCOR which applies the current calibration tables to the data before forcing the autocorrelations to unity. Be sure to set DOBAND to apply the bandpass by your preferred method. ACSCL will write an SN table with amplitude corrections which should be very close to one. Apply that table with CLCAL. However if you run VLBAAMP, ACSCL and CLCAL will be run for you, see § 9.5.4.6.

The interactive data editing task BPEDT lets one flag visibility data based on bad bandpass solutions. It can edit the BP table instead and even interpolate over the flagged values. It is a graphical editor much like EDITR and EDITA except that it displays bandpass solutions as a function of spectral channel. It is a quick way to check your bandpass table and, if necessary, to flag bad channels in your calibration source. If you do generate new data flags, you should run BPASS over again to generate a corrected BP table. Set BPASSPRM(9) = 1 to interpolate over any fully flagged channels. For really large BP tables, you may wish

to use BPFLG instead after determining clip levels with BPEDT although BPEDT has been made much faster for such BP tables.

9.5.4.6 Continuum amplitude calibration

After bandpass calibration is determined, the auto-correlations can be slightly offset from unity. To correct this offset ACSCL can be run, applying all the previous amplitude and bandpass calibration. Described below is a procedure that is part of VLBAUTIL called VLBAAMP that runs ACSCL as well as the final amplitude calibration steps.

The TY table can be examined using SNPLT with OPTYPE='TSYS' or 'TANT' or LISTR with OPTYPE='GAIN'; the GC table is examined with PRTAB. SNEDT may also be used to inspect the TY table and even used to delete or smooth some of the measurements. SNRMS may be used to examine statistics of the TY table. TYSMO can be used to delete discrepant system temperatures and replace the bad and/or good values with time-smoothed values. Note, however, that anomalously high system temperatures may indicate possible bad data (*e.g.*, due to weather) rather than bad measurements of the system temperature. On occasion, RFI in some IFs can render the values for those IFs in the TY table unusable. Task TYCOP will copy TY data between IFs and polarizations in a variety of ways in an attempt to replace the bad data with something reasonable even if not exactly correct for that antenna and IF. Note that flagging bad *uv* data will not change the appearance of the TY table since no flags are applied in plotting this table. EDITA may also be used to examine the TY data interactively and, if bad system temperatures are found, to flag the associated *uv* data. (This task does flag displayed $T_{\rm sys}$ when the data are flagged.) Tasks UVCOP and SPLAT will apply a flag table not only to the *uv* data, but also to the TY and SN tables. APCAL can then be used to derive an amplitude calibration (SN) table.

Atmospheric opacity becomes significant at high frequencies ($\geq$ 15 GHz). APCAL can fit for an opacity correction, if needed, using weather information and system temperature. The weather information can be taken from the WX table by setting INVERS (after October 7, 2003) or loaded from disk by setting CALIN. In order to have APCAL fit for opacity set OPCODE to 'GRID', 'OPAC' or 'LESQ' and DOFIT to 1 (both of these are necessary for an opacity fit). OPCODE 'GRID' and 'OPAC' need an initial guess for the receiver temperature (TRECVR and zenith opacity (TAU0). APCAL will estimate the initial guesses from the data if TRECVR and/or TAU0 are 0. If OPCODE is set to 'GRID', 'OPAC' or 'LESQ' and DOFIT $\leq$ 0 then the opacity correction is applied using the provided TRECVR. This is a good option if you have a reliable measurement of the receiver temperature. In 31DEC20, use OPCODE 'PLOT' to plot the data against the solution you wish to apply to make sure it is reasonable. The fits are fairly robust, but the plots that APCAL makes should be examined. Note: a large number of bad Tsys values can make the fits unreliable. APCAL will warn you if fits appear to be incorrect. The procedure VLBAAMP runs ACSCL, smooths the results with SNSMO, runs APCAL, and applies the solutions using CLCAL.:

> RUN VLBAUTIL C$_R$	to acquire the procedures; this should be done only once since they will be remembered.
> INDISK *n* ; GETN *ctn* C$_R$	to specify the input file.
> FREQID *ff* ; SUBAR *ss* C$_R$	to select the frequency ID and subarray numbers — only one of each per execution.
> INP VLBAAMP C$_R$	to review the inputs.
> DOFIT 1 C$_R$	to enable the opacity correction; $\leq$ 0 disables it.
> VLBAAMP C$_R$	to run the procedure.

VLBAAMP will normally be run after correcting the bandpass and loading any gain curves or system temperature data for non-VLBA antennas using ANTAB.

If you have not used VLBAAMP and do not want to correct for atmospheric opacity, typical inputs to APCAL are:

> TASK 'APCAL' ; INP C$_R$	to review the inputs.

> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> ANTENNAS 0 ; SUBARRAY 0 ↵	to select all antennas and subarrays.
> SOURCES '' ; STOKES '' ↵	to select all sources and Stokes.
> BIF 1 ; EIF 0 ; FREQID 1 ↵	to select all IFs of frequency ID 1.
> TIMERANG 0 ; OPCODE '' ↵	to select all times and use no opacity solutions.
> TYVER 0 ; GCVER 0 ↵	to use the latest TY and GC tables.
> SNVER 0 ↵	to create a new SN table.
> GO ↵	to run the program.

If atmospheric opacity correction is desired, set the following inputs as well as the above:

> OPCODE 'GRID' ↵	do a grid search.
> DOFIT 1 ↵	to fit the opacities.
> INVERS 1 ↵	to use WX table 1.
> TAU0 0 ; TRECVR 0 ↵	to let APCAL estimate initial values.
> GO ↵	to run the program.

The resulting solution (SN) table can be smoothed and clipped using SNSMO and applied using CLCAL as described in §9.5.1.1). Substantial smoothing of the TY table (task TYSMO), especially for VLBA-only observations, is not generally recommended since variations of the system temperature often reflect a real response to the weather. Note, however, that TYSMO is very useful for removing bad system temperatures before they are applied. Smoothing can be useful for data from the phased-VLA, when the amplitude calibration information reflects low signal-to-noise. For non-VLBA antennas, it is important to check with SNEDT that the TY information is associated with the correct source and SNEDT can be used to delete occasional bad system temperature measurements before they are applied to the data. When opacities are fit, APCAL generates plots of the receiver temperature versus zenith angle and the opacity versus time. APCAL will warn you about bad fits, but the plots should be checked for problems with the data or the fits.

On occasion, it has been noted that plots of system temperature show unreasonable variation between IFs. This suggests that the T_{cal} values may have systematic errors as a function of frequency. APCAL has an option to bring the T_{sys} values into better agreement with either their average or with a user-chosen IF, thought to be probably accurate. APARM(6) > 0 selects this function while APARM(7) may be used to specify a specific IF to calibrate the others. Note that a single value is used for each antenna/IF over all times.

The DiFX correlator in now capable of correlating multiple phase-stopping positions within a single antenna beam (pointing position). There is task called CLVLB designed to apply corrections to a CL table (writing a new one) for the offset of phase-stopping and pointing position. The task has internal tables for the beam shape and squint of the VLBA antennas and can take user input for these parameters for other antennas. This is a new task and should be used with caution — early results with it were less than satisfying — although the corrections it applies are certainly in the right direction.

9.5.5 Spectral-line Doppler correction

Normally, when observing, you will have kept the frequency constant throughout the run for ease of observing. Therefore, although your data will have the same frequency, the center velocity of your spectrum will change with time and the spectral-line signals will wander backwards and forwards through the spectrum. To ensure that the velocity is constant throughout the data you should run SETJY and then CVEL. The VLBA correlator compensates for the revolution of the antennas relative to the center of the Earth. In this case, the only movement which CVEL should compensate is the rotation of the antennas together with the Earth around the Sun. This movement should be the same for all antennas and gives a smaller effect than the rotation relative to the center of the Earth. Nonetheless, CVEL is required for the VLBA correlator at least to be able to compare observations made at different times. Without CVEL such comparisons will show the velocity shift of the Earth's orbit about the Sun.

SETJY will insert the velocity information required in the SU table:

> TASK 'SETJY' ; INP ↵	to review the inputs.
> INDISK *n1* ; GETN *ctn1* ↵	to select the data.
> SOURCE 'OH127.8' , ' ' ↵	to specify the line source whose velocity is to be specified.
> OPTYPE ' ' ↵	to switch off flux modification.
> SYSVEL -66.0 ↵	to specify the velocity of the "center" of the band in km/s.
> APARM 65, 0 ↵	to specify which spectral pixel is the "center" of the band, actually the pixel to which SYSVEL refers.
> RESTFREQ 1612e6, 231.09e3 ↵	to give the rest-frequency in Hz, *e.g.*, that of the OH transition. Note that the two single-precision adverb numbers are summed in double precision inside SETJY.
> VELTYP 'LSR' ↵	to select the rest frame of the velocity.
> VELDEF 'OPTICAL' ↵	to define velocities by the optical convention.
> GO ↵	to run the program.

Then run CVEL:

> TASK 'CVEL' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to select the data.
> OUTDISK 3 ; OUTCLASS 'CVEL' ↵	to specify the output file.
> SOURCE 'OH127.8' , ' ' ↵	to select the source(s) to be shifted, all others will be passed un-shifted.
> DOBAND 1 ↵	to apply the bandpass correction — important.
> BPVER 1 ↵	to specify the version of BP table to use.
> GO ↵	to run the program.

After applying the BP table, CVEL will not copy it to the output file to protect you from applying it twice. Although CVEL allows you to select which sources are to be shifted, the BP table, if DOBAND is set appropriately, will be applied to all sources found.

9.5.6 Spectral-line amplitude calibration

The calibration strategy suggested in *AIPS* for spectral-line VLBI data utilizes the total-power spectra method described in Lecture 12 of *VLBI, Techniques and Applications*, eds. Felli and Spencer, published by Kluwer Academic Publisher, 1988. The continuum method using T_{sys} values (see § 9.5.4.6) can also be used. The first step is to generate a so-called template spectrum. This is a high quality spectrum from the most sensitive antenna in the array that has been corrected for the effects of the bandpass filter. For example:

> TASK 'SPLIT' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> SOURCE 'OH127.8', ' ' ↵	to write the program source.
> BCHAN 0 ; ECHAN 0 ↵	to write all spectral channels.
> DOCALIB -1 ↵	to avoid applying any calibration.
> DOBAND -1 ↵	to skip the bandpass correction since the data will have been corrected already in CVEL.
> TIMERANG 0 22 0 0 0 22 30 0 ↵	to select the data from a range of times when the antenna elevation was high and the source spectrum of high quality.
> APARM 0, 0, 0, 0, 1 ↵	to pass only self-spectra.
> GO ↵	to run the program.

You should then run ACFIT to do a least-squares fit of the template total-power spectrum to the total-power spectra of all other antennas and to write the resulting time-dependent amplitude gain correction factors into an SN table.

> TASK 'ACFIT' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> IN2DISK *n* ; GET2N *ctn* ↵	to specify the template file.
> CALSOUR 'OH127.8 ' ↵	to select the source to use for calibration.
> DOCALIB -1 ↵	to avoid applying any previous calibration.
> DOBAND -1 ↵	to skip the bandpass correction since it was done when CVEL was run.
> SOLINT n ↵	to average the self-spectra over n minutes (*e.g.*, 10) before doing the least-squares fit.
> REFANT 1 ↵	to select the desired reference antenna from the template file.
> BCHAN 50 ; ECHAN 70 ↵	to set the range of spectral channels over which the fit is performed.
> BPARM 80, 120 ↵	to set up to 5 pairs of start and stop channels to use in determining the baseline polynomial to be removed from the source spectra. The order of the polynomial is specified in APARM(1).
> CPARM 80, 120 ↵	to set up to 5 pairs of start and stop channels to use in determining the baseline polynomial to be removed from the template spectra. The order of the polynomial is specified in APARM(2).
> XPARM 45.1, 48.0, 50.1, 49.5 ↵	to specify T_{sys} values in the first polarization for each IF for the template scan of the reference antenna. YPARM provides an equivalent list for the data, if any, from a second polarization.
> APARM 0, 0, 50, 0, 0.72 ↵	program control: APARM(1) and APARM(2) specify the orders of the polynomial spectral baseline to remove from the source and template spectra; APARM(3) and APARM(4) specify the sensitivity of the template antenna (in Jy/deg) in the first and second (if needed) polarizations; APARM(5) and APARM(6) specify the minimum and maximum relative antenna gains allowed, with defaults to allow all positive values; APARM(7) specifies the maximum allowed gain error, with 0 meaning all; APARM(8) specifies the print level, with 0 providing minimal information, 1 providing useful information on the gains determined for each antenna and solution interval and 2 giving the gory details for each fit; APARM(9) specifies that the fits are done after subtracting a spectral baseline (0) or without a baseline (1); and APARM(10) controls whether baseline-subtracted spectra are written to an output file.
> SNVER 0 ↵	to create a new SN table into which the solutions are to be written.
> GO ↵	to run the program.

ACFIT will generate an SN table, which has to be applied to the CL table. If needed, run SNSMO to smooth the amplitude correction factors determined by ACFIT before running CLCAL. A 30-minute smoothing interval for SNSMO (set using CPARM 0.5,0) should be sufficient.

9.5.7 Phase calibration

After carrying out amplitude calibration, the remaining calibration steps involve correcting the phase between the different integration periods. This will allow averaging of the data over both frequency and time without loss of coherence. The phase offsets may be corrected using *a priori* 'phase-cal' measurements if available and/or by directly fitting to the data.

After removing instrumental phase offsets from each IF, the data will in general still contain frequency and time dependent phase variations. The purpose of "fringe-fitting" is to determine these phase errors and then remove them from the data.

The primary AIPS task for fringe-fitting is FRING. This task estimates time variable station-based delays (phase derivatives wrt frequency) and rates (phase derivatives wrt time) using a self-calibration-like algorithm. Once these delays and rates are determined, the task CLCAL is used to produce the phase correction that should be applied to each integration period and spectral channel to correct for delay and rate effects. This use of FRING and CLCAL is discussed in detail in § 9.5.7.7. Two alternatives to FRING are the tasks BLING and BLAPP and the experimental version of FRING named KRING. BLING and BLAPP are discussed briefly in § 9.5.7.8. KRING provides a superset of the functionality in FRING with numerous enhancements such as: the use of extremely small scratch files, a parsimonious use of memory, possible solution extrapolation both backwards and forwards in time and a rationalized definition of SNR. For more information about KRING, type HELP KRING from within AIPS. When fringe-fitting to many small scans, KRING can be substantially slower than FRING. When fringe-fitting data sets with large numbers of spectral channels and long solution intervals, KRING can be substantially faster than FRING.

The process of fringe-fitting, and then interpolating the solutions using CLCAL, can be a very time consuming process. For this reason it is probably wise to run through the fringe-fitting procedure described in § 9.5.7.7 on a small amount of data first (say 30 minutes' worth) before attempting to process the whole data set. This is especially true if this is your first time processing multi-IF, multi-channel VLBI data. It is probably simplest to use UVCOP to copy out a short time range of data from your main file and to work only on this initially. Doing so also avoids the possible confusion of having many versions of extension tables. Note that, if your spectral channels are narrow, the use of the CHINC parameter can speed up FRING.

9.5.7.1 Special considerations: SVLBI

The existing fringe-fitting tasks within AIPS have been enhanced to improve their performance when dealing with SVLBI data. In addition, several new tasks have been written to address problems specific to SVLBI fringe-fitting. The primary SVLBI fringe-fitting tasks in AIPS are BLING and FRING and are discussed in § 9.5.7.7–§ 9.5.7.9. The tasks COHER and FRPLT, described in § 9.4.6 and § 9.4.7, may be of particular interest when reducing SVLBI data.

There may be delay discontinuities in the recorded data for a variety of reasons such as tracking station handoffs, clock glitches, etc. The recommended method for dealing with such discontinuities is to force scan boundaries at such events. The task INDXR can be used to generate a new NX table with scan boundaries at desired locations using an input text file. In practice, either INDXR will do the right thing by design, or your P.I. information letter should have contained instructions on how to construct a text file in the proper format for INDXR.

A new task, OBEDT, is available which allows selection of specific orbital parameter ranges, through the creation of an output flag (FG) table. This can be used to constrain initial fringe searching.

SVLBI data often contain tracking passes three or four hours in length, for which fringes are mostly (or wholly) not apparent. Typically, the space-ground baselines will have the highest correlation coefficients

near perigee, when those baselines are shortest. However, the imperfectly known orbit will cause high fringe rates and short coherence times. Near apogee, the coherence time is longer, and may be limited by the atmosphere above the ground telescopes, but the correlated flux also is much lower. Sometimes, it may be possible to find fringes for only 15 or 20 minutes, but that's better than nothing.

If no fringes are seen anywhere during a tracking pass, a useful trick is to set `APARM(7) = 0.01` to let through the highest SNR for each solution interval, and set `DPARM(5) = 1` to turn off the least squares solution. Then run `FRING` for an entire tracking pass, and use `SNPLT` with `OPTYP = 'DELA'` and `OPTYP = 'RATE'` to look for repeating values (usually easier to see in delay than in rate). Also, make plots with `OPTYP = 'SNR'` to see if slightly higher SNRs are found at a time when there seems to be some consistency in delay values. It may be necessary to try this process with several values of `SOLINT` in order to arrive at a guess for the fringe location. Use `VPLOT` to plot *uv* distance versus time for the space-ground baselines, then search using the `TIMERANG` and `REFANT` that provide the shortest projected baselines. Another possibility is to set `REFANT = ANTNUM('MK')`, since the atmospheric coherence time should be longest at Mauna Kea, and increase `SOLINT` to a fairly large value in hopes something will show up. (Note, however, that a large `SOLINT` with a wide-open search window in delay and rate may require a large-memory computer or great inefficiency due to page faulting.)

If fringes are found somewhere, use `CLCOR` to center the fringes (see § 9.5.4.4), then run `FRING` again with small delay and rate windows (*e.g.*, `DPARM(2) = 200` to 400 and `DPARM(3) = 40` to 80 at 1.6 GHz, or ~ 200 at 5 GHz). Set low SNR thresholds with `APARM(7) = 3.5`, and turn the least-squares solutions back on with DPARM(5)=0. Usually, it's best to turn on the exhaustive antenna search with `APARM(9) = 1`, since only a few space-ground baselines may show fringes. It can be helpful to use `SEARCH` to order the search from shortest to longest space-ground baselines.

It generally doesn't work well to use one tracking station to predict the results of another, because clock initialization offsets are typically relatively large and have unrelated errors, and fringe-rate errors also may be unrelated.

9.5.7.2 Special considerations: spectral-line

Delay and fringe-rate calibration of spectral-line VLBI data must be handled differently. The residual delay cannot be estimated from the source itself because, due to the very nature of the source, the delay is a rapidly varying function of frequency. The continuum calibrator, observed for this purpose, is first used to determine residual delays and fringe-rates which are then applied to the spectral-line source. A suitable channel or range of spectral-line channels "on-source" is then used to determine the residual fringe-rates. It is very important to note that in some situations, the residual fringe-rates determined from the calibrator may not be applicable to the line source because the fringe-rate residuals towards the two sources may be quite distinct. In such situations, the residual fringe-rates determined from the continuum source should not be applied to the line source. See § 9.5.7.10 for more details.

9.5.7.3 Special considerations: polarization

In addition to phase calibrating the LL and RR data separately, for polarization data the R-L phase and delay offsets must also be determined. This is outlined in § 9.5.7.11. After fringe-fitting, all parallel-hand fringe solutions need to be re-referenced to the same antenna.

9.5.7.4 Special considerations: phase-referencing

The process of phase referencing for VLBI data is conceptually very simple. Unfortunately, the technical difficulties in conducting a successful phase-referencing observation are primarily in setting up the schedule. So by the time you get around to reading this section, your project is either guaranteed to succeed or guaranteed to fail, depending upon how well your observations were designed. See the lecture by A. Beasley and J. Conway in "VLBI and the VLBA", 1995, (ASP), and VLBA Scientific Memo No. 24 (2000, by J. Wrobel, C. Walker, J. Benson, and A. Beasley) for more details on how to design phase-referencing observations.

In "phase referencing," the phase calibration for your target source is derived from a calibrator, or phase referencing, source observed for that purpose. First, you apply any available *a priori* phase-cals to both the target and cal source. Next, you fringe-fit, self-calibrate, and/or hybrid-map the cal source — whatever is needed to complete the phase calibration for that source. Finally, you apply the phase corrections so determined to the target source. In practice this is done by specifying the target as well as the cal source in the SOURCES adverb list whenever an SN table containing phase corrections for the cal source is applied using CLCAL. See § 9.5.1.2 for a specific example of how to run CLCAL to transfer phase corrections determined using a cal source to a target source.

Certain "instrumental" corrections such as those for unmodeled zenith delay may have subtle but significant effects on phase referencing. See § 9.5.7.5 for a discussion.

You can perform a "hybrid" form of phase referencing in some instances. It may be that your target source is too weak for initial fringe-fitting. In this case, you can fringe-fit the cal source data to determine the phase corrections to be applied to the target source data. Then, after averaging in frequency, the target source data may have adequate signal-to-noise to allow rate corrections to be determined for it by fringe-fitting. In this mode, you may or may not wish to zero the rate corrections determined on the cal source. If the cal source is "far" from the target source, the rate corrections may do more harm than good for the target source and should be zero'ed. On the other hand, your target source may be entirely too weak to fringe-fit on at all. In this case, you must rely on determining the phase corrections solely using your cal source.

If you are attempting phase-referenced astrometry, you may have a target source that is brighter than your cal source(s). In this case, you simply fringe-fit on the target source and transfer the solutions to the cal source(s). Be careful, if your goal is to extract absolute positional information, not to independently self-calibrate the cal and target sources.

9.5.7.5 Correcting for atmospheric delays

VLBI correlators remove some estimate of the non-dispersive atmospheric delay at the elevation and frequency of the observation from the data. These a priori models are usually fairly good, but careful observations can improve upon them. *AIPS* offers a number of options to deal with this problem. DELZN uses multi-band delay (see § 9.5.7.6) in an SN table to fit for the zenith tropospheric delay and the clocks as a function of time. It works best if the observations include data on a variety of calibrators well distributed around the sky. DELZN applies a correction to a CL table or writes a file to disk with zenith atmospheric delays and possibly clock offsets that can be used by CLCOR, OPCODE='ATMO' to correct a CL table. The second option is for the situation where the data used by DELZN is in a different file from the data that needs to be corrected.

To correct an attached CL table, the typical inputs for DELZN would be:

> TASK 'DELZN' ; INP ⏎ to review the inputs.
> INDISK *n* ; GETN *ctn* ⏎ to specify the input file.

> SNVER *snin* ↵ to select the SN table containing multi-band delays.
> GAINVER *clin* ↵ to select the CL version to which solutions are to be applied.
> APARM(4) 1 ↵ to create a new CL table.
> APARM(5) 1 ↵ to solve for atmosphere and clocks
> SOURCES 'DA913' , ' ' ↵ to specify the sources to be corrected.
> CALSOUR to specify the calibrator sources observed at a large variety of elevations
'0103+337', '0140+412', '0150-334', '0159+418',
'0202+319', '0244-297', '0358+210', '0425+174',
'0641+392' ↵
> OPTYPE 'MDEL' ↵ to use multi-band delay
> DOTV -1 ↵ to make PL files
> GO ↵ to run the program.

This will generate a CL table and several PL files that show the data and the fitted model. You are **strongly encouraged** to examine these.

To create an output file rather than correct a CL table use the same inputs as above except:

> APARM(4) 0 ↵ to create no CL table.
> SOURCES ' ' ↵ to correct no sources
> CALSOUR '*' ↵ to use all calibrator sources.
> OUTFILE 'MYVLB:BZ199.DELZN' ↵ output file name

Again, you are advised to examine the resulting plot files which show both the data and the fitted model. The output file can be read in with CLCOR (OPCODE='ATMO'; INFILE='MYVLB:BZ199.DELZN') to correct a CL table.

There is another task designed to deal with the effects of zenith delay in phase-referencing observations. Phases for the target source in phase referencing are corrected by the phases at the calibrator which usually is at a different elevation. Task DFCOR is a special version of CLCOR which applies the 'ATMO' operation to correct the CL table for the difference in elevation between the target source and adjacent calibration sources without applying the full atmospheric delay correction.

If the SN table contains dispersions as well as multi-band delays, DELZN may also be run to fit a zenith angle model to the dispersions (OPTYPE = 'DISP'). Like the 'MDEL' operation, the zenith and time function found by DELZN can be applied to a CL table or written to a text file for later application by CLCOR with OPCODE = 'DISP'.

9.5.7.6 Finding multi-band delays

For astrometric and geodetic experiments and to use DELZN (see § 9.5.7.5), the multi-band delay must be determined. The multi-band delay is the delay caused by errors in the station positions and the difference between the correlator model and reality for clocks and the troposphere. The multi-band delay is best determined over IFs which are widely spaced in the frequency band. After the instrumental phases have been corrected (§ 9.5.4.3–§ 9.5.4.4), the multi-band delay can be determined in one of two ways. For strong sources do a global fringe fit as described in § 9.5.7.7 setting APARM(5)=0, and then run MBDLY on the resulting SN table.

Typical inputs for MBDLY would be:

> TASK 'MBDLY' ; INP ↵ to review the inputs.
> INDISK *n* ; GETN *ctn* ↵ to specify the input file.
> INVERS SN *table from FRING* to select input SN table

> OUTVERS 0 ↵	make new SN table
> BIF 0; EIF 0 ↵	to select all IFs.
> SUBARRAY 0 ↵	to select all subarrays.
> APARM 0 ↵	the defaults are generally O.K.
> GO ↵	to run the program.

This will produce a new SN table with the MBDELAY columns filled in.

For weak sources, use APARM= 2 0 0 1 1 in FRING. This averages each IF and fits a multi-band delay across them. Note that this *does not* solve for single-band delays, unlike the previous method. This will also produce a new SN table with the MBDELAY columns filled in.

Note that MBDLY can fit the single-band delays for a multi-band delay plus a dispersion (phase varies with wavelength rather than frequency) with OPTYPE = 'DISP'. FRING can do a very similar fit, after finding the single-band delays, if you set APARM(10) = 1. Wide bandwidth observations at low frequencies may benefit from this.

9.5.7.7 Antenna-based fringe-fitting

To see an example of the residual phase errors in your data, use POSSM to view the phase on a short calibrator scan (at some time other than that used to solve for the phase-offsets in § 9.5.4.4). In general, there will be a gradient in phase between the IFs (due to the "multi-band" delay) and also small gradients within each IF (caused by small residual "single-band" delays). These time-variable phase gradients are mainly due to inaccuracies in the geometrical time delays that the correlator assumed for the time of arrival of the wavefront at each antenna. These inaccuracies arise from propagation effects through the troposphere and ionosphere, inaccurate Earth geometry, etc. and give *phase* errors which are proportional to frequency. Such phase errors prevent integration of the data over frequency (or cause a loss of coherence if you do). Similarly, VPLOT will show, on any single IF and spectral channel, phases which change rapidly with *time*. Again, these are due to unavoidable inaccuracies in the correlator model; such large "phase rates" prevent integration over time. Both of these points are illustrated in Figure 9.1.

You will want to run FRING to correct for these residual rates. You can help these tasks by making sure that the reference pixel in frequency is in the center of the band (Nchan/2+1 is best). Use task CENTR or any of the tasks with the FQCENTER adverb to fix data sets for which this is not true.

FRING and KRING use a global fringe-fitting algorithm described by Schwab and Cotton, 1983, *Astron. J.*, **88**, 688. Unfortunately, these are large and complicated tasks. The procedure VLBAFRNG is available to simplify access to FRING and VLBAKRNG access to KRING. Versions of these procedures for phase-referencing experiments are called VLBAFRGP and VLBAKRGP. For all these procedures, if the SOURCES adverb is set, then CLCAL is run once to apply the results of FRING (or KRING) for each source in SOURCES. For the phase-referencing procedures (VLBAFRGP and VLBAKRGP), any source that is in the SOURCES list that is *not* in the CALSOUR list will be phase referenced to the *first* source in the CALSOUR list. Note that, if every source in the SOURCES list occurs in the CALSOUR list, VLBAFRNG and VLBAKRNG will run identically to VLBAFRGP and VLBAKRGP, respectively. If the SOURCES list is empty, VLBAFRNG and VLBAKRNG will run CLCAL once over all sources, while VLBAFRGP and VLBAKRGP will run CLCAL once referencing all the sources to the first source in CALSOUR. These procedures will produce new (highest numbered) SN and CL tables.

Sample inputs for procedure VLBAKRNG are:

> RUN VLBAUTIL ↵	to acquire the procedures; this should be done only once since they will be remembered.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> TIMERANG 0	to include all times.

> BCHAN 0 ; ECHAN 0 C_R	to use all frequency channels.
> GAINUSE *CLin* C_R	to use the CL table with all the calibration up to this point.
> REFANT *n* C_R	to specify an antenna that is present most of the time as the reference antenna.
> SUBARRAY 0 C_R	to use all subarrays.
> SEARCH 0 C_R	to try all antennas as a reference antenna if fringes cannot be found using REFANT. *This is different from* FRING*; in* FRING *this must be set to try other reference antennas*.
> OPCODE ' ' C_R	to leave all solutions in the output SN table.
> CPARM 0 C_R	to use defaults for KRING steering parameters; this is okay for strong sources.
> CPARM(1) *x* C_R	to specify the minimum integration time in seconds.
> CPARM(8) 1 C_R	to avoid re-referencing solutions; do this *only* for polarization experiments.
> CALSOUR '*src1*', '*src2*' C_R	to specify the sources to fringe fit using KRING.
> SOURCES '*src1*', '*src2*' C_R	to have CLCAL run for each source using the interpolation method given below.
> INTERPOL 'AMBG' C_R	to use the "AMBG" interpolation method (linear phase connection using rates to resolve phase ambiguities).
> BADDISK 0 C_R	to use all disks for scratch files.
> VLBAKRNG C_R	to run the procedure.

Procedure VLBAKRGP sets the same adverbs as VLBAKRNG *except*

> SOURCES '*src1*', '*src2*', '*src3*' C_R	to have CLCAL run for each source using the interpolation method given by INTERPOL. Any source here that is not in the CALSOUR list will be phase referenced to the first source in the CALSOUR list. In this example, *src3* is phase referenced to *src1*.
> VLBAKRGP C_R	to run the procedure.

VLBAFRNG and VLBAFRGP are identical except there is no OPCODE (it is equivalent to DPARM(8)) and DPARM(4) and DPARM(7) in FRING are the same as CPARM(1) and CPARM(8) in KRING, respectively. Also note the different use of SEARCH in FRING and KRING.

Suitable inputs for the fringe-fitting task FRING are, in detail:

> TASK 'FRING' ; INP C_R	to review the inputs.
> INDISK *n* ; GETN *ctn* C_R	to specify the input file.
> CALSOUR ' ' C_R	to find solutions for all sources.
> TIMER 0 C_R	to find solutions for all times.
> DOCALIB 1 C_R	to apply the most complete calibration file including amplitude calibration and IF and channel phase offsets to both the visibilities and the weights.
> GAINUSE 3 C_R	to use CL table 3.
> SNVER 2 C_R	to write solutions into SN table 2.
> BCHAN 0; ECHAN 0 ; CHINC 1 C_R	to use all spectral channels within each IF channel.
> FLAGVER 0 C_R	to apply the most recent flag table.
> SMODEL 0 ; CLR2NAME C_R	to use a point-source at the origin model for the sources, rather than a Clean-component model.

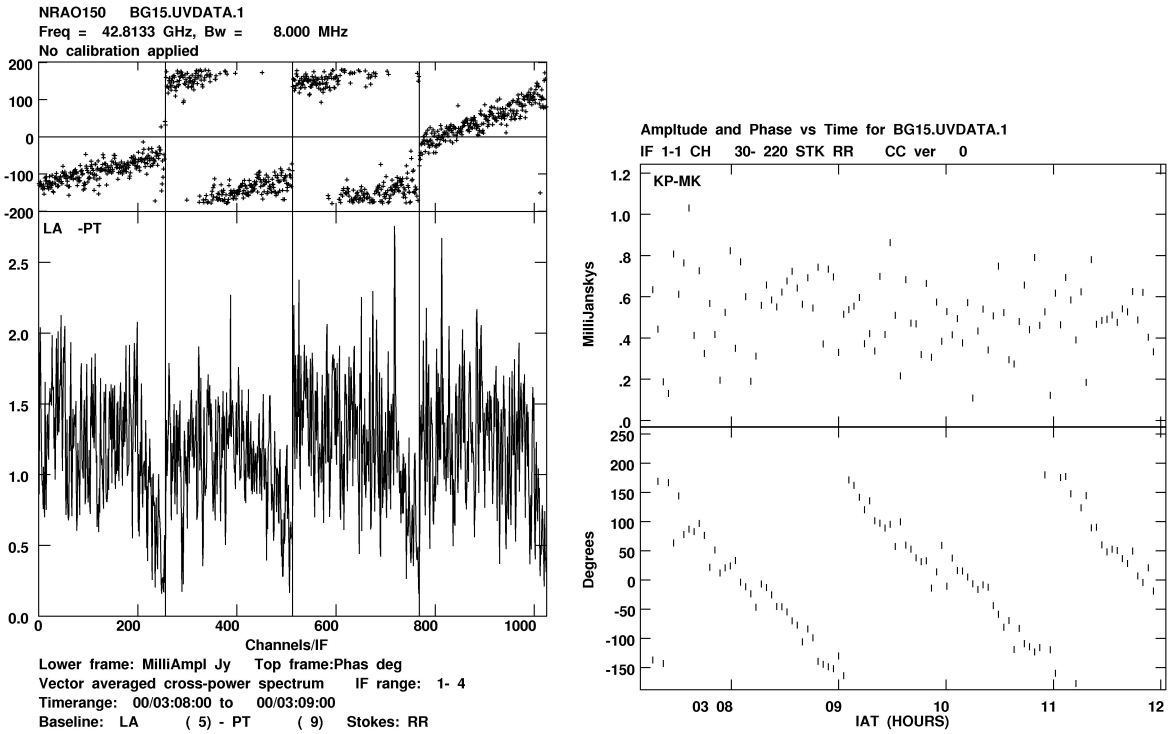


Figure 9.1: *left:* A POSSM plot of the 43-GHz spectrum of the quasar NRAO150 on the Los Alamos to Kitt Peak baseline. The plot shows that the observation was performed with 4 IFs, with 256 spectral channels within each IF. The upper frame shows the phase variation with frequency; within each IF, the small phase slope is caused by a residual delay error. The phase offsets between the IFs can be clearly seen as well. Both the residual delay error and the phase offsets must be determined and removed before the data can be spectrally averaged (see § 9.5.7). *right:* A VPLOT plot of the uncalibrated amplitude (upper frame) and phase (lower frame) as a function of time for the source NRAO150 at 43 GHz on the baseline Kitt Peak to Mauna Kea. Note how the phase varies as a function of time; this variation is equivalent to a residual fringe rate of 8.3 mHz. Unless the fringe rate is determined and removed (see § 9.5.7), the data cannot be averaged in time.

> SOLINT 3 $^{C}_{R}$	to set the solution interval in minutes; do not exceed the atmospheric coherence time (see below). Setting SOLINT to 0 sets solution intervals equal to scan lengths.
> SEARCH *ref2*, *ref3*, ... $^{C}_{R}$	to specify the order over which antennas are searched for fringes when the exhaustive search is requested. If APARM(9) is set, all antennas will be searched for fringes with SEARCH controlling the order of the search. Note, REFANT should be SEARCH(1). The use of APARM(9) and SEARCH makes FRING much more likely to find fringes to weak antennas and is highly recommended.
> APARM(1) 2 $^{C}_{R}$	to accept solutions when only 2 antennas are present; default is 6.
> APARM(2) 0 $^{C}_{R}$	to have the data divided by the model before fitting fringes; APARM(2) > 0 tells FRING that the data have already been divided by a model.
> APARM(3) 0 $^{C}_{R}$	to treat polarizations separately; APARM(3) > 0 averages RR and LL.

> APARM(4) 0 C_R — to use the frequencies individually within each IF; APARM(4) > 0 causes the frequencies within each IF to be averaged before the solution.

> APARM(6) 1 C_R — to get some useful, but limited, messages, including the SNR.

> APARM(7) 9 C_R — to avoid false detections by setting a moderately high minimum for the SNR accepted. You may wish to use a lower threshold (especially for SVLBI) although APARM(7) less than about 3 is probably not useful.

> APARM(8) 0 C_R — to set the maximum number of antennas (if no AN table).

> APARM(9) 1 C_R — to enable the exhaustive search mode.

> APARM(5) 0 C_R — to do least-squares fits in each IF. APARM(5) ≤ 0 means to solve separately for the rate, single band delay and phase of each IF. $1.5 >$ APARM(5) > 0 means to solve for one single rate and multi-band delay affecting all IFs. If $2.5 >$ APARM(5) > 1.5, the task additionally solves for the difference between the multi-band delay and single-band delay, *i.e.*, it allows for a different gradient of phase versus frequency within an IF than between IFs. Note, however, that unlike APARM(5)=0, this option assumes that the single-band delay is the same in each IF; it therefore solves for a single value for the difference between multi-band and single-band delay affecting all IFs. Normally users should use APARM(5)=0 for multi-IF data. Primarily for the VLA, a fourth option, APARM(5) = M has been added, with $M > 2$. It divides the IFs into $M - 1$ groups and solves for one delay per group.

> REFANT 5 C_R — to choose an antenna that will give fringes for most of the scans. This is important: FRING will search for fringes to this antenna first. If it fails for some reason, it will select another reference antenna, based on the ANTWT data, and, if it still fails, give up (see however the discussion of the new SEARCH adverb above). In this case, you should look for scans with no fringes or a bad reference antenna may be causing the problem. A big, sensitive antenna is often used as REFANT (*e.g.*, Effelsberg). Occasionally, it may be helpful to split your data set up into 2 or 3 sections, which are fringe-fitted with different REFANT (*e.g.*, a "European" and a "US" part of the observations). Changes in reference antenna should, in general, not cause problems.

> ANTWT 0 C_R — to apply no additional weights to the antennas before doing the solutions. If the amplitude calibration was incorrect, you can use this option to force antenna weights up or down to control the weight FRING gives to data to each station when making the global solutions. Unless the SEARCH option described above is chosen, ANTWT also controls the order in which antennas are tried as secondary reference antennas after failing to find fringes on the REFANT. Give higher weight to antennas you want to see used as secondary references.

> ANTENNAS 0; DOFIT 0 C_R

to allow all antennas to be fit. Only baselines between antennas listed in the ANTENNAS adverb are used in the fringe search. If any antennas are specified in DOFIT however, a solution is made only for those antennas; all other selected antennas are assumed to be already calibrated and are passed through with no additional corrections. See the HELP file for FRING under DOFIT. DOFIT **is only active if** APARM(9) **is set.**

> DPARM(1) 1 C_R

to use one baseline combination in the initial coarse (FFT) fringe search. This provides a starting guess for the least-squares solution. If you are searching for weak fringes you should consider using two and three baseline combinations in the search; this can improve sensitivity in the initial fringe search. See Lecture 19 in *Synthesis Imaging in Radio Astronomy*, edited by R. Perley, F. Schwab, and A. Bridle, for an explanation of how this global multi-baseline searching works. Note that if your source structure is complex and you have not divided the data by an accurate source model, then setting this parameter to one is safest. DPARM(1)>1 works even when the integration times are not equal.

> DPARM(2) 0 C_R

to set the full width of the delay window in nsec, centered around 0, to search; the default, chosen here, is to use the full Nyquist range defined by the frequency spacing. A smaller search window can permit a lower SNR threshold to be set, but can also result in lost data due to failed fringe searches. For the VLBA, a DPARM(2) = 1000 window is usually adequate.

> DPARM(3) 0 C_R

to set the full width of the fringe-rate window in mHz, centered around 0; the default, chosen here, is to use the full Nyquist range defined by the integration time. A smaller search window can permit a lower SNR threshold to be set, but can also result in lost data due to failed fringe searches. For the VLBA, a DPARM(3) = 200 window is usually adequate.

> DPARM(4) 2 C_R

to specify the correlator integration time in seconds; use GETITIME or DTSUM to find the correct value. For data from the VLBA correlator, DPARM(4)=0 will cause FRING to determine the correct integration time by examining the data file directly.

> DPARM(5) 0 C_R

to do both the coarse and the least squares solutions; set to 1 if you require only FFT solutions.

> DPARM(6) 1 C_R

to keep, for single-source files, frequencies separated in the output file; the default is to average frequencies within IFs. This parameter does not affect multi-source files.

> DPARM(7) 0 C_R

to re-reference solutions to a common reference antenna; when processing polarization data, set this to 1 to avoid the re-referencing.

> DPARM(8) 0 C_R

to disable the zero'ing options — see the HELP file. *WARNING,* DPARM(8) > 0 *will discard parts of the final solution. Be sure to use this option with extreme care.*

> DPARM(9) 0 C_R

to allow fitting for rate. DPARM(9)> 0, causes the task to suppress rate fitting entirely rather than zeroing it after the fact as in DPARM(8).

> INP ↵ to check the inputs.
> GO ↵ to do the fit — finally.

Note that FRING finds solutions in two steps. First approximate solutions are found in the FFT step using combinations of one, two, or three baselines (see DPARM(1) above). Then, as long as DPARM(5) < 1, a least-squares algorithm uses these approximate values as a starting point for refining the solutions. Especially for weak sources, the least-square solution may wander outside narrow constraints set by DPARM(2) and DPARM(3).

Note that the SOLINT interval should be selected with consideration of the atmospheric coherence time, but must be long enough that high signal-to-noise-ratio solutions are achieved. For observations between 1.6 and 15 GHz, solution intervals of 3–6 minutes (and often longer) should be fine. At other frequencies shorter solution intervals may be required. In these cases, experiment with different length solution intervals on short sections of data. Note that solution intervals greater than the scan length will never be used; the scan lengths are listed in the NX table and may be examined using PRTAB or LISTR with OPTYPE='SCAN', or LISUN.

If the source is complex, and especially if the visibility phase of the source changes during SOLINT, it is useful to divide the data by a Clean model derived from previous observations or from an earlier attempt at processing the data. This Clean model can be specified by filling in IN2NAME *et al*. Situations where this is useful include observations of equal doubles (where there are zeros in the amplitude and, hence, rapidly changing phases) or very large sources (of order arcseconds). If you are using multi-baseline searching (*i.e.*, DPARM(1) > 1), then solutions may be more sensitive to source structure and an input model may be useful if the structure phases are larger than one radian on many baselines. When using a model, convergence may be improved by weighting the data by $1/\sigma$ rather than $1/\sigma^2$; set WEIGHTIT = 1.

You should check the SNRs found by FRING carefully; they are printed if APARM(6) > 0. The SNRs estimated during the FFT search are used to determine if the SNR of a solution is $\geq$ the threshold set in APARM(7). If they are not, then that solution is flagged before being passed to the more accurate least-squares routine. Users should check that the SNRs found in the LSQ routine match those expected. If the detected SNRs are too low, SOLINT may be too long or too short or other parameters may be set wrongly.

Be warned that proper scaling of the SNRs by FRING depends upon whether or not the data weights have been properly calibrated. Task FIXWT may be used to calibrate the weights, but changing the SNR threshold for FRING directly (APARM(7)) usually produces satisfactory results.

The final delay and rate solutions and their SNRs should be inspected using LISTR:

> TASK 'LISTR' ; INP ↵ to review the inputs.
> INDISK *n* ; GETN *ctn* ↵ to specify the input file.
> OPTYPE 'GAIN' ↵ to list gain solutions.
> INEXT 'SN' ; INVER 2 ↵ to list SN table 2 (as above).
> DPARM 6, 0 ↵ to list delay; use 7 for rate, 1 for phase, and 8 for SNR.
> GO ↵ to run the program.

Alternatively, the solutions can be plotted against time to make sure they are sensible. Use SNPLT:

> TASK 'SNPLT' ; INP ↵ to review the inputs.
> INDISK *n* ; GETN *ctn* ↵ to specify the input file.
> OPTYPE 'RATE' ; DOBLANK 1 ↵ to plot rate solutions including failed ones, OPTYPE 'DELA' for delay solutions.
> OPCODE '' ↵ to plot each polarization and IF in separate plots.
> NPLOTS 5 ↵ to plot five antennas/IFs/polarizations per page.
> INEXT 'SN' ; INVER 2 ↵ to plot SN table 2 (as above).
> GO ↵ to run the program.

It is a good idea to plot the solutions for OPTYP 'RATE' and 'DELA' (single-band delay) as well as the

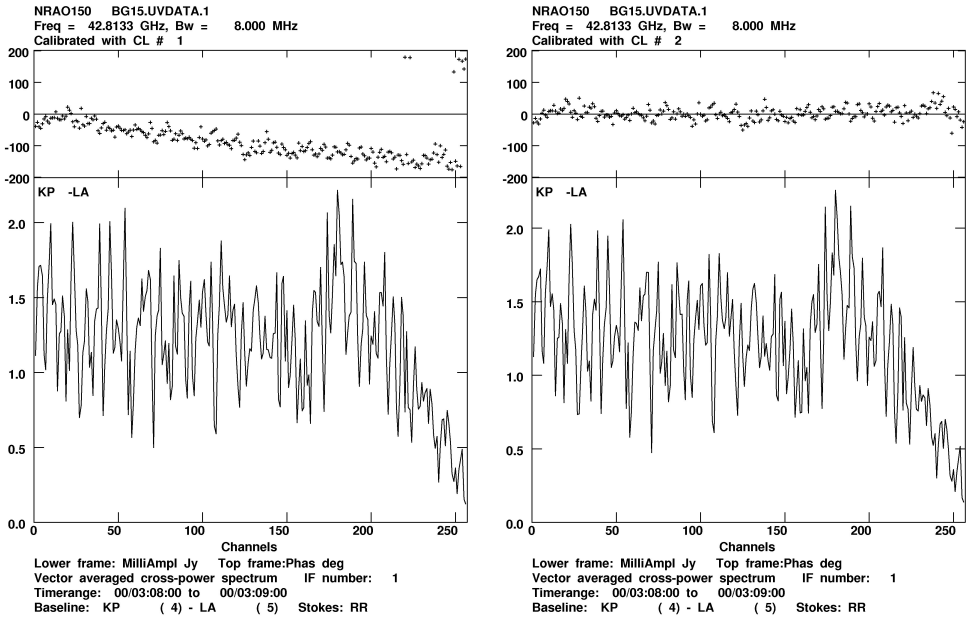


Figure 9.2: *left:* A POSSM plot of the uncalibrated spectrum of NRAO150 at 43 GHz on the baseline Kitt Peak to Los Alamos. The plot shows the spectrum for a single IF to show the effects of the residual delay error more clearly. The phase slope as a function of frequency is clear evidence for a small delay error in the correlator model. *right:* The same data as shown on the left, but corrected for a delay error of -55 nanosec and a residual fringe-rate of -2.0 milliHz. Note how the phase as a function of frequency is now flat and centered around zero degrees. These data can now be averaged in frequency, if desired.

associated 'SNR's. They should be smoothly varying functions. Delays and rates should be found only within your specified windows. Check for suspicious detections at the limits of the search windows — for instance, they could be detections of side lobes of the main fringe. If you used windows smaller than Nyquist, you may want to check your detections using bigger windows. Gaps in detections with time can occur if, *e.g.*, tapes were bad, antennas were off source, the source visibility is in a minimum, or the reference antenna choice was bad. It pays to investigate such problems at this point before proceeding.

Discrepant SN solutions can be removed using the interactive task SNEDT or the non-interactive task SNSMO. If, in the latter, the CPARM values are set, then the SN solutions will be clipped if they differ from the running mean by amounts which you can specify. If the BPARM values are set, the solutions are then smoothed and clipped entries can be replaced with mean values based on a boxcar- or median-window-filter average. See the explain file for details. Although SNSMO allows the option of modifying the input SN table, it is safest to have it create a new one. If you have a lot of discrepant SN values, you should also consider using option INTERPOL = 'POLY' in task CLCAL (see below).

If there isn't enough disk space to run FRING on all the data at once (because of the large scratch file that FRING insists on creating), you can run FRING multiple times specifying time ranges and explicitly setting SNVER to the same SN table. Judicious use of CHINC when your spectral channels are narrow (< 0.5 MHz) may also help with disk space and compute time.

If, for some reason, you set the parameters of FRING or SNSMO wrongly and the resulting SN table is unusable, it is wise to avoid confusion by deleting it using EXTDEST and starting over again.

Once a valid SN table has been produced, the next step is to interpolate the solutions found onto the finer grid of entries in a CL table using the task CLCAL with INTERPOL = 'AMBG' as described in § 9.5.1.1.

Once a final CL table is generated, its effect on the data can be viewed using tasks VPLOT and POSSM by setting DOCAL=1 and GAINUSE to the version number of the final CL table; see § 9.4. Optionally, in VPLOT, one can average over spectral channels and/or IF channels before plotting. Use VPLOT to plot a time range covering a few FRING solution intervals on a strong source. Phase variations should be small with no jumps. If this is not the case, check the inputs to FRING (especially SOLINT) and CLCAL. A comparison of before and after phases is shown in Figure 9.2.

9.5.7.8 Baseline-based fringe-fitting

Baseline-based fringe-fitting, implemented in BLING and BLAPP, is an alternative to using FRING and CLCAL described above. Whereas FRING searches and solves for station rates and delays globally, BLING makes independent fits to each baseline for delays and rates, creating a BS table of baseline-based solutions.

In most cases, the global fringe-fitting described in § 9.5.7.7 should be used since FRING should be able to fringe-fit weaker sources more reliably. However, there are some instances in which baseline-based fringe-fitting is to be preferred. Note that FRING may be used to do baseline-based fringe-fitting by running it many times, each time specifying only 2 antennas. BLING has not been actively maintained or used, and so may be less reliable. Amongst the advantages of the baseline-based fringe-fitting are:

1. BLING may be more robust than FRING even in the absence of an accurate source structure model.
2. Fringe solutions can be found for cross-polarized fringes without editing the *uv* header.
3. As presently implemented, for a given number of IF and spectral channels, BLING can solve for longer scans than can FRING.
4. BLING has the option of adjustable, non-zero centered fringe-search windows, which can be controlled from an external file. This option may be important in fringe-fitting Space VLBI data.

BLING is distinguished from FRING by the ability to directly control the fringe search on each separate baseline, and by the ability to solve for fringe acceleration if required. Separate fringe windows in delay, rate and acceleration, and different solution intervals can be set for each individual baseline. The fringe windows may have non-zero offsets and can be specified using the input adverbs or, more flexibly, by drawing up an external ASCII control file of fringe-prediction windows and BLING control parameters. The latter option allows the specification of time-variable fringe-search parameters across the observing file. The general algorithm follows that described by Alef and Porcas, 1986, (*Astron. Astrophys.*, **168**, 365); BLING also allows the stacking of data from different baselines, as discussed by Schwab and Cotton, 1983, *Astron. J.*, **88**, 688. In addition, model division is possible before the fringe-fitting is performed and cross-polarized fringe searches can also be conducted without editing the *uv* header. BLING writes the results to a BS table. These baseline-based solutions can be converted into antenna-based corrections using the separate task BLAPP and then applied to the data.

Acceleration may be solved for by conducting a coarse search in the specified acceleration window. The results of this search are then interpolated to estimate the final solution. Fringe acceleration searches can considerably increase the amount of CPU time it takes to run BLING. Therefore, you may wish to turn off the acceleration search (DPARM(7) to DPARM(9)) unless you need it for space VLBI.

Full details concerning BLING input parameters can be found by typing EXPLAIN BLING. This includes information concerning the format required for the external control file. Earlier versions of BLING are discussed in *AIPS* Memo 89 (1994, "Baseline-Oriented Fringe Searches in *AIPS*" by Chris Flatters). Typical input parameters to BLING are given below:

> TASK 'BLING' ; INP C_R to review the inputs.

> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> CALSOUR 'DA193', ' ' ↵	to specify the calibrator source.
> STOKES 'LL' ↵	to select the Stokes.
> TIMERANG 0, 10, 5, 0, 0, 11, 0, 0 ↵	to limit the time range.
> ANTENNAS 3 ; BASELINE 0 ↵	to select all baselines to antenna 3.
> SUBARRAY 1 ↵	to use subarray 1.
> FREQID 1 ; BIF 1 ; EIF 0	to use frequency ID 1 with all IFs.
> BCHAN 1 ; ECHAN 0 ↵	to use all spectral channels.
> DOCAL 1 ; GAINUSE *clin* ↵	to apply amplitude calibration before fringe fitting.
> CLR2N ; NMAPS 0 ↵	to do no model division.
> SOLINT 0.5 ↵	to use a 30-second fringe solution interval.
> INFILE ' ' ↵	to use the adverbs rather than an external control file.
> APARM(1) 2 ; APARM(2) 0 ↵	to set the integration time; no model division.
> APARM(3) 0 ↵	to do no stacking of baselines.
> APARM(4) 0 ; APARM(5) 0 ↵	to set minimum acceptable SNR to 5 and accept coherence of 20%
> DPARM 0 ↵	to use the default fringe windows and no acceleration search.
> DOUVCOMP 1 ↵	to use compressed scratch files.
> BADDISK 0	to use all disks for scratch files.
> GO ↵	to run the program.

Note that baseline-stacking (APARM(3)) is not implemented for data sets with unequal integration times. Also, note that the fringe-rejection criteria specified using APARM(4) and APARM(5) are important parameters. The use of compressed scratch files is recommended and is not believed to have a significant impact on precision. Note, if model division is required, APARM(2) should be set and the source should be entered explicitly.

BLING will execute with a summary line marking the start of each baseline processed. The resulting BS table can be examined using task BSPRT, with input parameters:

> TASK 'BSPRT' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> INVERS *bsin* ↵	to specify the BS table version number.
> DOCRT -1 ↵	to send output to an external file.
> OUTPRINT 'FITS:BSPRT.LIS' ↵	to define the output file name.
> GO ↵	to run the program.

For printing to the screen, select DOCRT=1.

The BS table output includes the estimated fringe parameters and their associated errors. Note that due to changes in FFT interpolation the errors may be an overestimate. The BLING solutions are interpolated and factorized into antenna-based gain solutions using BLAPP. This task either writes a solution (SN) table which can be applied using CLCAL, or allows a CL table to be updated with the calibration information directly. These options are selected using OPCODE='SOLV' or OPCODE='CAL' respectively. BLAPP can interpolate solutions with unequal time sampling and includes the acceleration term in interpolation if it is available. Typical inputs to BLAPP are:

> TASK 'BLAPP' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> INVERS *bsin* ↵	to specify the input BS table version number.
> SOURCES ' ' ; STOKES 'LL' ↵	to do all sources and a specific Stokes.
> FREQID 1 ↵	to select frequency ID 1.

> TIMERANG 0 ; ANTENNAS 0 ↵	to do all times and antennas.
> SUBARRAY 1 ; REFANT 3 ↵	to do subarray 1 with reference antenna 3.
> ANTWT 0 ↵	to use equal antenna weights.
> OPCODE 'SOLV' ↵	to solve for a SN table.
> GAINVER clin ↵	to specify the input CL table which defines the times for which solutions are desired.
> BADDISK 0 ↵	to use all disks for scratch files.
> GO ↵	to run the program.

The resulting solution (SN) table can be plotted using SNPLT in the standard fashion. For OPCODE='CAL' the output CL table also needs to be specified using GAINUSE. If an SN table is generated, it can be smoothed or clipped using task SNSMO and applied using CLCAL as described in § 9.5.1.1.

9.5.7.9 SVLBI-specific techniques

An alternative approach to direct fringe detection of each individual baseline to the orbiting antenna is to first calibrate the ground array using conventional fringe-fitting techniques, then coherently combine all ground antennas to improve the fringe detection sensitivity to the spacecraft. Several incarnations of this approach exist within *AIPS*. The *AIPS* tasks FRING, BLING, and KRING all allow baseline stacking which can be used to fringe fit the space baseline using composite baselines. It was shown in VLBA Scientific Memo No. 13 (1996, "Global ground VLBI network as a tied array for space VLBI", by L. Kogan) that the method of phasing a group of ground-based antennas and the method using global fringe fitting with baseline stacking give the same minimum detectable flux density. Therefore, baseline stacking with DPARM(1)=3 in FRING should yield the best possible sensitivity. There are other options which also may be explored. The adverb DOFIT in FRING and KRING can be used to solve for subsets of the available antennas in order to find good solutions for the ground antennas in a dual round of fringe-fitting. The exhaustive baseline search mode, used by default in BLING and KRING and activated in FRING by setting APARM(9)=1, allows more baselines to the spacecraft to be searched.

9.5.7.10 Spectral-line fringe-fitting

The determination of the delay and fringe-rate calibration is a two- or three-step process for spectral-line VLBI data. First, the residual delay and fringe-rates are estimated for each antenna from the continuum calibrators. Then, residual fringe-rates must be determined again for the line source using a "strong" channel or range of channels. As an intermediate step, the phases of the line source should be examined to check that the calibrator's residual fringe-rates haven't destroyed phase coherence; if so, then the calibrator's residual fringe-rates should not be applied to the line source.

If computer memory is limited, FRING must trade off the number of spectral channels against the length of the solution interval. For continuum calibrators in a spectral-line dataset, the large number of spectral channels may force FRING to require too short a solution interval. Judicious use of CHINC in FRING reduces this problem for spectral channels < 0.5 MHz in width. (FRING now allocates memory dynamically and may be able to handle large cases so long as the computer is adequately equipped with real and swap memory.) This limit can be overcome by running UVCOP to extract the continuum calibrators into a separate data file and then running AVSPC with AVOPTION 'SUBS' to average spectral channels coherently within each IF. INDXR should be run to regenerate an NX table. FRING will then allow more reasonable solution intervals.

Run FRING as follows only on the continuum calibrators to determine residual delays and fringe-rates:

> TASK 'FRING' ; INP ↵	to review the inputs.

> INDISK *n* ; GETN *ctn* C_R	to specify the input file.
> CALSOUR 'BLLAC','DA193' C_R	to select continuum calibrator sources.
> DOCALIB 1 C_R	to apply the current calibration.
> GAINUSE *clin*	to specify which CL table to use.
> SMODEL 0 C_R	to use the null source model (points at the origin).
> FLAGVER 0 C_R	to apply the most recent flag table.
> BCHAN 10 ; ECHAN 115 C_R	to exclude the edges of the band; normally the data in these channels are corrupted by the bandpass filters.
> REFANT 5 C_R	to select a reference antenna (see continuum discussion).
> SOLINT 6 C_R	to set the solution interval in minutes. It should not exceed the coherence time.
> APARM(6) 1 C_R	to get some useful, but limited, printout; gives SNR.
> APARM(7) 7 C_R	to avoid false detections by setting the minimum acceptable SNR. *Warning:* solutions with lower SNR will be flagged as bad which will ultimately flag the affected data.
> DPARM(1) 1 C_R	to use one-baseline combination in initial, coarse fringe search (FFT). This provides starting points for the least-squares solutions.
> DPARM(2) 10000 C_R	to select a delay window in nsec, centered around 0(!). The default is to use the Nyquist range. For a 250 kHz-bandwidth observation, setting this value to 10000 nsec is equivalent to setting the search window to 5 delay channels, which is usually sufficient.
> DPARM(3) 200 C_R	to select a fringe-rate window in mHz.
> DPARM(4)= 1 C_R	to tell FRING the correlator integration time.
> DPARM(5) 0 C_R	to do the least-squares solution.
> SNVER 0 C_R	to write solutions in a new SN table.
> GO C_R	to do the fit.

If the calibrators had been extracted to a separate data set, use TACOP to copy the resultant SN table back to the line data set. SNCOP may be used when the number of IFs in the two data sets is different. This is more likely to arise when you use a strong line signal in one IF to solve for the delays of a larger data set.

Run CLCAL to apply the delay and fringe-rate solutions to all sources, as is described in § 9.5.1.1, and then carefully examine the phase coherence of the line source in a suitable line channel (or group of channels) using POSSM or COHER before and after applying the new CL table. It may be that the fringe-rate solutions have made the phase coherence worse. In this case, you must run SNCOR using the 'ZRAT' option to zero the fringe-rates and then re-run CLCAL.

After this point, the calibrator data is usually of little or no interest. But, if you do plan to use the calibrator data further, remember to be careful to juggle the SN and CL tables correctly. Even if you decide not to apply the calibrator fringe-rates to the line-source, they are still applicable for the calibrator itself. The CL table created using the un'ZRAT'-ed SN table contains the proper corrections for the calibrator data while the CL table created using the 'ZRAT'-ed SN table contains the proper corrections for the line source.

Now re-run FRING to determine the residual fringe-rates for the line source, this time selecting a suitable line channel or group of channels:

> TASK 'FRING' ; INP C_R	to review the inputs.
> INDISK *n* ; GETN *ctn* C_R	to specify the input file.
> CALSOUR 'OH127.8' , ' ' C_R	to select the spectral-line source.
> DOCALIB 1 C_R	to apply the previous amplitude and delay/rate calibration.

> GAINUSE *clin*	to specify which CL table to use.
> FLAGVER 0 C_R	to apply the most recent flag table.
> SNVER 0 C_R	to write a new solution table.
> BCHAN 72 ; ECHAN 72 C_R	to select the strongest and/or simplest spectral channel as a reference.
> REFANT 5 C_R	to try to use the same reference antenna as in the previous run of FRING.
> SOLINT 6 C_R	to set the solution interval in minutes. It should not exceed the coherence time.
> APARM(6) 1 C_R	to get useful, but limited printout.
> APARM(7) 9 C_R	to avoid false detections by setting the minimum acceptable SNR. *Warning:* solutions with lower SNR will be flagged as bad which will ultimately flag the affected data.
> DPARM(1) 1 C_R	to use one-baseline combination in initial, coarse fringe search (FFT).
> DPARM(2) = -1 C_R	to prohibit a search in the delay domain by setting the delay window to a negative number. Remember setting this to 0 means to use the Nyquist value, which is not what we want.
> DPARM(3) 0 C_R	to search for fringes over the full Nyquist fringe-rate window since we don't know where the fringes are.
> DPARM(4)= 0 C_R	to let FRING automatically determine the correlator integration time (if this fails, you will have to set the proper value here — see DTSUM).
> DPARM(5) 0 C_R	to do the least-squares solution.
> GO C_R	to do the fit.

Then run CLCAL again to apply these solutions to the previous calibration tables. You have then generated a full set of calibration tables and data. Remember that the calibration information for the continuum and line sources are stored in different CL tables.

9.5.7.11 Polarization-specific fringe-fitting

The phase and delay corrections obtained using RR and LL data can only remove R-R and L-L offsets between different antennas. There still may be R-L phase and multi- and single-band delay offsets. The R-L delay corrections can be determined using the RL and LR data and a task named RLDLY is available for determining these effects. This still leaves an overall R-L phase offset.

Once the cross-hand calibration has been completed, the instrumental polarization (otherwise known as the feed D-terms), can be determined. Several different methods are available for this purpose, implemented in the tasks PCAL and LPCAL. SPCAL was designed for polarization calibration of spectral-line data sets, but PCAL is now more capable of doing a channel-dependent polarization solution. PCAL is discussed briefly below and further details for each of these tasks can be found in the appropriate EXPLAIN files.

The determination of the absolute polarization position angle is equivalent to the determination of the absolute R-L phase difference. For an unresolved source this can be measured in much the same way as for VLA data (see § 4.3.16) using task RLDIF to determine the cross-polarized phase on the polarization calibrator. Alternatively, a source with known polarization properties can be used, *e.g.*, 3C286 or 3C279. For a resolved polarization calibrator a sum of Clean components is required to compare to the integrated polarization position angle as measured by the VLA or single-dish observations nearby in time. Once the R-L phase correction is known it is applied using task CLCOR with OPCODE='POLR' or task RLCOR (see

§4.3.16). See also VLBA Scientific Memo No. 26, "Polarization Angle Calibration Using the VLA Monitoring Program," by G. Taylor and S. Myers, October, 2000.

After feed calibration and the determination of the absolute polarization position angle, the final Stokes Q and U images can be formed directly. Note that task PCNTR, which is used for displaying polarization images, allows an arbitrary rotation of all polarization vectors under control of input adverb ROTATE. This is only necessary if CLCOR wasn't used to correct the R-L phase; also the polarization angle specified for PCNTR is half the R-L phase difference.

9.5.7.12 R-L delay calibration

RLDLY is a task which replaces RUN file procedures VLBACPOL and CROSSPOL. It determines R-L delay differences and produces an SN table which corrects for these effects (after CLCAL). It is best run on a set of calibrator data for which the source is at least moderately polarized (source polarization dominates instrumental polarization). Several baselines should be averaged but RL or LR fringes (or both) must be detectable on each baseline to the reference antenna. This task should leave a single R-L phase difference that must be determined from a calibrator of known polarization angle.

> DEFAULT RLDLY ; INP C$_R$	to set the task name and examine the inputs.
> INDISK *n* ; GETN *ctn* C$_R$	to specify the input file.
> OUTDI 1 C$_R$	to use disk 1 for temporary files.
> FLAGVER 0 C$_R$	to use the highest numbered flag table.
> GAINUSE *CLin* C$_R$	to use the CL table with all calibration up to this point; *no default*.
> SUBARRAY 0 C$_R$	to do all subarrays.
> ANTENNAS 0 C$_R$	to use all antennas.
> REFANT $A_{\rm ref}$ C$_R$	to select the reference antenna; any antenna may be used but all baselines to it should have RL and LR fringes. REFANT 0 will loop over all possible (not necessarily good) reference antennas, averaging the result.
> APARM 6, 1, 10 C$_R$	to use only solutions with signal-to-noise ratio above 6, to not write a CL table even if there is only one calibration scan, and to omit all solutions with rms < 10 from the average over antennas.
> CALSOUR '*cal1*' , ' ' C$_R$	to specify the calibrator source to use.
> TIMERANG *d1 h1 m1 s1 d2 h2 m2 s2* C$_R$	to specify a time range with high SNR for RL and LR.
> SOLINT 0 C$_R$	to specify the minimum integration time in seconds; 0 causes it to be found from the data.
> GO C$_R$	to run the task.

RLDLY should be done after parallel-hand instrumental delays are removed (VLBAPCOR). It may be done before or, with a slight preference, after fringe fitting (VLBAFRNG, VLBAKRNG, VLBAFRGP, or VLBAKRGP). The corrections should be checked with VLBACRPL, by setting STOKES to 'RL' and/or 'LR'. The RL and LR phases should be continuous across the bandpass on each baseline and be flat if the RR and LL phases are flat (no residual delays).

9.5.7.13 Feed D-term calibration

The feed D-terms, or instrumental polarization terms, can be determined using PCAL. SOLTYPEs 'ORI-' and 'RAPR' are appropriate for VLBI data. 'ORI-' uses a non-linear orientation-ellipticity feed model and is appropriate if instrumental polarization exceeds a few percent (*e.g.*, EVN, or VLBA at 18 or 13 cm). 'RAPR' uses a linearized "D-term" model — this is faster but less accurate.

Typical inputs for PCAL would be:

> TASK 'PCAL' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> CALSOUR 'DA193' , ' ' ↵	to select the source.
> TIMERANG 0 ↵	to use all times.
> SELBAND 0; SELFREQ 0; FREQID -1 ↵	to select all frequencies.
> BIF 0; EIF 0 ↵	to select all IFs.
> ANTENNAS 0; UVRANGE 0 ↵	to select all antennas and baselines.
> SUBARRAY 0 ↵	to select all subarrays.
> FLAGVER 0 ↵	to apply the most recent flag table.
> DOCALIB 1 ↵	to apply the previous amplitude and delay/rate calibration.
> GAINUSE *clin*	to specify which CL table to use.
> IN2DISK *n*; GET2N *ctn* ↵	to specify Clean images as models for the I, Q, and U polarizations.
> INVERS 1 ↵	to select a CC table version.
> REFANT 1 ↵	to use the same reference antenna as in the previous run of CALIB.
> NCOMP 77; NMAPS 1 ↵	to specify the number of Clean components to use and to explicitly set the number of Clean images supplied.
> SOLINT 6 ↵	to set the solution interval in minutes. It should not exceed the coherence time.
> SOLTYPE 'RAPR' ↵	to specify the type of feed model.
> PRTLEV 0 ↵	to print minimal information
> BPARM(1) 0 ↵	to use the initial feed model if found in the AN table.
> BPARM(3) 0 ↵	to not fit for R-L phase difference.
> BPARM(4) 0 ↵	to not specify the initial R-L phase.
> BPARM(5) 0 ↵	to not solve for Vpol.
> BPARM(6) 0; BPARM(7) 0 ↵	to solve for the orientations of both polarizations of the reference antenna.
> BPARM(8) 0 ↵	to solve for all orientations.
> BPARM(9) 0 ↵	to solve for all ellipticities.
> BPARM(10) 0 ↵	to fit for the source polarization model parameters.
> CPARM(1) 0 ↵	to find separate solutions for each IF.
> CPARM(8) 0 ↵	to not limit the number of iterations.
> CPARM(9) 0 ↵	to use the default convergence tolerance.
> CPARM(10) 0 ↵	to use default convergence criterion.
> BADDISK 0 ↵	to specify which disks to avoid for scratch.
> GO ↵	to run the program.

The instrumental polarization may vary rapidly with frequency and independent solutions may be necessary in each IF. Both 'ORI-' and 'RAPR' model the source Q and U as scaled versions of I which

is generally only true in the limit of unresolved or unpolarized sources. The D-terms can be determined iteratively by subtracting estimates of source polarization (Q, U Clean components) in UVSUB. Note: this should be done on data for which instrumental polarization corrections have *not* been applied.

Tasks LPCAL and PCAL also can be used to compute D-term corrections for continuum and spectral-line polarization data respectively. Be forewarned that both of these tasks use linearized D-term models.

9.5.8 Complex Bandpass

For spectral line experiments and continuum observations where a high dynamic range is required, it is be a good idea to do a complex bandpass at this point. In §9.5.4.5 a scalar bandpass was done; *i.e.*, only the amplitude was calibrated but not the phase. Now that the phases have been calibrated in time (by fringe-fitting), a complex bandpass may be solved for. This will take out any dependence of the phase on frequency. To do this, run BPASS, again applying the CL table that includes all the calibration. With the inputs we recommend here, the phases *must be stable in time over the entire bandpass calibrator scan(s).* Check this with VPLOT before proceeding. Then

> TASK 'BPASS' ; INP ↵	to review the inputs.
> INDISK *n1* ; GETN *ctn1* ↵	to select the multi-source visibility data as the input file.
> CALSOUR 'BLLAC' , 'DA193' ↵	to specify the continuum source(s) which were observed for the purpose of bandpass calibration.
> DOCALIB 1 ↵	to apply calibration.
> GAINUSE *CLin* ↵	to indicate the CL table with all calibration up to this point.
> BPVER -1 ↵	do not apply previous bandpass table
> SOLINT 0 ↵	to average data over whole scans before determining the bandpass.
> BPASSPRM 0 ↵	to do a complex bandpass and set rest to 0.
> BPASSPRM(5) 1 ↵	to not divide by "channel 0."
> BPASSPRM(9) 1 ↵	to interpolate over flagged channels.
> BPASSPRM(10) 6 ↵	to normalize the amplitude and phase of the bandpass solutions, using *power* rather than voltage for amplitudes.
> ICHANSEL $chan_{beg}$, $chan_{end}$, 1, IF_{num}, ... ↵	to set channels for entire bandwidth for normalization, if this left 0 then the inner 75% is used and can cause up to 15% error in the amplitude.
> GO ↵	to run the program.

As recommended in §9.5.4.5, you should look at the bandpass with POSSM and BPEDT. If necessary, use BPEDT to flag channels in your bandpass table or calibration data. In the latter case, re-make the BP table. This is the bandpass that should be applied when the data is calibrated and averaged (*e.g.*, with SPLIT).

Note, if your bandpass calibrator is not stable or strong enough during the observation (your bandpass calibrator data should have a better S/N than the data you're trying to correct with the bandpass), you could consider using the phase reference source (if there is one) and use SOLINT = -1 (include all scans to make one bandpass solution). If the bandpass calibrator is strong enough, but the average phase varies through the scan, then divide each record by "channel 0" by setting BPASSPRM(5) = -1 to adjust phase only. You should select a range of channels that have similar phases to be averaged as channel 0 using adverb ICHANSEL.

9.5.9 Baseline-based errors

Baseline-based non-closing phase and amplitude errors can limit the dynamic range of the final images. One way to proceed is to try to solve directly for the non-closing effects using bright, point-like calibrator observations and the task BLCAL. This task writes a BL table containing the estimated non-closing baseline-based errors which can later be applied in SPLIT, or any of the other calibration tasks. To use BLCAL the noise in the calibrator-source images should approach the theoretical limit. Furthermore, the signal-to-noise ratio in the visibility data must be at least 100:1 on baseline-averaged data. BLCAL will divide your data by your best model and then write a BL table containing the baseline-based corrections. Use this task carefully only after reading the EXPLAIN file thoroughly. As an example:

> TASK 'BLCAL' ; INP ↵	to review the inputs.
> INDISK *n1* ; GETN *ctn1* ↵	to select the multi-source visibility data as the input file.
> IN2DISK *n2* ; GET2N *ctn2* ↵	to select your best image as the input model file.
> SOURCE 'DA193' ↵	to select your calibration source.
> DOCALIB 1 ; GAINUSE *clin* ↵	to select the CL table to use.
> BLVER 1 ↵	to create BL table version 1.
> SOLINT 1440 ↵	to determine one complex gain correction per baseline for the whole observation.
> GO ↵	to run the program.

Set ICHANSEL to select only the most desirable spectral channels to average. Task BLPLT may be used to plot the BL table. Task BLCHN is available to compute baseline-based errors on a channel-by-channel basis. It writes a BD table which can be plotted by POSSM and BPLOT, but the calibration itself is not applied by the standard calibration routines. BLCHN writes a calibrated output data set and BDAPL can apply the BD table to another data set. In 31DEC23, procedure BLPOLCAL uses task BLPCL to carry out a similar calibration for both parallel-hand and cross-hand polarizations.

9.6 After initial calibration

9.6.1 Applying calibration

Having obtained your best possible calibration CL table (and BP and BL tables if bandpass or baseline-dependent errors were found), you finally get to make a calibrated data set. This is done with SPLIT, which applies the calibration and splits the database into separate files, one for each source observed.

> TASK 'SPLIT' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> SOURCE ' ' ↵	to write all sources.
> DOCALIB 1 ; GAINUSE clin ↵	to specify which CL table to use.
> DOBAND 1; BPVER 1 ↵	to apply BP table 1 if present.
> BLVER *blin* ↵	to apply BL table *blin* if present.
> APARM(1) 1 ; NCHAV 0 ↵	to have all spectral channels within each IF averaged: read the help file closely, other useful averaging options are available.
> APARM(2) 2 ↵	to have an amplitude correction made for the correlator integration time (in seconds)
> GO ↵	to run the program.

The options for SPLIT given above will apply calibration and then average the spectral channels within each IF, but not average IF channels together . To average over IF channels as well, set APARM(1) = 3 in SPLIT.

(The task AVSPC no longer averages IFs although it is useful in averaging spectral channels in calibrated data sets.)

The task SPLAT can be used instead of SPLIT to do time averaging and different options in spectral averaging. SPLAT can be used also to assemble the selected sources into a multi-source file after applying the specified calibration and averaging. This option allows the user to continue calibration on a smaller data set.

At this point, it is well worth spending time to examine your output visibility data carefully. You may plot the data against time with VPLOT, IBLED, EDITR, or UVPLT, and list them with LISTR, PRTUV, or UVPRT. POSSM is now no longer useful since you have averaged your data in frequency.

9.6.2 Time averaging

It is now convenient to average the data in time using UVAVG both to reduce the bulk of the data and to increase the signal-to-noise for subsequent iterations of the self-calibration/imaging cycle. However, it is important to realize that the fringe-fitting process to this point has only removed gradients of phase over the fringe-fitting solution interval. There will still be stochastic atmospheric (and clock) phase errors affecting the data on short time scales. These phase errors can be significant over minutes at frequencies of 22 GHz and above (and possibly even at 15 GHz) and a reduction in amplitude can occur if data are directly averaged. The ionosphere can cause similar problems at lower frequencies. Self-calibration should remove such phase errors.

For data at frequencies below 15 GHz (and ≈ minute integrations), it should be safe to proceed with UVAVG (see below). For higher frequency data, it may be worth your while to examine the phase coherence of the data first. VPLOT can be used to examine your target or calibrator data to see directly the level of residual phase error over your chosen averaging time. Alternatively, the task IBLED allows you to view the degree of coherence of data averaged over different averaging times. If there are coherence problems (and the target data has enough SNR), CALIB can be run to align the phases prior to coherent averaging. Try:

> TASK 'CALIB' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to specify the single-source input file.
> OUTNA INNA ; OUTCL 'ALIGN' ↵	to specify the output file.
> CALSOUR ' ' ; SMODEL 1, 0 ↵	to use the source with a point-source model.
> DOCALIB -1 ; GAINUSE 0 ↵	to not apply any tables to the input data.
> SOLTYPE ' ' ↵	to use normal least squares.
> SOLMODE 'P' ↵	to solve for phase.
> SOLINT (10.0/60.0) ↵	to solve for phase in 10-second intervals. This should probably be set as low as the strength of the source will allow. The limit is the integration time that gives a SNR > 2 on most baselines.
> ANTWT 1 ↵	to use weights from calibration with no additional weights applied to the antennas. For the purposes of phase alignment, it is appropriate to use the data weights; this allows the noise in the solution to be distributed over the noisiest baselines. This may not be the case when using CALIB for self-calibration in the hybrid mapping sense (see § 9.7).
> APARM(1) 3 ↵	to require 3 antennas present for solution.
> APARM(6) 0 ↵	to skip diagnostic printout.

> APARM(7) 1 ↵ to set the minimum allowed SNR. This limit should be low since the SNR is calculated as a phase difference from model and this can be large. Start with a value ≤ 1.

> GO ↵ to run the program.

Note that CALIB will only give valid solutions if the signal-to-noise over the solution interval on most baselines is greater than 2 (and preferably much higher). At high frequencies on weak sources, it may not be possible to select a solution interval long enough that the signal-to-noise satisfies this criterion, yet short enough to follow the atmospheric phase variations. In such cases, it is probably best not to attempt to self-calibrate the data, but instead to use a short averaging time and to live with any coherence losses in the data.

When the data are sufficiently phase coherent, they should be averaged over time down to a reasonable size using UVAVG:

> TASK 'UVAVG' ; INP ↵ to review the inputs.

> INDISK *n* ; GETN *ctn* ↵ to specify the CALIB output file as the UVAVG input file.

> OUTNA INNA ; OUTCL 'UVAVG' ↵ to specify the output file.

> YINC 30.0 ↵ to set the time-averaging interval to 30 seconds.

> OPCODE ' ' ↵ to enable the averaging operation. There are several options controlling the averaging interval selection and reported times.

> GO ↵ to run UVAVG.

If SPLAT was used to assemble the selected sources into a multi-source file (while applying the preliminary calibration), CALIB will write an SN table which must be converted to a CL table by CLCAL. This new CL table can be applied by SPLAT with time averaging.

9.6.3 Verifying calibration

Before proceeding to image your data, it's worth checking that the calibration performed in § 9.5 is sensible. For each of your sources, produce a plot of the correlated flux density against *uv* distance using UVPLT. As well as identifying bad data which can then be deleted with SPFLG, EDITR, WIPER, or UVFLG, these amplitude versus distances plots (especially those of your calibrator sources) can be used to identify stations where the amplitudes are too high or too low. Furthermore, by fitting simple models to the calibrator data, constant correction factors can be determined for each station which can be used to correct the amplitude calibration. It is often the case that the amplitude calibration to a certain station (particularly non-VLBA stations) is out by a constant factor, either due to uncertainties in the antenna gain or in the noise calibration signal.

In 31DEC20, two tasks allow you to examine selected portions of a *uv* data set statistically. UVRMS averages the selected data in SOLINT time intervals and then can plot the average values as a function of time and as a histogram. A variety of statistical parameters are computed and printed. SPRMS determines and plots the mean and standard deviation of the selected data as a function of spectral channel. Note that both tasks will combine data from multiple sources, baselines, etc. if so instructed. In 31DEC21 RIRMS computes the mean and rms of the real and imaginary parts of the visibilities. It prints these in a baseline matrix and can plot them by baseline as functions of time and as histograms. These tasls have a full set of calibration and data selection adverbs, so they may be run before you run SPLIT.

Most VLBI calibrator sources can be adequately described by one or two Gaussian components. The task UVFIT can be used to fit such a model while finding constant correction factors for the antenna gains. Note that UVFIT can handle no more than 2,500,000 visibilities, so further averaging with UVAVG may be required. The following example shows how to fit antenna gains and a single elliptical Gaussian model of known position and flux to one of the single-source data sets produced by SPLIT or SPLAT (with spectral averaging)

> TASK 'UVFIT' ; INP C_R	to review the inputs.
> INDISK *n* ; GETN *ctn* C_R	to specify the input *uv* data set.
> OPCODE 'GAUS' ; NGAUS 1 C_R	to specify 1 Gaussian component.
> GMAX 1.2 ; DOMAX FALSE C_R	to fix the flux at 1.2 Jy.
> GPOS 0 ; DOPOS FALSE C_R	to hold the position fixed at the origin.
> GWIDTH 0.002 , 0.001 , 45. C_R	to provide an initial guess of the Gaussian widths as 2×1 mas at a position angle of 45 deg.
> DOWIDTH TRUE C_R	to fit for size.
> GAINERR 1 C_R	to fit for all antenna gains with initial guess $= 1$.
> NITER 50 C_R	to limit the fitting to 50 iterations.
> IMSIZE 0.0005 , 0.01 C_R	to limit sizes to be in the range 0.5 to 10 mas.
> DOCAT TRUE ; INVER 1 C_R	to save the solution in a CC file of version number 1.
> INP C_R	to check the inputs.
> GO C_R	to run UVFIT.

UVFIT can also be applied to the multi-source file, but SRCNAME, DOCAL, GAINUSE etc. must then be set. It can handle up to 20 sources to be fit, using a channel-dependent input text file and can write the results in a compact text file form.

Another way to test your amplitude calibration is to use the task UVCRS for bright sources with long tracks. This task calculates correction factors for the amplitudes of the stations using regions of the *uv* plane where *uv* tracks cross. UVCRS can write the correction factors into an SN table.

Once the scale factors are determined, there are a number of options for correcting the data. The simplest option is to apply the correction factors to the single-source data sets using task VBCAL. Alternatively, the correction factors can be incorporated into the highest version CL table of the multi-source data file and task SPLIT run again to make new calibrated data files. The corrections to the CL table are done with CLCOR. Unlike VBCAL, CLCOR must be run separately for each antenna whose calibration you wish to alter; ANTENNA must be set to the antenna number you wish to change, OPCODE = 'GAIN', and CLCORPRM(1) set to the amplitude scale factor found in UVFIT; note that these are *voltage* gains. All higher values in the array CLCORPRM should be zero. The effect of the altered calibration can be viewed using UVPLT with DOCAL = 1. If it is satisfactory, SPLIT can be re-applied to the data. We recommend using CLCOR to perform such amplitude corrections. It now produces a new CL table each time it is used unless you specify both GAINVER and GAINUSE as having the same, non-zero version number.

Another way of incorporating amplitude corrections is to edit the calibration text files used by ANTAB. This can be accomplished by setting the FT parameters for affected stations. For instance, if the Bonn scale factor is 1.043, set the FT parameter on the BONN TSYS card to FT=(1.043*1.043). If amplitude calibration was carried out *after* fringe-fitting (*not recommended!*, then it is only necessary to rerun ANTAB, delete the latest CL table containing amplitude calibration and rerun APCAL using the highest CL table produced in the fringe-fitting step. If however, as we have described in this chapter, the amplitude calibration was done prior to fringe-fitting, then correcting the amplitudes is more involved. It is probably best to delete all CL tables except the first one and start again at § 9.5. However, it may not be necessary to carry out the time consuming FRING solutions again. If the amplitude changes are small, the phase, rate and delay solutions will be essentially unchanged. Therefore, with care, the existing SN tables can be used in lieu of re-running FRING.

9.7 Self-calibration, imaging, and model-fitting

We are now in a position to make images. As with VLA data, we do this by iteratively self-calibrating the data and deconvolving using Clean, MEM *et al*. We describe below a typical self-calibration and imaging sequence for VLBI data. The tasks used are described in more detail in Chapter 5.

1. CALIB self-calibrates the *uv* data.
2. IMAGR images and Cleans. IMAGR is now the preferred task for imaging VLBI data in *AIPS*.
3. SCMAP images, Cleans, and self-calibrates. SCMAP is meant to provide the functionality of the popular VLBI data analysis package difmap. SCIMG is a multi-field version of SCMAP.

The main difference between the processing of VLBI and VLA data is that, unless phase-referencing is used, absolute phases of VLBI data are un-calibrated. Therefore, many more iterations around the imaging loop are required for the VLBI case; dozens of self-calibration iterations are not uncommon. Given this, it may be convenient to use tasks SCMAP or SCIMG which cycle between imaging and self-calibration with an option to edit the UV data. If you wish to edit the Clean components or make plots, you will need to do the manual Clean/self-cal process with IMAGR and CALIB.

Note that VLBI imaging is not an exact science and there are a number of different views on the "correct" imaging method and the "correct" software to accomplish this method. Some users take their *AIPS* data into a package written at CalTech to use the difmap program (see ftp://phobos.caltech.edu/pub/difmap/). Others use the *AIPS* tasks SCMAP and SCIMG (see § 5.4). It is beyond the scope of this document to explain in detail all aspects of VLBI imaging. For more details, see Craig Walker's chapter on "Practical VLBI Imaging" in the publication *VLBI and the VLBA*, **1995**, edited by A. Zensus, P.J. Diamond, and P.J. Napier which is available on the World-Wide Web (www.cv.nrao.edu/vlbabook). Here we make a few suggestions on how to control CALIB and IMAGR. Again, please note that the latter is preferable to all previous imaging tasks; SCIMG and to a lesser extent SCMAP offer the same improved imaging techniques. They do not offer several experimental algorithms found in IMAGR including Clean component filtering, the SDI Clean algorithm, and multi-resolution Cleaning.

9.7.1 CALIB

1. Start by correcting antenna phases only, *i.e.*, use SOLMODE = 'P' ↵. Switch on the amplitude correction only after you have converged to a fairly good image. On the first iteration, you will need to invent an input model. For most extragalactic continuum sources, a point-source model is a good choice. Set SMODEL(1) to the zero-spacing flux density as extrapolated by eye using UVPLT and consider using a circular Gaussian model at the origin to reduce the impact of the longest spacings. Start with the so-called SNR parameter APARM(7) small ($\leq$ 1) and gradually increase it as the image improves. If this parameter is large during early iterations, when the model used is far from correct, then large portions of your data in the output file may be flagged. This is not too important if you use the original input *uv* file as the input to CALIB in all iterations. You can also set APARM(9) = 1 to leave data affected by failed solutions uncalibrated.
2. On subsequent iterations, use the Clean image as produced by IMAGR as your input model. VLBI applications usually require Clean components well beyond the first negative component to be used in calculating the source model. One possibility is to use PRTCC to find the point where a significant fraction (*e.g.*, one third) of all new Clean components are negative. An alternative is to use all of the Clean components, but to use tight windowing in IMAGR — which can now be done interactively on the TV as IMAGR progresses. Alternatively, use tight windowing and clipping of the Clean components in IMAGR (IMAGRPRM(8) and IMAGRPRM(9)) or afterward with CCEDT or CCSEL before running CALIB. Tight windowing is especially important when *uv* coverage is poor. Editing Clean components after

IMAGR, but before CALIB, can be effective in removing possibly spurious features; if they are real they will usually reappear in later iterations. IMAGR offers a filtering option to remove weak, isolated components when requested from the TV and at the end before the components are restored to the image. This removes much of the need for CCSEL.

3. When carrying out the next CALIB iteration with the new Clean model, you can either self-calibrate the original data set or, alternatively, self-calibrate the data set (output by CALIB) which was used to produce that Clean model. It is advantageous to use the original data set at least until you turn on amplitude calibration. At that point, you should stick with the file produced from the original data with the best phase-only solution. Amongst other things, this can prevent the telescope amplitudes from "wandering" (see below).

4. When the model contains extended structure, there may be problems with convergence when weighting the data "correctly" (*i.e.*, by $1/\sigma^2$.) The WEIGHTIT adverb allows you to use less extreme weights, such as $1/\sigma$ or $1/\sqrt{\sigma}$. If your array contains antennas that have a wide range of sensitivities, *e.g.*, the VLBA plus the phased-VLA and/or the Effelsberg 100-m, it is helpful to alter the weights of the antennas in your CALIB solutions. If this is not done, then your solution will be dominated by only a few baselines and the uniqueness of the solution is not guaranteed. Use PRTUV to inspect the weights of your data. Then set the CALIB input array ANTWT, which provides multiplicative factors adjusting the weights for each antenna prior to the CALIB solution. Set these parameters so that the effective range of baseline weights is only 10 to 100. Alternatively, use WTMOD to raise the original weights to a power between 0.25 and 0.5.

5. As you iterate, keep an eye on how the model image is converging to fit the data. Use VPLOT, CLPLT, CAPLT, and UVPLT.

6. When your source has a lot of extended structure and/or your VLBI array has relatively few short spacings, you should consider setting UVRANGE to only include the range of spacings in which the model provides a good fit to the data. However, given the relatively small number of antennas in most VLBI observations, you may need to compromise to allow in enough baselines to get good self-cal solutions. Set `WTUV > 0`.

7. When you are finally ready to solve for amplitude corrections, you should first apply all previous phase calibration including the final phase-only self-calibration solution. Then run CALIB setting `SOLMOD 'A+P'`, initially setting the solution interval (SOLINT) to a longer time than used for phase-only solutions (*e.g.*, 3 times). Try to prevent the antenna amplitudes from "wandering," which can sometimes happen if there is still a significant amount of short spacing flux density missing from the source model. Setting UVRANGE is useful, as is setting `CPARM(2) = 1` to constrain the mean amplitude solutions over all antennas to be one. You can also set `SOLMODE='GCON'` and the array GAINERR to the expected standard deviation of the gains for each antenna. This constrains amplitude solutions to conform to the expected statistics. Setting the gain constraint factor SOLCON to values larger than 1 will increase the importance of these gain error constraints. Finally, going back and self-calibrating starting with the original data set and the best available Clean model is useful way to prevent amplitude wander.

CALIB has been enhanced to improve its usefulness for SVLBI data sets. This has involved the implementation of improved antenna selection and partial array calibration, through the new adverb DOFIT. The possibility of solving only for a subset of selected antennas has been implemented in this manner and may prove useful for SVLBI data. The implementation of adverb DOFIT in CALIB is analogous to its implementation in FRING (see § 9.5.7.7).

9.7.2 IMAGR, SCIMAG, and SCMAP

1. Before using IMAGR or SCMAP or SCIMG, print out and read the EXPLAIN file. They are powerful and complicated tasks with many adverbs — some of which are new — and shouldn't be used blindly.

2. The quality of images produced may depend on the type of weighting used. With VLBA-only experiments, the best quality images are often produced using natural (UVWTFN 'NA' ↵) weighting in IMAGR. These images will represent the extended structure of the source better. If the highest resolution is required, try uniform (UVWTFN 'UN' ↵) weighting. The ROBUST parameter allows weightings intermediate between these two extremes often with both good signal-to-noise characteristics and a narrow synthesized beam. It may also be worth experimenting with the UVBOX parameter to allow smoothing of weights over larger areas of the *uv* plane (*i.e.*, to use "super-uniform weighting"). If the array contains antennas with very different sensitivities, (for instance, if it includes Effelsberg, the phased-VLA, and/or HALCA), then it may be advantageous to alter the weights of baselines to these antennas. Although this increases the thermal noise in the image, it will improve the *uv* coverage, which, otherwise, will contain effectively only the baselines to the most sensitive antennas. One way of doing this is to use UVWTFN = 'UV' ↵ in IMAGR. This option takes the fourth root of the input weights before applying uniform weighting. SCMAP and SCIMG also support these weighting options. Another flexible (but deprecated) approach is to use task WTMOD to change the weights in the data set prior to running IMAGR.

3. After an initial self-calibration against a point-source starting model, the deconvolved image will often show spurious symmetric structure. Convergence can be speeded up by placing Clean boxes CLBOX) around the side showing the brighter structure. Note that CLBOX may be used to produced circular as well as rectangular windows for use with IMAGR. Alternatively, CCEDT can be used to edit the Clean components after IMAGR, but before the next CALIB. IMAGR, SCIMG, and SCMAP allow this to be done interactively at the start of each major Clean cycle including the first.

4. Use DOTV = 1 ↵ to view the residuals and possibly modify the Clean boxes as you Clean. You can stop Cleaning if you feel that you are including spurious structure into your model or if you feel you need to reset a Clean box to include a new feature. Note the BOXFILE and OBOXFILE options which allow you to retain interactively set Clean boxes for use in the next self-cal iteration.

5. SCMAP and SCIMG, at each self-calibration cycle, offer a powerful interactive data editing tool which displays input and residual data from up to 11 baselines simultaneously. This is the same editor as found in task EDITR.

The hints outlined above are by no means the whole story when it comes to self-calibrating and imaging VLBI data. Unfortunately, it can still be somewhat of an art form. Very experienced users can produce noise-limited images, but there is no simple recipe that will enable inexperienced users to do the same.

9.7.3 Non-conventional methods of imaging

Since fringe rate is a function of source position, we can use measurements of the fringe rate to estimate the positions of sources. Fringe rate imaging is widely used with maser sources which are usually widely separated point objects. The angular resolution of fringe rate images is not as good as conventional aperture synthesis imaging, but it can be used for an initial determination of the location of emitting clusters. This will help in setting field shifts and Clean windows for use by IMAGR. Having measured the fringe rate $FR(i)$ for each sample in (baseline-time) and computed the derivative of fringe rate ($UDOT(i)$) wrt to right ascension (X) and the derivative ($VDOT(i)$) wrt declination (Y), we obtain a set of equations of straight lines

$$FR(i) = UDOT(i) * X + VDOT(i) * Y$$

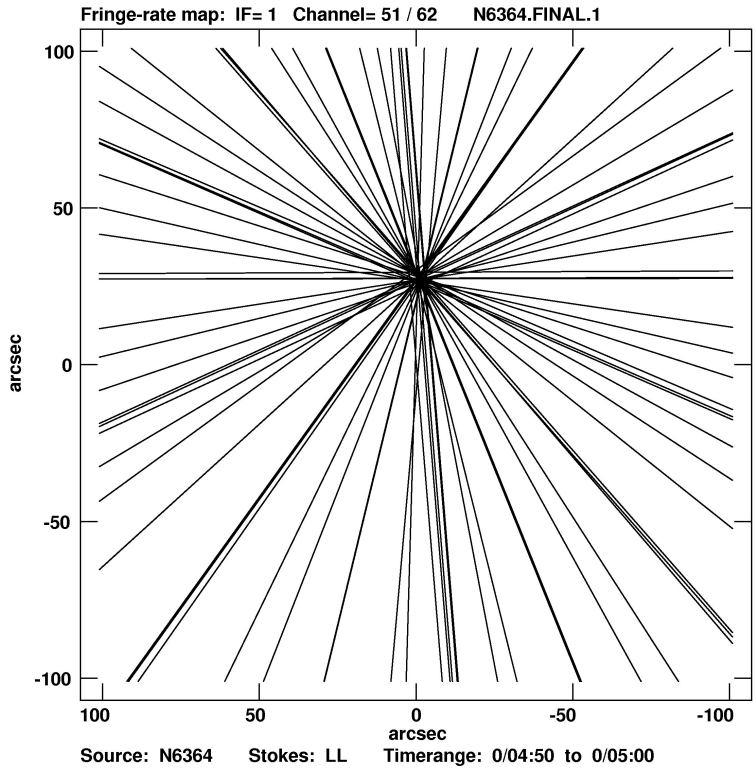


Figure 9.3: An example of a preliminary determination of the source location using fringe-rate analysis. The position of the source in channel 51 is shifted relative to the reference channel (62) by ≈ 25 asec to the north. Conventional imaging can be carried out near this position.

describing the loci of constant fringe rate for each observation. The positions of the source component(s) can be determined by finding the places of highest density of crossing lines. (Details can be read in Giuffrida,T.S, 1977 Ph.D.thesis MIT and in Walker R.C., 1981, *Astr. J.*, **86**, 9, 1323.)

This algorithm is implemented in *AIPS* as the task FRMAP. The task plots the straight lines on the TV or prepares a plot file. Then it determines the positions of higher density of crossing lines. The coordinates in RA and DEC of the components found are written in an output file. The fringe rate method is especially attractive in SVLBI because of the better angular resolution due to faster movement of the orbiting antenna. Figure 9.3 shows an example of a fringe rate image.

FRMAP can be used for more accurate definition of the source coordinates that is required by the correlator. In this case the correlator carries out two passes. After the first one (short) FRMAP is used to find a more accurate position of the source. The new coordinates are then used in the second pass covering the whole experiment.

9.8 Summary of VLBI calibration tables

Several different tables are supplied with a VLBI data set created by the Socorro and other correlator; various other tables (*e.g.*, SN tables) are created by the calibration process. A list of these tables is given below for your edification. Not all tables will be present with all files. See *AIPS* Memos 114 and 117, reachable via https://www.aips.nrao.edu, for complete documentation of the table and data formats used by *AIPS*.

AN Antenna table. Contains a list of the antenna names and station coordinates. Also contains instrumental polarization terms.

AT Antenna characteristics table. Contains additional information about antenna properties, including some time variable quantities.

BL Baseline offset table. Contains non-closing baseline-dependent phase and amplitude errors as determined by BLCAL.

BS Baseline solution table. Contains baseline delay, rate and phase solutions as determined by BLING.

CL Calibration table. Version 1 contains, amongst other things, the default calibration parameters for the amplitude (usually unity), phase, and single-band delay (usually zero) for each source for each IF as a function of time. It also contains polynomial coefficients allowing the correlator delay and phase models to be recomputed. As calibration proceeds, higher versions of this table are created which incorporate more and more calibration effects into the phase, delay, and amplitude entries.

CQ Correlator parameter frequency table. Contains VLBA correlation parameters for each *AIPS* IF, and activates VLBA delay decorrelation corrections. Type EXPLAIN FXVLB for further information.

CT CALC table. Contains the input parameters passed to CALC to generate the polynomials recorded in the IM table.

FG Flag table. Contains information used to delete selected portions of the data.

FQ Frequency table. Contains information about the IF frequencies, channel spacings, bandwidths, etc.

GC Gain Calibration table. Contains the expected zenith gain and gain-elevation curve for each antenna. It is used for amplitude calibration.

HF Haystack FRNGE table. Contains information generated from the *AIPS* tables that can be exported to the CALC and SOLVE package.

IM Interferometer model table. Contains the actual polynomial coefficients which the VLBA correlator used to calculate the geometrical model. Unlike the coefficients in the CL table, these have not been re-interpolated onto the CL time grid, but have time stamps corresponding to the times at which the correlator computed the geometrical model.

MC Model Components table. Contains the various components of the geometric model used in the VLBA correlator to generate the IM table.

NX Index File. Contains information about the time, source, sub-array and location within the data file of each observation or "scan." It is used by some *AIPS* tasks in accessing the main data file and subsidiary tables.

OB Spacecraft Orbit table. Contains information about the positions and velocities used by the correlator for an orbiting antenna.

PC Phase-calibration table. Contains phases within each IF computed from the injected phase calibration signals. It is used to determine the phase offsets and single-band delays for each IF channel.

SN Solution table. Contains antenna delay, rate, phase, and amplitude corrections solved for by CALIB and FRING and other tasks.

SU Source table. Contains a list of the sources found within the multi-source file, including information on source positions and flux density.

TY System temperature table. Contains the system temperature as a function of time for each antenna and IF channel. It is used for amplitude calibration.

VT VLBA Tape table. Contains tape playback statistics for use mainly by the VLBA correlator group.

WX Weather table. Contains weather-related information for each station.

9.9 Additional recipes

9.9.1 Bananes rôties

1. Preheat oven to 375° F.
2. Place 6 (peeled) **bananas** in a baking dish.
3. Sprinkle bananas with juice of 1/2 **lemon**.
4. Pour 2 tablespoons melted **butter** and 2 tablespoons **dark rum** over the bananas. Sprinkle with 2 tablespoons **brown sugar**.
5. Place in oven for 10 minutes.
6. Pour on 2 more tablespoons **melted butter** and 2 more tablespoons **dark rum** and bake for 5 minutes more.
7. Serve at once, spooning some sauce over each banana.

9.9.2 Banana curried chicken

1. Fry 2 chopped **onions** in 50 ml **cooking oil** until light brown.
2. Add 1/4 cup **cake flour** and mix well. Add 1 (cup?) **chicken stock** gradually while stirring.
3. Add 1 cup **raisins**, 1 teaspoon **salt**, 2 pounds cooked, boned **chicken**, 5 sliced **bananas**, 2 grated **apples**, 2 tablespoons grated **lemon rind**, 1 tablespoon **sugar**, 1 1/2 tablespoons **curry powder**, 1 **bay leaf**, 4 **peppercorns**.
4. Cover saucepan and simmer for 20 minutes.
5. Remove bay leaf. Add 1 cup **cream** and heat just before serving.
6. Serve on a bed of rice. Decorate with pineapples if preferred.

Thanks to Turbana Corporation (www.turbana.com).

10 Single-Dish Data in $\mathcal{AIPS}$

$\mathcal{AIPS}$ was not originally intended as a reduction package for single-dish data and cannot be considered as such today. However, because of the similarity of single-dish data taken at "random" pointings on the sky to interferometric data taken at "random" locations in the *uv* plane, $\mathcal{AIPS}$ was seen as a system to be used to solve large imaging problems arising from single-dish observations. Many of the $\mathcal{AIPS}$ *uv*-data tasks are able to do something sensible — or even desirable — with single-dish data and a few special tasks to process single-dish data have been written. The present chapter contains a discussion of the representation of single-dish data in $\mathcal{AIPS}$ followed by a description of how such data may be calibrated, corrected, converted into images, and analyzed by $\mathcal{AIPS}$. A final section on using single-dish observations to improve the imaging of interferometric data represents what little we now know about this potentially important process.

10.1 $\mathcal{AIPS}$ format for single-dish data

Single-dish data in $\mathcal{AIPS}$ is treated as *uv* data with different, but related random parameters and with the imaginary part of the visibility replaced by an additive calibration or offset. The *u* and *v* random parameters are replaced by parameters labeled RA and DEC, although other labels such as ELON, ELAT, GLON, and GLAT are also recognized. (Conversion between these coordinate systems is not provided in the "*uv*" plane although some conversion can be done on images.) The random parameter data are the sample coordinates in degrees. The TIME1 random parameter is the time (IAT) since midnight on the reference date in days as with real *uv* data. The BEAM random parameter corresponds to BASELINE and is used to separate data which should be edited and calibrated separately (*e.g.*, separate beams of a multi-feed system, different polarizations or observing runs of a multi-polarization system). The actual beam number is recorded as 257 times the desired number so that visibility-data tasks will recognize the "baseline" as auto-correlation data. Two other random parameters, SCAN and SAMPLE, have no relation to any visibility parameters and are simply used to retain the "scan" number and sample number within the scan which are traditional in single-dish observations. Very little is made of these, but INDXR will make a new index entry when the scan number changes and PRTSD will display the scan and sample numbers. SDVEL uses the scan numbers to determine when to update the reference velocity in some observing modes. Single-dish data may be stored in compressed form, in which the weight and compression scale are stored as random parameters exactly as in true visibility data. This should *not* be done if the applied offset is large as in beam-switched continuum observations.

The measured single-dish flux, usually in units of degrees Kelvin, appears in the real part of the "complex visibility." The imaginary part of the visibility is sometimes used to hold an offset which can be applied to the data to remove, for example, a time-variable bias. The data weight is used to weight the data and should be proportional to σ^{-2}, where σ is the uncertainty in the flux. The visibility sample can contain multiple polarizations, described with the STOKES axis (values 1 through 4 for I, Q, U, and V, respectively). The sample can also contain multiple spectral-line frequencies, described with a FREQ axis giving the observed reference frequency and increment in Hz.

The *uv*-data header in an $\mathcal{AIPS}$ data set is expected to contain the reference (usually central) longitude and latitude given either as 1-pixel coordinate axes and/or in the "observed" coordinate location. The convolution size is usually used to hold the single-dish beam width (fwhm) and rest frequency and velocity information should also appear with spectral-line data. Many of the parameters can be added to the header by the user if they are missing and needed. Verbs ADDBEAM, ALTDEF, and PUTHEAD are useful for this purpose. A complete data set will also have an antenna extension file giving the location of the antenna. This allows tasks to compute things like zenith angles and Doppler corrections when needed.

10.1.1 On-the-fly data from the 12m

At the present time, the only reliable routes for single-dish data into *AIPS* are provided by the tasks OTFBS and OTFUV. These tasks work only on beam-switched continuum and on spectral-line observations, respectively, from the NRAO 12m telescope. They use files in the UniPops native format and do not read the FITS table format written by UniPops. Both programs are designed for "on-the-fly" or "OTF" observing modes in which the telescope takes data rapidly while continuously changing its pointing position.

10.1.1.1 Listing OTF input files

To read OTF files, you must first define an environment variable to point to the disk area in which your data resides. This environment variable and your file names should be in upper case letters, but there is an AIPS "feature" which allows you to use lower case. On Unix systems, you may set the environment variable and rename the files to upper case with

% cd /my/disk/directory C_R	to switch to the disk directory containing your data.
% setenv MYAREA 'pwd' C_R	to define $MYAREA under c shell, or
% export MYAREA='pwd' C_R	to define $MYAREA under Bourne, bash, korn shells.
% mv mysdd.file MYSDD.FILE C_R	to rename the data file to upper case letters.
% mv mygsdd.file MYGSDD.FILE C_R	to rename the gain file to upper case letters.

Then start your AIPS session.

To review the contents of your data set, use the task OTFIN which will list SDD modes, IF and scan numbers, times, coordinates, velocities, and number of samples. This output should help in setting the range of scan numbers to be loaded by OTFUV or OTFBS. Type:

> TASK 'OTFIN' ; INP C_R	to list the required inputs on your screen.
> DATAIN 'MYAREA:MYSDD.FILE' C_R	to specify the name of the 12m raw data file, where MYAREA is an environment variable which points at a disk data area and MYSDD.FILE is the name of your file in that area. See § 3.10. If your environment variable and/or your file name contain lower case letters, type the name carefully with the correct case for all letters and leave off the second (close) quote mark. When you use this "feature" of the AIPS compiler, you cannot type anything following the DATAIN name (or other string adverb) on that line.
> BCOUNT 0 ; ECOUNT 0 C_R	to include all 12m scans in the file.
> BIF 0 C_R	to include all SDD "IFs."
> DOCRT -1 C_R	to print the listing on the line printer, or
> DOCRT 1 C_R	to view the listing, one page at a time, on your terminal window. The width given (if > 72) should match the width desired; a width of < 72, as given here, uses the actual width of the window and so maximizes the information per line.
> INP C_R	to review the parameters.
> GO C_R	to run the task.

10.1.1.2 Reading spectral-line OTF files into *AIPS*

To run OTFUV after running OTFIN, type

> TASK 'OTFUV' ; INP ↵	to list the required inputs on your screen.
> DATAIN 'MYAREA:MYSDD.FILE' ↵	to specify the name of the 12m raw data file, where MYAREA is an environment variable which points at a disk data area and MYSDD.FILE is the name of your file in that area. See § 3.10.
> DATA2IN 'MYAREA:MYGSDD.FILE' ↵	to specify the name of the 12m gain file corresponding to the file specified with DATAIN.
> BCOUNT n_1 ; ECOUNT n_2 ↵	to include 12m scans n_1 through n_2 in the output file.
> BIF 0 ; EIF 0 ↵	to include all SDD "IFs" matching the lowest numbered one found. IFs which do not match in central frequency or channel width are skipped.
> DOUVCOMP TRUE ↵	to write the data in a compressed format. This reduces the size of the file by nearly a factor of 3 with no significant loss of information in this case.
> XINC 1 ; YINC 1 ↵	to write out all data samples with no time averaging. One can smooth by YINC samples and write out the data every XINC sample times in order to reduce the size of the output data set and improve the signal-to-noise of the individual samples with only a minor loss of information..
> DOWEIGHT 1 ↵	to use offs and gains interpolated to the time of each observation. This seems to produce better results.
> DETIME 0 ↵	to add no offset to the actual observation times.
> BCHAN 0; ECHAN 0 ↵	to include all spectral channels.
> CHANSEL 0 ↵	to flag no channels. CHANSEL 31,34,3 ↵, for example, would mark channels 31 and 34 as bad. Data may be edited later more selectively.
> INP ↵	to review the parameters.
> GO ↵	to run the task.

While OTFUV runs, it will show you (on the message monitor or your window) the name and location of the output *AIPS* file created and then provide a list of the scans and IFs read and the gain scans used upon them.

In many cases, the 12m in OTF mode observes two separate polarizations using the same center frequency and spectral resolution. In the UniPops/12m nomenclature, these are separate "IFs." A similar nomenclature is used to distinguish the feeds in the multi-feed system. OTFUV can now read up to eight IFs at the same time, avoiding the necessity of multiple runs of OTFUV, followed by a data sort to restore time order. OTFUV will distinguish the IFs not by an *AIPS* "IF axis," but by assigning them beam numbers equal to the SDD IF number (or autocorrelator baseline number equal to the SDD IF number with itself).

You may append data from another IF in the first input data set or data from another OTF pass on the source to the *AIPS* data set created above, by entering new DATAIN and DATA2IN names and new BCOUNT and ECOUNT ranges, if needed and

> BIF m_1 ; EIF m_2 ↵	to load IFs m_1 through m_2.
> DOCONCAT TRUE ↵	to enable the concatenation mode.
> OUTDISK n ; GETONAME m ↵	to select the output file, where n and m are the output disk and catalog slot number used by the first run of OTFUV.

> FQTOL *ff* ↵ — to allow data sets within *ff* MHz of each other to be concatenated. Doppler tracking will cause two OTF passes to appear to be at separate frequencies. Narrow-band, wide-field observations should not be concatenated in this way; see the discussion of SDVEL below (§ 10.2.4).

> GO ↵ — to run the task appending the additional data.

Another way to concatenate two 12m IFs — or multiple observing runs — is to create two output files with OTFUV and then concatenate them with DBCON. If the two OTFUV files are in time order, then DBCON will actually merge the two data sets, retaining the time order. Avoid the use of multiple sub-arrays, which are a useless complication in this case, by setting DOARRAY = 0. To have the most "complete" antenna file, put the data set with the higher 12m IF in the first input name set (INNAME, INCLASS etc.)

10.1.1.3 Reading continuum OTF files into *AIPS*

The NRAO 12m telescope can observe in a beam-switched continuum on-the-fly mapping mode. Such data may be read into *AIPS* and reduced, in a somewhat experimental fashion, into images. To read in the data (after using OTFIN), enter

> TASK 'OTFBS' ; INP ↵ — to list the required inputs on your screen.

> DATAIN 'MYAREA:MYSDD.FILE' ↵ — to specify the name of the 12m raw data file, where MYAREA is an environment variable which points at a disk data area and MYSDD.FILE is the name of your file in that area. See § 3.10.

> BCOUNT n_1 ; ECOUNT n_2 ↵ — to include 12m scans n_1 through n_2 in the output file.

> BIF 0 ; EIF 0 ↵ — to include all SDD "IFs" matching the lowest numbered one found. IFs which do not match in central frequency or channel width are skipped.

> INP ↵ — to review the parameters.

> GO ↵ — to run the task.

While OTFBS runs, it will show you (on the message monitor or your window) the name and location of the two output *AIPS* files created (one for "plus" and one for "minus" beam throws) and then provide a list of the scans and IFs read with the number of samples. The two output files will have the same names except for a "+" and a "−" as the sixth character of the output class.

10.1.2 Other input data formats

Another method for getting single-dish data into *AIPS* is through the use of FITS-format binary tables. If the data are able to be put in a usable table, then the *AIPS* FITS reading tasks such as FITLD (see § 5.1.2) can be used to read them into a disk table attached to a cataloged file. Then SDTUV can be used to convert the table into the *uv* format described above applying a variety of calibrations along the way. Unfortunately, the non-*AIPS* program that did the UniPops to FITS conversion has been lost and the *AIPS* FITS readers cannot handle the FITS tables written by UniPops. There are two problems with the latter: *AIPS* is unable to handle tables with more than 128 columns while UniPops writes tables with around 200 columns. Even if *AIPS* could be extended in some special task, it would be unable to handle the current UniPops tables since the parameters given do not correctly describe the contents. Specialized unpublished knowledge about each receiver is required to disentangle the coordinate information and data structure.

The task SDTUV expects a sequence of related tables each with a number of keywords giving useful information such as scan, observer, telescope, object, scan start UT date and time, sample rate, velocity, and the like. The data are then a regular time sequence with each row of the table containing the right

ascension, declination, and data for N receivers. Breaks in the time sequence are assumed to be new scans found in the next table. SDTUV has the ability to apply receiver position offsets and pointing corrections and to fit and remove receiver baselines using a sliding median window and spline fit. Interference rejection, lateral defocusing corrections, and a priori baseline removal are also offered. At present SDTUV is an example of what can be done rather than a directly usable task. It is limited to continuum problems currently and is moderately restricted in the number of data samples that can be read in any one scan.

Therefore, it will be necessary to write some sort of program in addition to those in the standard AIPS release to get single-dish data into AIPS. We encourage anyone who develops such a program to provide it to the AIPS group so that we may offer it to other single-dish users.

10.2 Single-dish data in the "uv" domain

Once you have gotten your data into AIPS, a wide range of tasks become available to you. In addition to the single-dish specific tasks discussed below, these include data movement tasks (UVCOP, UVSRT, DBCON), data averaging (AVER, UVAVG, AVSPC), non-interactive editing (CLIP, UVFLG), interactive editing (SPFLG, EDITR, TVFLG), data backup and restore (FITTP, FITLD), and data display (PRTAN, PRTUV, UVPRT, UVPLT).

10.2.1 Using PRTSD, UVPLT, and POSSM to look at your data

In the process of calibrating, modeling, editing, and imaging of single-dish data, there are occasionally problems that seem to arise because users are not aware of the data that they actually have. PRTSD is the task for such users. It displays the data with or without calibration for selected portions of your data set. This will help you identify what pointing positions actually occur in your data, which channels are highly variable or bad, and the like. SPFLG, UVPLT, and others are good for looking at the data set as a whole, but PRTSD really shows you what you have.

To run it, type:

> TASK 'PRTSD' ; INP ↵	to list the required inputs on your screen.
> INDISK n ; GETN ctn ↵	to select the single-dish "uv" file to be displayed.
> DOCRT 1 ↵	to select the on-screen display at its current width; make sure your window is at least 132 characters across for the best results.
> DOCELL -1 ↵	to look at the data values; DOCELL > 0 causes the offsets that have been removed (usually 0) to be displayed.
> CHANNEL m ↵	to display channels m through $m + 5$.
> DOCAL FALSE ↵	to apply no calibration. Note that the 12m off scans and instrumental gains are applied by OTFUV; this parameter applies only to any additional calibration contained in CS files. See § 10.2.3.
> TIMERANG 0 ↵	to look at all times.
> ANTENNAS $a1, a2, \ldots$ ↵	to look at beams/IFs $a1, a2, \ldots$ only.
> BPRINT bb ↵	to begin the display with the bb^{th} sample in the data set *before* application of the other selection criteria (TIMERANG, ANTENNAS, etc.)
> NPRINT 2000 ↵	to shut off the display interactively or after a lot of lines.

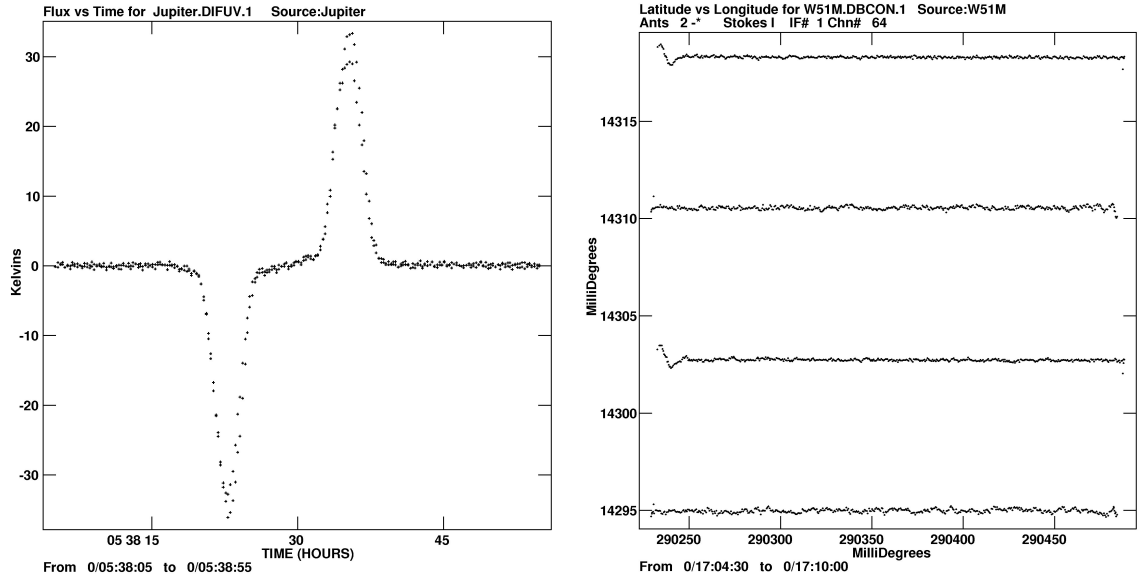


Figure 10.1: *left:* UVPLT display of 12m beam-switched continuum data on Jupiter. The time range is set to display one row of the OTF observation and the "minus" beam throw data have been subtracted from the "plus" throw. *right:* UVPLT display of the right ascension and declination of each sample in spectral-line OTF data set over a limited time range.

> XINC x ↵ — to display only every $x^{\rm th}$ sample of those selected by the other criteria.
> INP ↵ — to review the inputs.
> GO ↵ — to start the task.

PRTSD will start and, after a pause to get through any data not included at the start of the file, will begin to display lines on your terminal showing the scan number, time, coordinates, and data for six spectral channels. After 20 or so lines, it will pause and ask if you want to continue. Hit ↵ to continue or type Q ↵ or q ↵ to quit. If you decide to get hard copy, set DOCRT = -1 and the output will be printed. To save the display in a text file, without printing, set DOCRT = -1 and give the name of the file in the OUTPRINT adverb. See § 3.2 and § 3.10.1 for more information on printing.

There are a number of tasks which plot *uv* visibility data; see § 6.3.1. The most basic of these is UVPLT, which can be useful for single-dish data sets. For example, to generate the plot of flux versus time in 12m OTF beam-switched continuum differenced data seen in the accompanying figure (Figure 10.1), the parameters given below were used:

> TASK 'UVPLT' ; INP ↵ — to review the inputs.
> INDI n ; GETN ctn ↵ — to select the disk and catalog entry of the data set.
> DOCALIB FALSE ↵ — to apply no calibration; UVPLT does not understand single-dish calibration.
> BPARM = 11,9,0 ↵ — to plot time in hours on the x axis and flux in Kelvins on the y axis. The other parameters can be used to specify fixed scales on one or both axes, but are just self-scaled in this example.
> XINC 1 ↵ — to plot every selected sample.
> BCHAN 1 ; ECHAN 1 ↵ — to plot only "spectral channel" 1, the actual data values.
> ANTENNA 1,0 ; BASELINE 0 ↵ — to do all baselines with antenna 1, namely 1–1 or, in 12m nomenclature, IF 1..
> TIMER = 0, 5, 38, 5, 0, 5, 38, 55 ↵ — to restrict the times to a single scan.

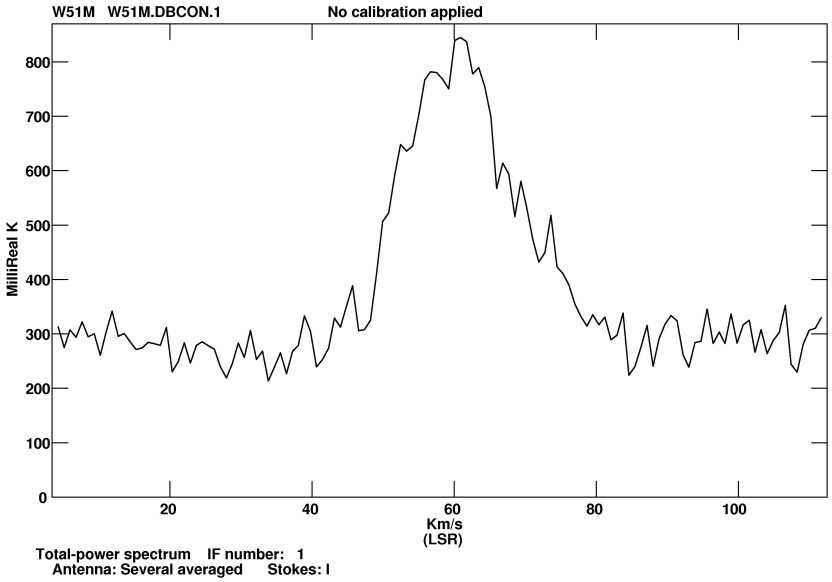


Figure 10.2: POSSM display of all of a 12m observation taken on W51. All samples (on and off the actual source) and both "antennas" are averaged together.

> DOCRT = -1 ; GO C_R — to make a plot file of these data.

After UVPLT is running, or better, after it has finished:

> PLVER 0 ; GO LWPLA C_R — to plot the latest version on a PostScript printer/plotter.

The second plot in Figure 10.1 was generated with BPARM = 6, 7 and shows where samples occur on the sky in a different data set.

With spectral-line data, POSSM will plot observed spectra averaged over selected "antennas," time ranges, and the like. Thus,

> TASK 'POSSM' ; INP C_R — to review the inputs.
> INDI *n* ; GETN *ctn* C_R — to select the disk and catalog entry of the data set.
> BCHAN 0 ; ECHAN 0 C_R — to plot all channels.
> ANTENNAS 0 ; BASELINE 0 C_R — to average all 12m IFs.
> TIMERA 0 ; SOLINT 0 C_R — to average all times into one plot.
> APARM(7) = 2 C_R — to have velocity labels on the *x* axis.
> GO C_R — to run the task.

LWPLA was then used to make a PostScript version of the plot seen in Figure 10.2.

10.2.2 Using UVFLG, SPFLG, and EDITR to edit your data

Editing is the process by which you mark data samples as "unreliable" or "bad." In *AIPS*, there are two methods for doing this. The simplest is to have the editing software alter the weight of the sample to indicate that it is flagged. If the data are not compressed, this is a reversible operation. If the data are compressed, however, then the data themselves are marked as "indefinite" and the operation is not reversible. The second method is the use of a flag (FG) extension table attached to your *uv* data set. This method requires that the data be sorted into time order for large FG tables and is supported by most, but not all, tasks. Small (< 6000 row) FG tables may be used with data in any sort order. If the task does not

have the FLAGVER adverb, then it does not support flag tables. However, since flag tables can be applied to the data by SPLIT, we use them in the recipes below.

To sort the data into "time-baseline" (TB) order,

> TASK 'UVSRT' ; INP ↵	to review the inputs.
> INDI *n* ; GETN *ctn* ↵	to select the disk and catalog entry of the data set.
> SORT 'TB' ↵	to sort into time-baseline order.
> ROTATE 0 ↵	to avoid damage to the coordinates.
> INP ↵	to check the parameters, *e.g.*, the output name.
> GO ↵	to run the task.

The most direct flagging task is UVFLG, which puts commands into the flag table one at a time (or more than one when read from a disk text file). To use this task to flag channel 31 from 7 to 8 hours on the first day of observation from the second input (single-dish nomenclature) IF:

> TASK 'UVFLG' ; INP ↵	to review the inputs.
> INDI *n* ; GETN *ctn* ↵	to select the disk and catalog entry of the sorted data set.
> OUTFGVER 1 ↵	to select the desired flag table.
> TIMERANG 0, 7, 0, 0, 0, 8, 0, 0 ↵	to set the time range from 7 to 8 hours.
> BCHAN 31 ; ECHAN 31 ↵	to flag only channel 31.
> BIF 0 ; EIF 0 ↵	to do all *AIPS* IFs.
> ANTEN 2, 0 ; BASELIN 2, 0 ↵	to select "baseline" 2-2, the 2nd IF in 12m nomenclature.
> APARM 0 ↵	to ignore amplitude in flagging.
> OPCODE 'FLAG' ↵	to flag the data.
> REASON 'Bad channel' ↵	to store away a reason.
> INP ↵	to check the full set of adverbs.
> GO ↵	to add one line to the flag table, creating one if needed.

Multiple runs of UVFLG may be done to incorporate what you know about your data into the flagging table. Use PRTSD and the plot programs to help you find the bad data. If you have a long list of flagging commands, you may find it easier to use the INTEXT option of UVFLG to read in up to 100 flagging instructions at a time from a free-format text file.

The task CLIP is popular on interferometer data sets since it automatically flags all samples outside a specified flux range without interaction with the user. This blind flagging is often acceptable for interferometer data since each *uv* sample affects all image cells so that the damage done by a few remaining bad samples is attenuated by all the good samples. However, a bad sample in single-dish data affects only a few image cells and is hence not attenuated. Thus it is important to find and remove samples that are too small as well as those that are too large. For this reason, we do not recommend CLIP, but suggest that you look at your data and make more informed flagging decisions.

The best known of the interactive editing tasks is TVFLG (§ O.1.6). This task is not suitable for single-dish data since it displays multiple baselines along the horizontal axis. The data on these baselines are related in interferometry, but, in single dish, they are from separate feeds or polarizations and hence neither numerous nor necessarily related. For spectral-line single-dish data, the task SPFLG is an ideal task to examine your data and to edit portions if needed. SPFLG is a menu-driven, TV display editing task in which spectral channel varies along the horizontal axis of the TV display and time along the vertical. (The spectral channels for each interferometer IF are displayed on the horizontal axis, but single-dish data in *AIPS* has only 1 of this sort of IF.) The data may be displayed with as much or as little time averaging as desired and is very useful for examining your data even if you do not think that editing is needed. The latest version of SPFLG is documented in an *AIPS* Memo[1] which should be consulted for current details.

[1]Greisen, E. W. 2025, "Interactive flagging in *AIPS*", *AIPS* Memo 127, https://www.aips.nrao.edu/aipsdoc.html

To run SPFLG, type

> TASK 'SPFLG' ; INP ↵	to review the inputs.
> INDI *n* ; GETN *ctn* ↵	to select the disk and catalog entry of the sorted data set.
> FLAGVER 1 ↵	to select the use of a flag table *on the input data.*
> OUTFGVER 0 ↵	to write a new flag table including all flags applied to the input data.
> BCHAN 0 ; ECHAN 0 ↵	to view all spectral channels.
> DOCALIB FALSE ↵	to inhibit *interferometer* calibration of your data.
> IN2SEQ 0 ; DOCAT FALSE ↵	to create a new, but temporary "master file" each time.
> ANTEN 0 ; BASEL 0 ↵	to include all "baselines."
> DPARM 0, 1, 0, 0, 0, 0.1 ↵	to include autocorrelation data and to set the fundamental interval used to average data into the master file. The defaults for these parameters are not suitable for single-dish data. The other DPARM parameters may be ignored since they can be altered during the interactive session.
> INP ↵	to review the inputs.
> GO ↵	to begin the interactive display and editing.

The task will then read your data to determine which times occur in the included portions (you may set TIMERANG, restrict autocorrelations, etc.) and then construct a master grid file with spectral channel as the first axis, pseudo-regular times on the second axis (gaps are mostly suppressed), and, if needed, baseline number on the third axis. SPFLG tells you the size of the resulting file, *e.g.*, `SPFLG1:  Basic UV image is 128 14079 pixels in X,Y (Ch,T).`

At this point, SPFLG selects an initial *display* smoothing time long enough to fit all of the master grid onto your TV window. It then averages the data to this interval and creates a display not unlike that seen in Figure 10.3. Move the TV cursor to any menu item (it will change color to show which has been selected) and press button D for on-line help information or press buttons A, B, or C to select the operation. Normally, you will probably begin by reducing the smoothing time (ENTER SMOOTH TIME menu option followed by typing in the new smoothing multiple on your AIPS window). Note that the display does not change other than to add an asterisk after the smoothing time to indicate that that will change on the next image load. This behavior is to allow you to alter a number of choices before doing the potentially expensive TV display. In this typical example, you would either ENTER BLC and ENTER TRC by hand and finally LOAD the sub-image or you can do this interactively with SET WINDOW + LOAD. You may examine data values (like CURVAL) and flag data with the options in the fourth column. Flagged data are removed from the display. You may review the flags you have prepared, undo any that you dislike, re-apply the remaining ones to make sure the display is correct, and modify the appearance of the display with the options in the first column. The image may be shown in zoom only during editing in order to give you greater accuracy in examining the data values and locations. If you are doing some time smoothing within SPFLG, the DISPLAY RMS option allows you to view images of the rms rather than the value of the time average. Such a display allows you to find excessively noisy portions of the data quickly.

Finally, when you are done, select EXIT. If you have prepared any flagging commands, SPFLG will ask you if you wish to enter them into your input data set. Answer yes unless you want to discard them or you have set DOCAT TRUE to catalog the master file in order to use it for multiple sessions. If you set OUTFGVER to zero, then the flag commands are put into a new flag table which can be deleted later if you wish.

SPFLG is not useful on continuum data; the interactive editor of choice for such data used to be the task IBLED, but is now EDITR. These tasks are also useful for spectral-line data in that they can display the average (and rms) of a selected range of channels. The spectral averaging should let you see more subtle level problems than can be seen on individual channels (*i.e.*, in SPFLG). EDITR is a menu-driven, TV display editing task, but it does not use grey-scales to show data values. Instead, it plots time on the horizontal axis and data value on the vertical axis. The full data set for the chosen baseline is displayed initially in

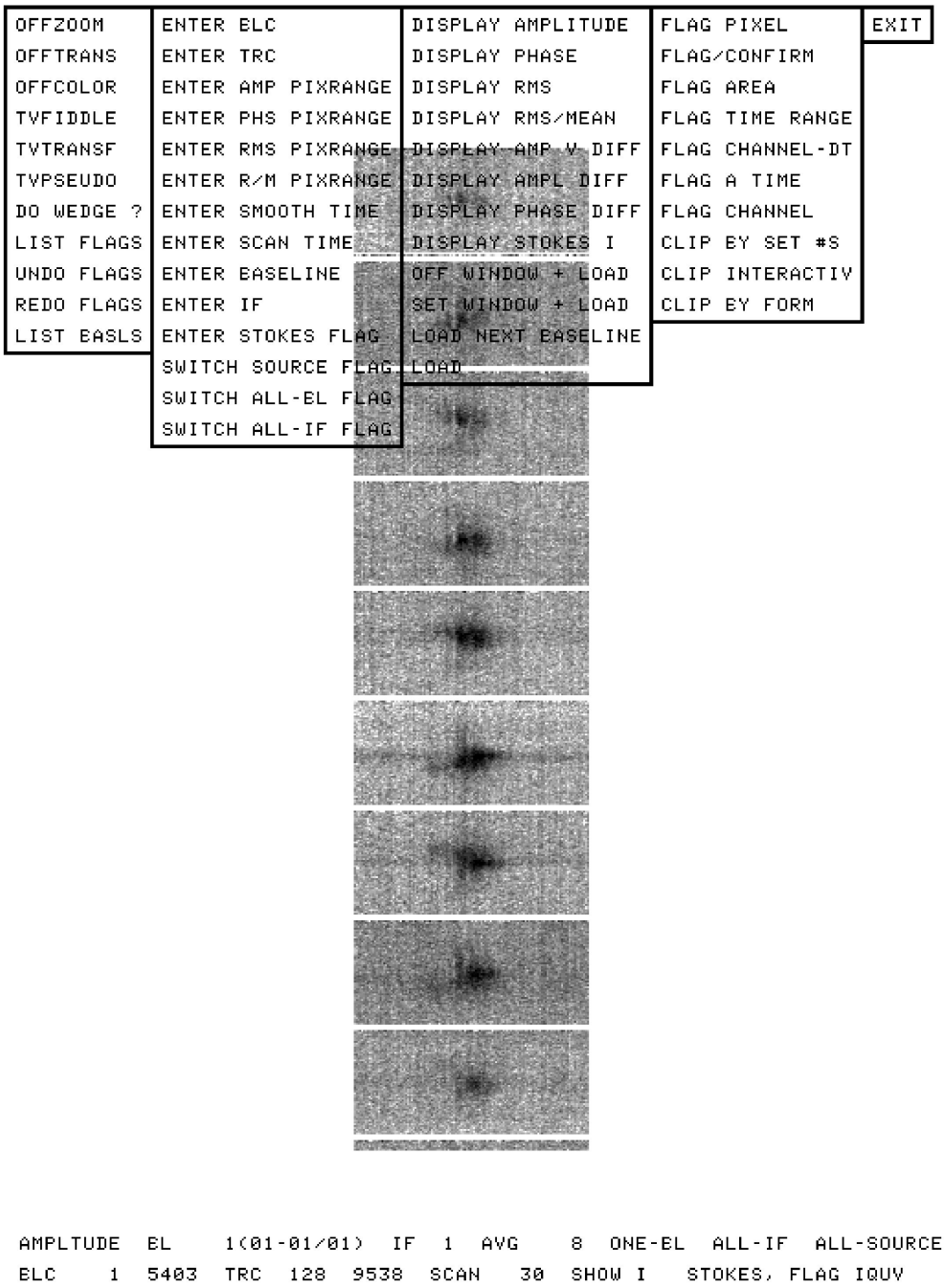


Figure 10.3: A display of a sample TV screen from SPFLG on single-dish data, made using the *AIPS* task TVCPS to produce a negative black-and-white display. The SPFLG menu (in the boxes) and status lines (at the bottom) are displayed in a graphics plane which is normally colored light green. The data are grey scales in a TV memory and may be enhanced in black-and-white or pseudo-colored. The data actually displayed range in intensity from -1.7 to 5.2 Kelvins (as stated during the image loading) and have been averaged to 0.8 seconds. The entire master grid contains 14079 times, but the current window includes only times 5403 through 9538. Flag commands generated at the moment illustrated will flag all source names, all IFs (in the *AIPS* sense), only the displayed baseline, and all Stokes. *Note that the menu displayed is now out of date, more options are available in later versions. See AIPS Memo 127.*

a potentially crowded area at the bottom of the TV window. This area is available for editing. If DOTWO is true, then it also displays above the edit area a second observable (initially the difference between the amplitude and a running mean of the amplitude) for the primary baseline. EDITR allows you to display up to ten other "baselines" (*e.g.*, 12m-antenna IFs) in frames above the active editing frame. These should speed the process of editing and guide you in the choice of flagging one or all baselines at the time of the observation. A smaller time range or window into these full data sets may be selected interactively to enable more detailed editing. Be sure to set SOLINT to specify an appropriate averaging interval. Unlike SPFLG, no further time averaging is possible. The menu options allow you to work your way through all of your data, selecting time windows and baselines as desired. Consult § 5.5.2 for more details about EDITR.

EDITR has the ability to display a second data set for reference in parallel with the one being edited. This option is likely to prove useful for beam-switched continuum observations. Select one of the beam throws for editing and the other for reference display. Then, if editing is required, reverse the roles. It may also be useful to look at your beam-switched data in its differenced form. The task DIFUV may be used to difference the plus and minus throws, followed by EDITR (or any other *AIPS* uv-data task) to look at the differences. Be sure to tell DIFUV that the time difference between the plus and the minus beam throws should not be considered significant, *i.e.*, SOLINT = 1 / 8 / 60 or a little bit more to avoid round-off effects.

10.2.3 Using CSCOR and SDCAL to calibrate your data

The current calibration routines for single-dish data in *AIPS* are fairly rudimentary. The concept is similar to that used for interferometers. Corrections are developed in an extension table (called CS in single-dish, CL in interferometry) which can be applied to the data by some tasks. In particular, the single-dish tasks PRTSD and SDGRD are able to apply the CS table to the data without modifying the data as stored on disk. They do this using the DOCAL and GAINUSE adverbs. Other *uv* tasks, designed primarily for interferometry, also use these adverbs, but do not understand or apply CS tables. For such tasks, you should carefully turn off the calibration option. If you do not, such tasks will fail.

There are two tasks which can create CS tables: SDTUV discussed above and INDXR. To use the latter, enter

> TASK 'INDXR' ; INP ↵	to review the inputs.
> INDI *n* ; GETN *ctn* ↵	to select the disk and catalog entry of the data set.
> CPARM $T_1, T_2, \Delta T$ ↵	to set the largest gap (T_1) and longest scan (T_2) times expected in the data set (for the index table) and to set the time interval (ΔT) in the CS table, all in minutes.
> GO ↵	to run the task to create an index (NX) and a calibration (CS) table attached to the main data set.

Note that this task requires the data to be in time order and expects an antenna (AN) table. You may set CPARM(5) to the maximum antenna number (beam number) in your data set and, with a few grumbles, INDXR will still create and initialize a CS table when you do not have an antenna table.

At this writing, the CS table may be used to correct the recorded right ascension and declination (*i.e.*, the pointing) and to correct the amplitudes for atmospheric opacity and other gain as a function of zenith angle effects. To add an atmospheric opacity correction to the CS table produced by INDXR, type:

> TASK 'CSCOR' ; INP ↵	to review the inputs.
> TIMERAN 0 ; ANTENN 0 ↵	to do all times and antennas.
> GAINVER 1 ; GAINUSE 2 ↵	to modify the base table, producing a new table.
> OPCODE 'OPAC' ↵	to do the opacity correction.
> BPARM O_z, 0 ↵	to specify the zenith opacity in nepers.
> GO ↵	to run the task.

Note that CSCOR only writes those records in the output file that you have selected via TIMERANG, ANTENNAS, etc. To make a new CS table to work for the full data set, you should first use TACOP to write the new table and then set GAINVER and GAINUSE to both point at the new table. CSCOR needs to compute the zenith angle and therefore needs to have an antennas file. If your data set does not have one, you may give the antenna longitude and latitude in the CPARM adverb. The other operations offered by CSCOR are GAIN, PTRA, and PTDC which apply as second-order polynomial functions of zenith angle corrections to the gain, right ascension, and declination, respectively. The format of the CS table allows for an additive flux correction as well. There are no tasks at this time to determine such a correction.

The basic single-dish tasks PRTSD and SDGRD can apply the CS table to the data as they read them in. Other *uv* tasks which are more directed toward interferometry data cannot do this. If you need to use such tasks with corrected data, then you must apply the corrections with SDCAL and write a new "calibrated" data set. To do this:

> TASK 'SDCAL' ; INP ↵	to review the inputs.
> INDI *n* ; GETN *ctn* ↵	to select the disk and catalog entry of the data set.
> TIMERA 0 ; FLAGVER 1 ↵	to do all times and apply any flagging.
> BCHAN 1 ; ECHAN 0 ↵	to get all channels.
> DOCAL TRUE ; GAINUSE 0 ↵	to apply the highest numbered CS table.
> APARM 0 ↵	to do no averaging of spectral channels.
> GO ↵	to run the task.

The output file from SDCAL can then be fed to UVPLT, SPFLG, or any other *uv*-data task including of course PRTSD and SDGRD.

10.2.4 Using SDLSF and SDVEL to correct your spectral-line data

It may be convenient to remove a spectral baseline from each sample before the imaging step. Doing so may allow you to skip the removal of a spectral baseline from the image cubes (as described in § 10.4.1). To do this, type:

> TASK 'SDLSF' ; INP ↵	to review the inputs.
> INDI *n* ; GETN *ctn* ↵	to select the disk and catalog entry of the data set.
> NCOUNT 1 ↵	to solve for a slope as well as a constant in the baselines.
> DOALL 1 ↵	to fit a single baseline to all samples taken at a particular time. This is useful for single-beam, multi-polarization data, but, for multi-beam data, it is found that instrumental problems dominate weather and require DOALL = -1 ↵ instead.
> DOOUT -1 ↵	to avoid writing a continuum data set.
> FLUX 0 ; CUTOFF 0 ↵	to write all data with no flagging.
> CHANSEL $s_1, e_1, i_1, s_2, e_2, i_2 \ldots$ ↵	to use every i_1 channel from s_1 through e_1, every i_2 channel from s_2 through e_2, and so forth to fit the baseline. Be sure to avoid dubious channels, if any, at the ends and any channels with real line signal. It is important to have regions at both ends of the spectrum to fit the slope.
> INP ↵	to review the inputs.
> GO ↵	to run the task.

You may, and probably should, use FLUX and CUTOFF to flag those data having excessive noise or excessive signals in individual channels. These "excesses" are measured only in the channels selected by CHANSEL for fitting the baseline.

If you have observed a wide field with relatively narrow spectral channels, there is an effect which you

should consider. The "velocity" corresponding to a particular frequency of observation depends on the velocity definition (*e.g.*, LSR or heliocentric), the direction at which the telescope pointed, the time of year, the time of day, and the location of the telescope. Most telescopes adjust the observing frequency to achieve the desired velocity for some reference time and position and many adjust the frequency periodically to account for time changes. However, few, if any, can adjust the observing frequency for every pointing direction and time in a rapidly scanned on-the-fly observing mode. The 12m telescope now sets the frequency once per image with respect to the reference coordinate (usually the image center). In this mode, the maximum velocity error in a 2 degree by 2 degree image is about 1.16 km/s (in LSR velocities) and 0.79 km/s (heliocentric). Since mm lines are often narrow, this can be a significant effect. Fortunately, single-dish OTF data may be fully corrected for this effect so long as your spectra are fully sampled in frequency. The task SDVEL shifts each spectrum so that the reference channel has the reference velocity for its pointing position. The DPARM adverb array is used to tell the task how the telescope set reference velocities and to ask the task to report any excessive shifts and even flag data having really excessive shifts. The latter are to detect and/or remove times in which the telescope pointing was significantly in error (*i.e.*, high winds). DPARM(1) should be set to 0 for 12m data taken after 5 May 1997 and to 2 for data taken before that date. The task VTEST was written to help you evaluate the magnitude of this effect.

10.2.5 Using SDMOD and BSMOD to model your data

It is sometimes useful to replace your actual data with a source model or, if your continuum levels are well calibrated, to add or subtract a model from your data. The task to do this is called SDMOD and allows up to four spatially elliptical Gaussians (or an image) to replace the data, or to be added to the data, with either a Gaussian or no frequency dependence. When the data are replaced, a random noise may also be added. SDMOD has options for modeling beam-switched continuum data (set BPARM(1) = 1) as well as for spectral-line data. For example, to see what a modestly noisy point source at the origin would look like after all of the imaging steps:

> TASK 'SDMOD' ; INP ↵	to review the inputs.
> INDI *n* ; GETN *ctn* ↵	to select the disk and catalog entry of the data set.
> BCHAN *n* ; ECHAN *n* ↵	to get one channel only.
> NGAUSS 1 ; APARM 0 ↵	to get one Gaussian with no frequency dependence.
> GWIDTH 0 ; GPOS 0 ↵	to do a point source (convolved with the single-dish beamwidth in the header) at the coordinate center.
> GMAX 1, 0 ; FLUX 0.05 ↵	to do a 1 K object with rms noise of 0.05 K.
> GO ↵	to run the task.

The output file from SDMOD can then be fed to SDGRD, BSGRD, or any other appropriate task as if it were regular data. The input model is convolved with the single-dish beamwidth given in the *uv* data header before being used to replace or add to the input data. The history file will show in detail what was done.

Beam-switched observations may be modeled with task BSMOD. No input data set is needed. Instead two regular grids of switched data are constructed from a specified model plus noise and a variety of instrumental defects.

10.3 Imaging single-dish data in *AIPS*

10.3.1 Normal single-dish imaging

The process of imaging in single-dish is a process of convolving the "randomly distributed" observations with some convolving function and then resampling the result on a regular image grid. Tasks SDGRD and SDIMG combine the data calibration, selection, projection, sorting, and gridding in one task capable of imaging all spectral channels into one output data "cube." They are relatively easy to run, but selecting the correct input adverb values is more difficult. Choose SDGRD for most single-dish applications; SDIMG is very similar but can handle larger output images at the cost of making a sorted copy of the entire input data set (which can be very large). Type:

> TASK 'SDGRD' ; INP ↵	to review the inputs.
> INDI *n* ; GETN *ctn* ↵	to select the disk and catalog entry of the data set.
> TIMERA 0 ; FLAGVER 1 ↵	to do all times and apply any flagging.
> BCHAN 1 ; ECHAN 0 ↵	to get all channels.
> DOCAL FALSE ↵	to apply no calibration.
> OPTYPE '-GLS' ↵	to make the image on a "global sinusoidal" kind of projection.
> APARM 0 ↵	to use the observed right ascension and declination given in the header as the center of the image. For concatenated data sets, use APARM to specify a more appropriate center.
> REWEIGHT 0, 0.05 ↵	to have an "interpolated" or best-estimate image for output, cutting off any cells with convolved weight < 0.05 of the maximum convolved weight.
> CELLSIZE c ↵	to set the image cells to be c arc seconds on a side.
> IMSIZE N_x , N_y ↵	to make the image of each channel be N_x by N_y pixels centered on the coordinate selected by APARM.
> XTYPE 16 ; XPARM 0,0,0,0,50 ↵	to select convolution function type 16 (a round Bessel function times Gaussian) with default parameters and 50 samples of the function per pixel. The default of 20 samples/cell is probably adequate.
> INP ↵	to review the inputs.
> GO ↵	to run the task.

SDGRD begins by reading the data selecting only those samples which will fit fully on the image grid. It reports how many were read and how many selected. If you have made the image too small, with IMSIZE or CELLSIZE, then data will be discarded. Use PRTSD with a substantial XINC to determine the full spatial distribution of your data. It does not hurt to have the output image be a bit bigger than absolutely necessary. If you are uncertain about the parameters to use, try running SDGRD on a single channel to begin with since it will be much faster.

A number of these parameters require more discussion. REWEIGHT(1) selects the type of output image. The data are multiplied by their weights (which depend on the system temperature), convolved by the sampled convolving function and then summed at each image pixel. REWEIGHT(1) = 1 selects the result, which is not calibrated in any way since its scaling depends on the scaling of the data weights and the convolving function and on the distribution of data. While the program "grids" the actual data it also does the same process on the data replaced by 1.0. That result, the convolved weights may be obtained with REWEIGHT(1) = 2. The most meaningful image, which is obtained with REWEIGHT(1) = 0, is the ratio of the former to the latter. This is the interpolated or best-estimate image and will be similar to the convolved image in well-sampled regions except for having retained the calibration. REWEIGHT(1) = 3 tells SDGRD to compute an image of the expected noise (actually $1/\sigma^2$) in the output image of type 0; see WTSUM below (§ 10.4.2) for its use.

REWEIGHT(2) controls which pixels are retained in the output image and which are blanked by specifying a cutoff as a fraction of the maximum convolved weight. It is important to blank pixels which are either simple extrapolations of single samples or, worse, extrapolations of only a couple noisy samples. In the latter case, it is possible to get very large image values. Thus, if the output is $(W_1D_1 + W_2D_2)/(W_1 + W_2)$ where the D's are data and the W's are convolved weights and if, say $W_1 = 0.1001, D_1 = 1.0, W_2 = -0.1000, D_2 = -1.0$, then the output would be 2001. Such large and erroneous values will be obvious, but will confuse software which must deal with the whole image and will also confuse people to whom you may show the image. In simple cases, in which all data have roughly the same data weights (system temperatures), setting REWEIGHT(2) = 0.2 or even more is probably wise. However, if some portions of the data have significantly lower weights than others, then you may have to set a lower value in order to keep the low-weight regions from being completely blanked.

The choices of CELLSIZE and the widths of the convolving function are related to the spatial resolution inherent in your data, *i.e.*, to the single-dish beamwidth. If the pixels and function are too small, then data samples which are really from the same point in the sky will appear as if different in the output image. If, however, they are too large, then too much data will be smoothed together and spatial resolution will be lost. The latter may be desirable to improve signal-to-noise, but image smoothing can be done at a later stage as well. You may wish to experiment with these parameters, but it is usually good to start with a CELLSIZE about one-third of the beamwidth (fwhm) of your telescope. The default parameters (XPARMs) of all convolving functions may be used with this cell size. You may vary these parameters in units of cells or in units of arc seconds; enter HELP UV*n*TYPE C$_R$ to look at the parameters for type n ($n = 1$ through 6). If you give XTYPE = $n + 10$, then you get a round rather than square function which is perhaps better suited to this type of data. If you wish to change the cell size, but retain the same convolving function in angular measure on the sky, you may give XTYPE < 0 and specify the XPARMs in arc seconds rather than cells.

The choice of convolving function affects the noise levels and actual spatial resolution in the output image. In effect, the Fourier transform of the convolving function acts to modify the illumination pattern of the feed horn onto the aperture. Figure 10.4 shows slices through the Fourier transforms of six of the available convolving functions. The ideal function would be flat all the way across and then suddenly zero at the edges. Type 14 is the widest, but has a deep dip in the middle. This leaves out the center portion of the dish and illuminates the outer portions, effectively improving the spatial resolution of the image over that of the normal telescope, but with a noticeable loss of signal-to-noise ratio. The spheroidal functions, on the other hand, illuminate the center fully and leave out the outer portions. This degrades the spatial resolution, but noticeably improves the noise levels. Types 4 and 16 seem to be the best compromise. Type 16 is preferred since it is zero at the edges. Round functions require more computer memory than square ones, so type 4 would be preferred on computers with small memories.

Images may be built up from observations taken at significantly different times. The simplest way to do this is to concatenate the two "*uv*" data sets on disk with OTFUV or DBCON (§ 10.1.1.2) and then use SDGRD once to make the image. Some single-dish data sets are so large — or the time interval so great — that this is not practical. SDGRD combines observations taking into account the data weights which are based on the measured system temperatures. You can get the same weighted averaging in the image plane if you first compute a "weight" image and then use the task WTSUM to do the averaging. To get a weight image:

> TGET SDGRD C$_R$	to get the inputs used for the actual image cube.
> REWEIGHT(1) 3 C$_R$	to get the weight image which is proportional to $1/\sigma^2$ expected from the actual gridding done on the data whose weights are assumed proportional to their $1/\sigma^2$.
> BCHAN n ; ECHAN BCHAN C$_R$	to image a single channel when there is no channel-dependent data weights and flagging.
> GO C$_R$	to get the weight image.

See § 10.4.2 for details about WTSUM.

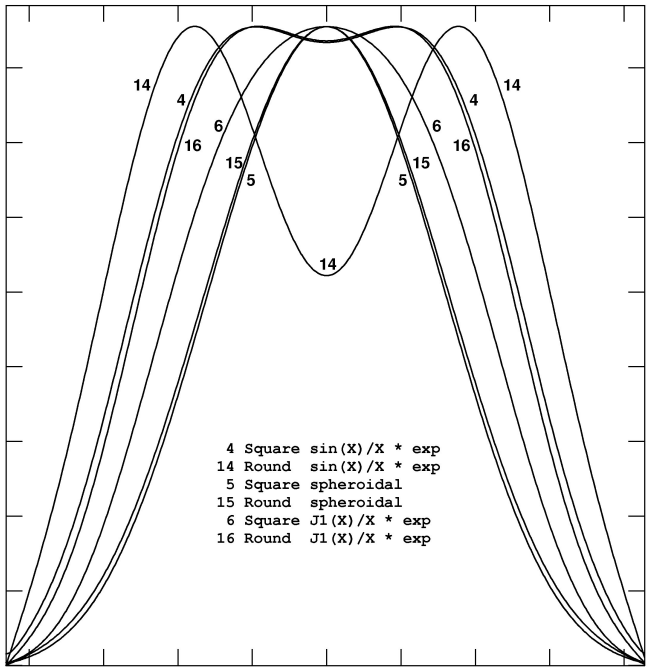


Figure 10.4: Slices through the Fourier transforms of six convolving functions using the XTYPE numbers shown on the plot with default values of the XPARMs.

10.3.2 Beam-switched continuum imaging

The construction of images from beam-switched on-the-fly continuum observations is more properly a research question than one of production software. Observers in this mode should be aware that the optimal methods of data reduction are probably not yet known and that the methods currently provided require the user to determine three critical correction parameters. In this mode, the telescope is moved in a raster of offsets in azimuth and elevation with respect to the central coordinate. The beam is switched rapidly from a "plus" position to a "minus" position at constant elevation. On the 12m, there are four plus samples and four minus samples taken each second, all taken while the telescope is being driven rapidly in azimuth at a constant (relative to the central source) elevation. In principle, each pair of plus and minus points contain the same instrumental bias but different celestial signals. It is then the job of the software to disentangle the time variable bias from the two beams' estimates of the sky brightness.

A technique for doing the disentangling was first described by Emerson, Klein, and Haslam (Astronomy and Astrophysics, 76, 92–105, 1979). The plus and minus samples are differenced removing the instrumental bias and creating two images of the sky, one positive and one negative. Problems arise because the two images potentially can overlap and because, in the OTF mode of observing, the telescope positioning is not exactly along rows of the output image and the relative positioning of the plus and minus beams varies both due to the wobbles in the telescope pointing and due to the reversing of the direction of telescope movement. The Emerson *et al.* technique involves a convolution of each row in the differenced image with a function which is a set of positive and negative delta functions (or $\frac{\sin x}{x}$ functions when the total beam throw is not an integer number of image cells). It turns out that the problem of image overlap is largely solved by this technique. Unfortunately, differences in the position of the plus and minus beam with respect to the source and to the image cells appear to limit the quality of the images produced with this technique.

The principal task used to produce images from data is called BSGRD. It makes two images from the two "uv" data sets written by OTFBS, gridding each sample at the coordinate at which it was observed

(neglecting the throw but not the telescope movement between plus and minus). If the beam throw was not exactly along constant elevation, it then shifts the two images. Then it applies the Emerson *et al.* technique, fitting and removing baselines, differencing the two images, and convolving the difference image with an appropriate $\frac{\sin x}{x}$ function. Finally, BSGRD re-grids the data from relative azimuth-elevation coordinates onto a grid in normal celestial coordinates. This task is a combination of four tasks, SDGRD described above to make the images, OGEOM to do the rotation correction, BSCOR to apply the Emerson *et al.* technique, and BSGEO to re-grid the data onto normal celestial coordinates.

To use BSGRD, type

> TASK 'BSGRD' ; INP ↵	to review the inputs.
> INDI *n* ; GETN *ctn* ↵	to select the disk and catalog entry of the data set. Note that the class name is assumed to have a plus sign in the sixth character for the plus throw data set and a minus sign in that character for the minus throw data set.
> TIMERA 0 ; FLAGVER 1 ↵	to do all times and apply any flagging.
> DOCAL FALSE ↵	to apply no calibration.
> OPTYPE '-GLS' ↵	to make the image on a "global sinusoidal" kind of projection.
> APARM 0 ↵	to use the observed right ascension and declination given in the header as the center of the image. For concatenated data sets, use APARM to specify a more appropriate center.
> REWEIGHT 0, 0.05 ↵	to have an "interpolated" or best-estimate image for output, cutting off any cells with convolved weight < 0.05 of the maximum convolved weight.
> CELLSIZE *c* ↵	to set the image cells to be *c* arc seconds on a side.
> IMSIZE N_x , N_y ↵	to make the image of each throw be N_x by N_y pixels centered on the coordinate selected by APARM.
> XTYPE 16 ; XPARM 0,0,0,0,50 ↵	to select convolution function type 16 (a round Bessel function times Gaussian) with default parameters and 50 samples of the function per pixel. The same function is used in both convolutions.
> FACTOR f ↵	to multiply the recorded throw lengths by f in doing the Emerson *et al.* correction.
> ROTATE ρ ↵	to correct the throws for being ρ degrees off from horizontal.
> DPARM $1, 1, x_1, x_2, x_3, x_4$ ↵	to specify that the two beams have the same relative amplitude and to give the pixel numbers to be used to fit baselines in *both* images.
> ORDER 1 ↵	to fit a slope as well as a constant in the horizontal baseline in each row.
> DOCAT -1 ↵	to delete the intermediate images created by BSGRD.
> INP ↵	to review the inputs.
> GO ↵	to run the task.

BSGRD takes three correction parameters which you must supply: the throw length error FACTOR, the throw angle error ROTATE, and the relative beam gain error DPARM(1). To estimate these, you will need data on a relatively strong point source. Use SDGRD to make an image of each throw of these data, setting ROTATE = 0 since rotation *must* be done later and setting ECHAN = 1 to eliminate the coordinate information which is confusing to MCUBE and used only by BSGEO. The tasks IMFIT and/or JMFIT (§ 7.5.2) may be useful in fitting the location and peak of the two beams. Since there is likely to be a significant offset from zero in these images, be sure to fit for the offset using a second component of CTYPE = 4. For reasons that are not clear, these tasks may not provide sufficiently accurate positions. Another approach then is to take the two

images produced by SDGRD and then run OGEOM and BSCOR for a range of rotations and factors. Find the image that is most pleasing and put its parameters into BSGRD for the program source. Of course, it is not clear that these correction factors are constant with time or pointing, so this could all be bologna.

For example,

> TASK 'OGEOM' ; INP ↵	to review the inputs.
> INDI n ; GETN ctn_+ ↵	to select the disk and catalog entry of the plus image.
> APARM 0 ↵	to do no shifts or re-scaling.
> DOWAIT 1 ↵	to wait for a task to finish before resuming AIPS.
> OUTCLA 'OGEOM+' ↵	to set the output class to show the throw sign.
> FOR APARM(3) = -2 , 2.01 BY 0.1 ; GO; END ↵	to produce 41 plus images each with a slightly different rotation.
> GETN ctn_- ; OUTCLA 'OGEOM-' ↵	to select the minus image as input and specify the output class.
> FOR APARM(3) = -2 , 2.01 BY 0.1 ; GO; END ↵	to produce 41 minus images each with a slightly different rotation.

Then apply the Emerson *et al.* corrections to each of the 41 with

> TASK 'BSCOR' ; INP ↵	to review the inputs.
> INDI n ; GETN ctn_+ ↵	to select the disk and catalog entry of one of the rotated plus images.
> IN2DI n ; GET2N ctn_- ↵	to select the disk and catalog entry of one of the rotated minus images.
> FACTOR f ↵	to multiply the recorded throw lengths by f in doing the Emerson *et al.* correction. Use 1.0 as an initial guess.
> DPARM $1, 1, x_1, x_2, x_3, x_4$ ↵	to specify that the two beams have the same relative amplitude and to give the pixel numbers to be used to fit baselines in *both* images. The choice of the x_n is significant.
> ORDER 1 ↵	to fit a slope as well as a constant in the horizontal baseline in each row.
> FOR INSEQ = 1 : 41; IN2SEQ = INSEQ ; GO ; END ↵	to produce 41 "corrected" images.

It is convenient to look at the images with tools such as TVMOVIE (§ 8.5.4) and KNTR (§ 10.4.5). To build the "cube", use MCUBE as:

> TASK 'MCUBE' ; INP ↵	to review the inputs.
> INDI n ; GETN ctn ↵	to select the disk and catalog entry of the first of the corrected images.
> IN2SEQ 41 ; IN3SEQ 1 ↵	to set the sequence number loop limit and increment.
> AXREF 0 ; AX2REF 41 ; NPOINTS 41 ↵	to set the locations of the images in the cube explicitly.
> DOALIGN -2 ↵	to have MCUBE ignore the differing image rotations.
> DOWAIT -1 ; GO ↵	to resume normal task functioning and to build the data cube.

Examine the cube to find the "best" plane and use the rotation of that plane to run similar tests varying the throw length correction factor. One of these cubes, testing rotation, is illustrated in Figure 10.5.

The determination of throw length is similar:

> TASK 'BSCOR' ; INP ↵	to review the inputs.
> INDI n ; GETN ctn_+ ↵	to select the disk and catalog entry of the plus image at the best rotation.
> IN2DI n ; GET2N ctn_- ↵	to select the disk and catalog entry of one of the corresponding rotated minus image.

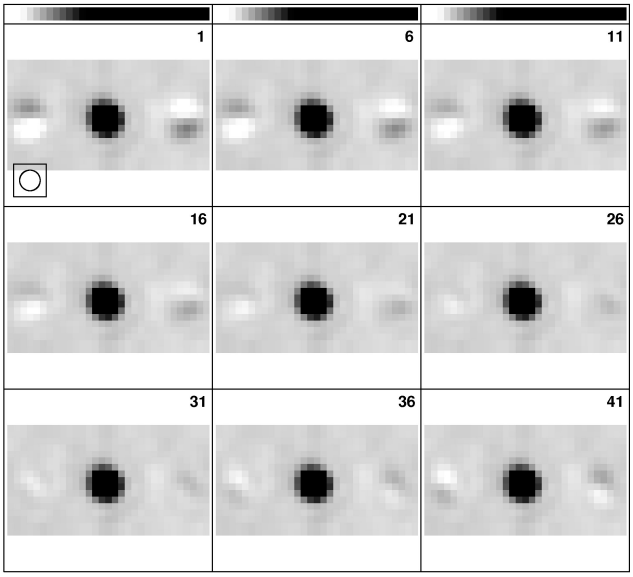


Figure 10.5: Images at selected rotations from 2.0 to -2.0 by -0.5 degrees. Rotations between -0.5 and -1.0 appear to minimize the artifacts due to the incomplete cancellation of the plus and minus beams.

> OUTNA 'ROTATE TEST' ↵ — to assign a new output name.

> DPARM $1, 1, x_1, x_2, x_3, x_4$ ↵ — to specify that the two beams have the same relative amplitude and to give the pixel numbers to be used to fit baselines in *both* images. The choice of the x_n is significant.

> ORDER 1 ↵ — to fit a slope as well as a constant in the horizontal baseline in each row.

> DOWAIT 1 ↵ — to run the task in wait mode.

> FOR FACTOR = 0.9 ; 1.101 BY 0.005; GO ; END ↵ — to produce 41 "corrected" images.

It is convenient to look at the images with tools such as TVMOVIE (§ 8.5.4) and KNTR (§ 10.4.5). To build the "cube", use MCUBE as:

> TASK 'MCUBE' ; INP ↵ — to review the inputs.

> INDI *n* ; GETN *ctn* ↵ — to select the disk and catalog entry of the first of the new corrected images.

> IN2SEQ 41 ; IN3SEQ 1 ↵ — to set the sequence number loop limit and increment.

> AXREF 0 ; AX2REF 41 ; NPOINTS 41 ↵ — to set the locations of the images in the cube explicitly.

> DOWAIT -1 ; GO ↵ — to resume normal task functioning and to build the data cube.

Examine the cube to find the "best" plane and use the scaling factor of that plane in later imaging. One of these cubes, testing throw length, is illustrated in Figure 10.6.

BSGRD is in fact a deconvolution algorithm to remove the plus-minus beam from the difference image. An experimental Clean algorithm has been made available in BSCLN. Although initial tests seemed promising, it appears to converge to the EKH solution after very many iterations and to have systematic problems before that point. The one-dimensional display task BSTST will allow you to evaluate and compare the two algorithms on model (one-dimensional) data.

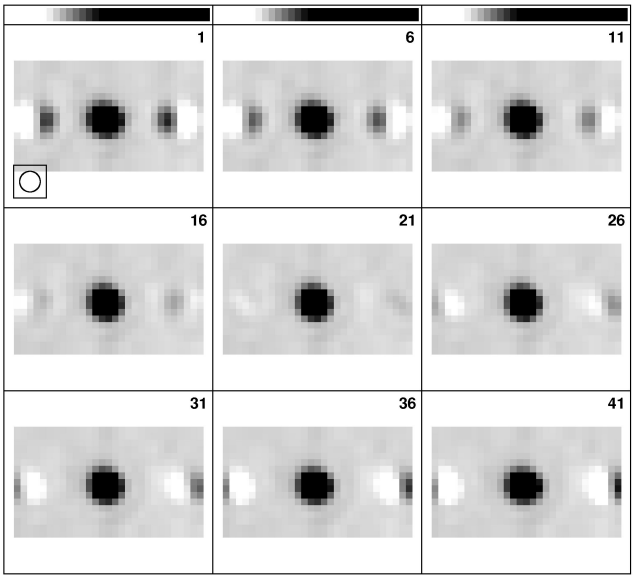


Figure 10.6: Images at selected beam throw corrections from 0.9 to 1.1 by -0.5. Note that rotation errors cause vertical separations of the plus and minus images while throw length errors cause horizontal separations (and hence incomplete cancellation) of the beams.

10.4 Analysis and display of single-dish data

The analysis and display of images produced from single-dish data are not, in general, different from those produced by interferometers. See Chapter 6 for a discussion of display tools, Chapter 7 for a variety of analysis tasks, and § 8.5 and § 8.6 for spectral-line analysis and display. Some matters of particular interest to single-dish users will be discussed below.

10.4.1 Spectral baseline removal

As the SPFLG display in § 10.2.2 shows, one of the first things most users will want to do is remove a spectral baseline at each pixel in the their image. This is frequently done with SDLSF (§ 10.2.4). To do this in the image plane, you must first transpose the data cube to make the frequency axis be first:

> TASK 'TRANS' ; INP ↵	to review the task's parameters.
> INDI n_1 ; GETN ctn_1 ↵	to select the input image from disk n_1 catalog slot ctn_1.
> TRANSCOD '312' ; OUTCL 'VXY' ↵	to move the frequency axis from $3^{\rm rd}$ to $1^{\rm st}$.
> GO ↵	to transpose the cube.

You must also determine, using this input image if needed, which spectral channels are completely free of real emission or absorption. TVMOVIE is often useful; see § 8.5.4. Then:

> TASK 'IMLIN' ; INP ↵	to review the task's parameters.
> INDI n_2 ; GETN ctn_2 ↵	to select the input image (output from TRANS) from disk n_2 catalog slot ctn_2.
> ORDER 1 ↵	to subtract linear baselines; up to $4^{\rm th}$ are allowed.
> NBOXES n ↵	to select n contiguous regions along the spectral axis to be used in fitting the channels.

> BOX $c_{11}, c_{12}, c_{21}, c_{22}, \ldots c_{n1}, c_{n2}$ ↵ — to use spectral channels c_{11} – c_{12}, c_{21} – c_{22}, up to c_{n1} – c_{n2} to fit the baselines at each pixel.

> INP ↵ — to review the inputs.

> GO ↵ — to fit the baselines, writing a new data cube.

It is sometimes useful to specify DOOUT TRUE to obtain images of the fit parameters and of their uncertainties. The uncertainty in the DC offset is a good measure of the uncertainty in the image.

The output from IMLIN is the baseline-corrected image in the familiar position-velocity form, with a third axis giving multiple positions on the second celestial coordinate. To go back to sky images as a function of frequency:

> TASK 'TRANS' ; INP ↵ — to review the task's parameters.

> INDI n_3 ; GETN ctn_3 ↵ — to select the input image (output from IMLIN) from disk n_3 catalog slot ctn_3.

> TRANSCOD '231' ; OUTCL 'XYV' ↵ — to move the frequency axis from $1^{\rm st}$ to $3^{\rm rd}$.

> GO ↵ — to transpose the cube back again.

10.4.2 Using WTSUM and BSAVG to combine images

To do a weighted average of multiple images of the same field, be sure to make all images with the same geometry type, the same cell size, and the same center coordinate. If you have two images,

> TASK 'WTSUM' ; INP ↵ — to review the inputs.

> INDI n_1 ; GETN ctn_1 ↵ — to select the first input image from disk n_1 catalog slot ctn_1.

> INDI n_2 ; GET2N ctn_2 ↵ — to select the second input image from disk n_1 catalog slot ctn_2.

> INDI n_3 ; GET3N ctn_3 ↵ — to select the first weight image from disk n_3 catalog slot ctn_3.

> INDI n_4 ; GET4N ctn_4 ↵ — to select the second weight image from disk n_4 catalog slot ctn_4.

> DOINVER FALSE ↵ — to state that the weight images are weights rather than rms's.

> GO ↵ — to compute an averaged image cube and a new weight image.

The weight images can be either a single plane or a cube that matches the corresponding image cube. All must be on the same spectral and celestial coordinate system.

If you have more than two images of the same field, then all images must have the same name parameters, differing only by having consecutive sequence numbers. All weight images must have the same name parameters with corresponding consecutive sequence numbers. The verb RENAME may be used to correct problems in naming. Then

> TASK 'WTSUM' ; INP ↵ — to review the inputs.

> INDI n_1 ; GETN ctn_1 ↵ — to select the first input image from disk n_1 catalog slot ctn_1.

> CLR2NAME ; IN2SEQ m_2 ↵ — to select the looping mode and set the highest image sequence number.

> INDI n_3 ; GET3N ctn_3 ↵ — to select the first weight image from disk n_3 catalog slot ctn_3.

> CLR4NAME ↵ — to clear the unused fourth name set.

> DOINVER FALSE ↵ — to state that the weight images are weights rather than rms's.

> GO ↵ — to compute an averaged image cube and a new weight image.

If m_1 is the sequence number of the first image (in ctn_1) and w_1 is the sequence number of the first weight image (in ctn_3), then images of sequence numbers m_1 through m_2 will be weighted with corresponding weight images of sequence number w_1 through $w_1 + m_2 - m_1$. All weight images must be a single plane or all weight images must be a full cube matching the images.

BSAVG is a special task written to average beam-switched continuum images. Each image is Fourier transformed and weighted to give no weight to Fourier components at the beam switching spatial frequency and direction (since the images lack any non-noise information at these lines in the Fourier domain). Images made at different parallactic angles (*i.e.*, different hour angles) have these zero-weight lines at different angles while images made with different throw lengths have these zero-weight lines at different spatial frequencies. Thus, averaging images in this way (and Fourier transforming them back) should produce images with less noise and more information content. This algorithm works only on images that are made very quickly. If there is a significant rotation of the parallactic angle during the observation of one image, then the zero-weight "line" is actually curved and smeared away from the center (in Fourier space). The failure of this algorithm when observations are made with constant-elevation throws is one reason why some telescopes are designed to beam-switch in celestial coordinates.

10.4.3 Spectral moment analysis

A data cube may be reduced to a line-sum and a predominant-velocity image when the spectral shape is fairly simple at all points of the image. The simplest task to do this is:

> TASK 'XMOM' ; INP ↵	to review the inputs.
> INDI *n* ; GETN *ctn* ↵	to select the input image from disk *n* catalog slot *ctn* — use the output from IMLIN with velocity as the first axis.
> FLUX *x* ↵	to include only pixels $> x$ in brightness when computing the moments.
> GO ↵	to compute images of the $o^{\rm th}$ through $3^{\rm rd}$ moments plus an image of the number of pixels used at each position.

This simple prescription will produce a result which should tell you whether this mode of analysis is interesting. If it is, then the regions of signal should be separated from regions of no signal so that the latter do not contribute to the noise in the moment images. See the discussions in § 7.4 and § 8.6 for methods of doing this. After the non-signal regions are blanked, the moments should be recomputed.

10.4.4 Source modeling and fitting

Gaussian fitting of images is discussed in some detail in § 7.5 while source modeling may be done in the "*uv*" data domain with SDMOD (§ 10.2.5) and in the image domain with IMMOD. The task SAD will find, and fit Gaussians to, sources in your image. Although it works on a plane of the image at a time, it records the plane number in its output model-fit (MF) table. This will allow you to examine the fits to your sources as a function of frequency. To run SAD on a number of image planes:

> TASK 'SAD' ; INP ↵	to review the inputs.
> INDI *n* ; GETN *ctn* ↵	to select the image cube from disk *n* catalog slot *ctn*
> BLC 0 ; TRC 0 ↵	to search for sources over the full plane.
> DORESID FALSE ↵	to delete the residual image after fitting; the fit results are kept in an MF file attached to the input image.
> NGAUSS 10 ↵	to allow up to 10 possible sources to be fit; make this enough to allow for a noise spike or two.
> CUTOFF *x* ↵	to fit "islands" of flux $> x$ only — this is probably the most important parameter.
> DOCRT 132 ↵	to display results on your workstation rather than the line printer.
> DOALL 1 ; DOWIDTH 1 ↵	to allow the task to fit multiple sources to an island and to fit the source widths.

> OUTVERS -1 ↵	to suppress writing of CC files.
> INVERS 1 ↵	to use one MF file for all fits.
> DOWAIT TRUE ↵	to resume AIPS only when the task finishes; this allows looping without tripping over ourselves.
> INP ↵	to recheck the inputs.
> FOR BLC(3) = c_1 TO c_2 ; GO; END ↵	to fit channels c_1 through c_2.

SAD will reject dubious solutions for a variety of reasons. The DPARM adverb allows you to control these reasons and PRTLEV controls how much of an explanation you get.

SAD offers a printer option to provide a detailed account of each execution. To view a simpler summary of the current contents of one or more MF files, use

> TASK 'MFPRT' ; INP ↵	to review the inputs.
> INDI n ; GETN ctn ↵	to select the image cube from disk n catalog slot ctn as input to SAD.
> INVER n_1 ; IN2VER n_2 ; XINC 1 ↵	to view MF file versions n_1 through n_2.
> DOCRT 132 ↵	to see the display on your monitor.
> FLUX 0 ; IMSIZE 0 ↵	to see all components.
> SORT 'C' ↵	to see the file in channel number order.
> GO ↵	to run the task.

Setting DOCRT FALSE and specifying OUTPRINT will produce a file suitable for some non-*AIPS* modeling programs.

10.4.5 Image displays

The subject of displays in *AIPS* has been treated extensively in earlier chapters. To make a printer representation of your image, see § 6.2.2 for a discussion of PRTIM. See § 6.3.2 for a discussion of plotter displays of images including tasks CNTR, PCNTR, GREYS, PLROW, PROFL, IMVIM, and IMEAN. Spectral-line displays are described in some detail in § 8.5.4 including tasks KNTR and PLCUB and the TV-movie display verbs TVMOVI and TVCUBE. The use of the TV for display, image enhancement, parameter setting, data examination, image comparison, and the like is described in detail in §§ 6.4.

For tutorial purposes, we will include one example here. The contouring task of choice is now KNTR since it can display images in grey-scales and/or contours with one or more planes per display and with an optional beam display. It also can plot polarization and has several "coloring" options. For example, to display several spectral channels as contours with the $0^{\rm th}$-moment (total CO) image as a grey scale on each display, enter

> TASK 'KNTR' ; INP ↵	to review the inputs.
> INDI n_1 ; GETN ctn_1 ↵	to select the image cube from disk n_1 catalog slot ctn_1.
> IN2DI n_2 ; GET2N ctn_2 ↵	to select the $0^{\rm th}$-moment image plane from disk n_2 catalog slot ctn_2.
> DOCONT 1 ; DOGREY 2 ; DOVECT -1 ↵	to have contours drawn of the first image, grey-scale of the second image, and no polarization.
> BLC 0 , 0 , c_1 ; TRC 0 , 0 , c_1 ↵	to draw the full plane from channels c_1 through c_2.
> ZINC Δc ↵	to display every $\Delta c^{\rm th}$ channel.
> PIXRANGE B_1, B_2 ↵	to do grey scales from B_1 through B_2 only, clipping the most negative and positive values if desired. The default is the full range of image DOGREY.

> FUNCTYPE 'SQ' ↵	to use a square-root transfer function on the grey scales to emphasize the lower levels.
> OFMFILE ' ' ↵	to do no pseudo-coloring in KNTR.
> DOWEDGE 1 ↵	to plot a step wedge along the top.
> CLEV 0.1 ↵	to plot 0.1 K as the basic contour level.
> LEVS 2.7, 7.4, 20.1, 54.6, 148.4, 403.4 ↵	to do logarithmic contours, starting at 0.27 K.
> CBPLOT 18 ↵	to plot a half-power beam contour in the upper right corner and fill it in.
> LABEL 1 ↵	to label each pane with its coordinate (velocity usually).
> DOTV -1 ; INP	to make a plot file and to review the inputs.
> GO ↵	to run the task.

The contour lines will be drawn in a contrasting color when the background grey-scale intensity is high. When KNTR has finished:

> PLVER 0 ↵	to plot the most recent plot file for the image.
> OUTFILE ' ' ↵	to print the plot immediately rather than saving it in a file.
> GO LWPLA ↵	to translate the plot file into PostScript on a suitable printer.

LWPLA offers additional control over fonts, paper size, line width, the grey-scale plotting (if PIXRANGE was not quite right), image pseudo-coloring, coloring of lines and backgrounds, and number of copies. It can make an "encapsulated" PostScript file for inclusion in other documents, such as this *CookBook*. See HELP POSTSCRIPT for information on other things that can be done with PostScript plot files.

10.4.6 Backing up your data

The next chapter describes how to help the *AIPS* programming team (with "GRIPEs"), to exit AIPS (with EXIT), to delete your data (with ZAP and ALLDEST), and, most importantly, to back up your data. Do not assume that data on disk is permanent. Disks can fail and users can make mistakes, so it it is wise to make backups to some demountable medium.

10.5 Combining single-dish and interferometer data

We add this section to this chapter with some trepidation since the combination of single-dish data into interferometric imaging is still an area more suited to research than to production. In principle, the problem is fairly simple. You begin by observing a region of sky with a single-dish telescope rather larger than the individual telescopes of the interferometer. From these observations, you make an image which you correct if necessary (*e.g.*, by removing spectral baselines). Then you deconvolve the image removing the convolution of the sky with the beam of the large single-dish telescope. The "sky" observed with the interferometer is the product of the real sky (estimated by your deconvolved image) and the beam of the individual telescopes of the interferometer. Therefore, you multiply your deconvolved image with an image of the single-dish beam and Fourier transform the result. Adjusting the flux scales (usually of the single-dish data), you append or "feather in" the "visibilities" produced by the Fourier transform.

This is a lot of steps and contains several dangers, namely pointing, image alignment, the deconvolution, and the flux re-calibration. *AIPS* can provide you with some help. The imaging and image correction software is described earlier in this chapter. The deconvolution is tricky. Try DCONV first. It attempts an iterative solution of the deconvolution problem in the image plane. If that is not acceptable, try CONVL with OPCODE 'DCON' (in 15JAN96 and later releases). This is a brute force deconvolution that will be very noisy at high spatial frequencies, but these frequencies will be tapered or truncated away later. A third approach

is to use PATGN (OPCODE 'GAUS') to make an image of the single-dish beam of the large telescope. APCLN (§5.3.7) can then be persuaded to do a Clark image-based Clean; use a small restoring beam. Remember that this image will be tapered in the *uv* plane. It does not have to be beautiful in detail in the image plane.

The next step is to make an image of the interferometer single-dish beam on the same cell size and center as your deconvolved image. Use PATGN with OPCODE 'BEAM' for this. Then multiply the result by the deconvolved image with COMB using OPCODE 'MULT' (§7.1). If this produces an image with any blanked pixels, run REMAG to convert the blanks to zeros. Then start trying IM2UV to produce a *uv* data set. Use UVTAPER to weight down longer spacings, FLUX to scale the visibilities, and UVRANGE to omit the outer spacings. (The first two options appear only in 15JAN96 and later releases.) You should use PRTUV, UVPLT, and even UVFLG on the output of IM2UV to make sure that the visibility phases and amplitudes of your single-dish and interferometer data are in reasonable agreement. Finally, combine the two data sets with DBCON and have fun with IMAGR (Chapter 5).

10.6 Additional recipes

10.6.1 Chicken salad with banana mayonnaise and grapes

1. Place 3 medium **bananas** cut in chunks, 2 teaspoons chopped **garlic**, 3/4 cup non-fat **plain yogurt**, 1 tablespoon **honey**, 2 teaspoons **lemon juice**, and 1/4 teaspoon **salt** in a blender or food processor. Blend until creamy.
2. Arrange 12 cups mixed **lettuces** on six plates.
3. Toss 6 **chicken breasts** cooked and cubed with banana mayo; divide onto salads.
4. Sprinkle with 2 bunches ($\approx$ 48) halved **grapes** and 1/2 cup **walnut** or **pecan** halves.

Thanks to Chiquita Bananas.

10.6.2 Curried bananas

1. Melt 2 tablespoons **butter** in saucepan and cook 2 tablespoons minced **onion** in it for 2–3 minutes.
2. Mix 1 tablespoon **curry powder**, 1 teaspoon **salt**, 1/4 cup **flour**, and a dash of **cayenne pepper** with a little **milk** to make a paste.
3. Add paste to onion, cooking gently for 10 minutes. Add balance of 2 cups **milk** slowly, stirring until it boils.
4. Slice 7 small green **bananas**, and cook gently in the sauce until tender.
5. Serve as a vegetable in a ring of hot cooked rice.

From *Everyday BANANA Recipes*, Banana Distributing Co., New Orleans, published by Bauerlein, Inc. New Orleans, 1927.

10.6.3 Hawaiian banana cream pie

1. Preheat oven to 375° F.
2. In a bowl, combine 1 cup chopped cashew or macadamia **nuts**, 1/2 cup flaked **coconut**, and 2 tablespoons **brown sugar**.
3. Beat 1 **egg white** until stiff; fold into nut mixture.
4. Press mixture evenly into an 8-inch pie plate, building up the sides slightly. Bake for 7 minutes or until crust is lightly browned. Crust will tighten as it cools (use a rack).
5. In a medium-sized saucepan, beat 3 **egg yolks**. Mix in 5 tablespoons **cornstarch** and 3/4 cup granulated **sugar**. Stir in 1.5 cups **milk**, 1/4 teaspoon **salt**, and 1 tablespoon unsalted **butter**.
6. Cook mixture slowly over medium heat, stirring constantly, for 5 to 7 minutes. Filling should be bubbling and thick.
7. Remove from heat and stir in 1 teaspoon **vanilla extract**. Transfer this custard to a glass bowl, cover with plastic wrap, and refrigerate for 2 hours.
8. Two hours before serving, whip 1/2 cup heavy **whipping cream** to stiff peaks and fold into custard. Peel and slice one **banana**, arranging evenly on bottom of crust. Spoon custard filling into crust. Cover again with plastic wrap and chill for 2 more hours.
9. Sprinkle 1/2 cup finely chopped cashew or macadamia **nuts** evenly over the filling. Peel, slice and arrange a second **banana** in a circular fashion around the outside top of the pie, placing a few slices decoratively in the center.

10.6.4 Banana Dream Pizza

1. Preheat oven to 400° F. In a large bowl, combine 2 1/2 cups **all-purpose flour**, 2 tsp **baking powder**, and a pinch of **salt**. Add 4 Tsp softened **sweet cream butter** and blend. Add 3/4 cup warm **milk** and mix well. If the dough is still sticky, add a small amount of flour.
2. Form the dough into a ball. Knead it on a floured surface until it is smooth. Roll out the dough and place it in an oiled, 16-inch pizza pan. Bake for 15-20 minutes, or until the crust is light brown.
3. In a nonmetallic bowl, mash 4 **bananans**. Add 1 teaspoon **lime or lemon juice** and 6 tablespoons **honey**; mix well.
4. Slice 2 **bananas** horizontally and place the slices in water to cover. Add 1 teaspoon **lime or lemon juice** to prevent discoloration.
5. Spread the banana mixture on the crust.
6. Drain the sliced bananas and blot them with paper towels. Place them in a circular pattern on the banana mixture. Baste the banana slices with 3 tablespoons **melted butter**.
7. Bake for 20–30 minutes at 400° F until the crust is golden brown.
8. Remove from the oven and top with 1 quart **vanilla ice cream** and 1/2 cup chopped **macadamia nuts** while still hot. Serve immediately.

11 Exiting from, and Solving Problems in, $\mathcal{AIPS}$

This chapter contains a grab-bag of miscellaneous advice on exiting from $\mathcal{AIPS}$ and on solving a variety of common problems that may arise. The latter are also addressed in § Z.1.4.

11.1 Helping the $\mathcal{AIPS}$ programmers

Comments, suggestions and bug reports about any facet of $\mathcal{AIPS}$ are very useful to the $\mathcal{AIPS}$ programming and management group. Note that "gripes" are only useful when they are informative — *e.g.*, giving details of the circumstances under which a task failed with accompanying system error messages (if any). Terse gripes along the lines of "UVCOP doesn't work!" whilst perhaps true in some circumstances, are unlikely to arouse the $\mathcal{AIPS}$ programmers' enthusiasm. In many cases, it may be necessary for the programmer to use your data to fix the bug. A FITS-disk file read over the net is a common means to this end. The $\mathcal{AIPS}$ group may often seem unresponsive to your gripe. This is an unavoidable consequence of the breadth of the $\mathcal{AIPS}$ project combined with the small size of the group. Nonetheless, if you do not tell the programmers that there is a problem or a good idea, then you are almost certain to encounter the same problem years later and never to see your good idea put into practice. The $\mathcal{AIPS}$ group depends on help from users.

Gripes can be entered into a site-wide "GR" file and automatically mailed to daip@nrao.edu by typing:

> GRIPE C$_R$

while in AIPS. Follow the directions to record your comment. Current gripes in the file may be read via

> GRINDEX C$_R$ — to display an index of all gripes in the file.

> JOBNUM n ; GRLIST C$_R$ — to list the $n^{\rm th}$ gripe in the file.

and a gripe may be deleted with

> JOBNUM n ; GRDROP C$_R$ — to delete the $n^{\rm th}$ gripe in the file and notify daip.

Note that the deleted gripe has already been e-mailed to the $\mathcal{AIPS}$ group in Socorro, so dropped ones get sent too. Do not do a GRDROP unless you realize that the gripe was erroneous. (An explanatory "gripe" would be appreciated.) If you change your mind about a gripe before you finish it, type _forget or _FORGET (case sensitive!) to stop the gripe before it is mailed and entered in the file. The addition of automatic e-mail gives immediacy to all gripes and provides, for the first time, real access to the gripe system for sites outside of the NRAO.

11.2 Exiting from $\mathcal{AIPS}$

Before ending a period of data reduction with $\mathcal{AIPS}$, you should back up those data files which you wish to keep, delete all of your disk data files, tidy up your work area, and then issue the KLEENEX (to stop the TV, Tek, and message servers too) or EXIT (to leave the servers running) command to the AIPS program. Of course, if the computer and disks are part of your very own workstation in your office, you may ignore all this advice. Backups on magnetic tape used to be a good idea, but there are few if any such devices now available. Disk files are easily deleted due to software or user malfunction, or lost due to disk hardware malfunction.

Note that, when you exit AIPS you leave behind some nearly invisible files even if you have attempted to remove al of your data files. The "hidden" files are in your AIPS data area number one and include message (MS), readline (RL), SAVE/GET (TG), and task data (TD) files. In 31DEC23, task CLNUP will list these files as well as data files older than a specified interval and will delete old "hidden" files. Old data files must be deleted using ZAP or ALLDEST.

11.2.1 Backups

While processing and particularly just before exiting from AIPS, please delete as many of your own data sets as possible. Images and *uv* data may be backed up on disk in FITS format using the task FITTP. This task can write only one disk file at a time; tapes were more cooperative. For example, to backup a *uv* file, type

> TASK 'FITTP' ; INP ↵	to review the inputs.
> DOALL FALSE ↵	to specify that only one file is written.
> IND *n* ; GETN *ctn* ↵	where *n* and *ctn* select the disk and catalog numbers of the data set to be deleted.
> DATAOUT 'FITS:Mydata.fits ↵	Output file name; note the lack of a close quote to maintain case.
> DOPLOT 1 ↵	to include any plot and slice files in 31DEC25.
> INP ↵	to check the inputs.
> GO ↵	to write the disk file.

The FITS format used for plot and slice files will not be useful to any software other than AIPS, but it does allow saving the plot files for later use in AIPS. The RUN file WRTPROCS contains two procedures to assist in writing more than one AIPS data set or image to disk. FITDISK generates the full DATAOUT from an input DATAOUT and the AIPS file name. WRTDISK generates the full DATAOUT from an input DATAOUT and the AIPS disk and catalog numbers. These procedures and FITAB support the plot file option.

Task FITAB also writes FITS disk files. For *uv* data it has the advantage of being able to write the data in compressed form, saving disk space, and of writing the data in multiple "pieces" for increased reliability. Unfortunately, the table form of data used may not be understood by other software systems. Note that FITAB is used for processed *uv* data by the NRAO archive — it will matter to many users to update to a modern release. FITAB may apply a quantization to images on output that allow the FITS files to be compressed very much more efficiently. If the quantization level is set below 1/4 of the image noise, then the noise in the output image will only be 1–2% larger than in the input image.

11.2.2 Deleting your data

Please delete redundant images and data as soon as possible to preserve disk space for yourself and other users. It is tempting to work on many sets of data at the same time, but this generally takes a lot of disk space and users should limit the amount of data resident on disk to that which will be processed during the session. A data set and all extension files can be deleted by:

> IND *n* ; GETN *ctn* ↵	where *n* and *ctn* select the disk and catalog numbers of the data set to be deleted.
> ZAP ↵	to do the deletion.

To delete data in contiguous slots from *n* to *m* in a catalog, set the INDISK and use the loop:

> FOR I = *n* TO *m* ; GETN I ; ZAP ; END ↵

For massive deletions — the kind we hope you will use when you leave an NRAO site — use:

> ALLDEST ↵ to destroy all data files consistent with the inputs to ALLDEST.

And to compress your message file, after using PRTMSG to print any you want to keep, use:

> PRNUM -1 ; PRTASK '' ; PRTIME 0 ↵ to do all messages.

> CLRMSG ↵ to do the clear and compress.

DO NOT DELETE OTHER USERS' DATA OR MESSAGES WITHOUT THE EXPLICIT PERMISSION EITHER OF THE OTHER USER OR OF THE AIPS MANAGER.

11.2.3 Exiting

To exit from AIPS type:

> EXIT ↵ to leave TV, message, and graphics servers running, or

> KLEENEX ↵ to kill server processes as well as AIPS.

Please clean up any papers, tapes, etc. in the area around your terminal before you go.

11.3 Solving problems in using *AIPS*

On all computer systems things go wrong due to user error, program error, or hardware failure. Unfortunately, *AIPS* is not immune to this. The section below reviews several general problem areas and their generalized solutions. Refer to § Z.1.4 for the details appropriate to NRAO's computer systems. Some well-known possibilities follow.

11.3.1 "Terminal" problems

If your workstation window is alive, but AIPS has "disappeared" you may have "suspended" it by typing CTRL Z. The AIPS can be left in a suspended state, placed into the "background" with bg, or returned to the "foreground" again with fg after which it will resume accepting terminal input. If your *AIPS* appears to be "suspended", try typing jobs to see which jobs are attached to your window and then use fg %*n* to bring back job *n* where *n* is the job number of the suspended AIPS. If no *AIPS* job is suspended from the current window, check all other windows you have running on the workstation for the missing simian before starting a new AIPS. Otherwise, you may run out of allowed AIPSes and/or encounter mysterious file locking problems.

If your workstation window (or terminal on obsolete systems) is "dead", *i.e.*, refuses to show signs of talking to your computer, you have a problem. There are numerous possible causes. If typed characters are shown on the screen, but not executed, then

1. Are you executing a long verb, *e.g.*, REWIND, AVFILE, RESCALE? If so, be patient.
2. Are you executing some interactive TV or Tek verb which is waiting for input from the cursor or buttons? If so, provide the input.
3. Have you started a task with DOWAIT set to TRUE (+1.0)? If so, wait for the task to finish. Most tasks report their progress on the message monitor window (or your input window).
4. Is AIPS waiting while a tape rewinds or skips files or is it waiting to open some disk file currently being used by one of your tasks? Be patient.

If typed characters do not appear, then

1. Have you stopped output to your window accidentally by hitting the appropriate NO SCRL or other XOFF control sequence? If so, hit the XON control sequence. (These are CTRL S and CTRL Q, respectively.)

2. Do other windows connected to the computer appear to be "alive"? If so, use one of them and inquire about the status of your AIPS program and tasks; on Linux and Berkeley Unix try ps aux ↵ and on Linux, Solaris and other Bell Unix try ps -elf ↵. It might be necessary to stop your old AIPS session from your new window and then use that window to start a new AIPS.

3. Can you abort AIPS at your window using the appropriate system commands (*i.e.*, CTRL C on Unix machines)?

4. If all windows appear dead, then your computer or its X-Windows server may have "crashed." Try a remote login from another computer. If that works, check on your processes and try to kill the server and other tasks. This should return your computer to a login state. Otherwise, report the problem to your *AIPS* Manager or System Administrator. If you feel you must reboot the system, do so *only* after checking that all current users and the System Administrator (if available) agree that that action is required.

5. If even a reboot fails, report the problem to the System Administrator or hardware experts and go do something else. UNDER NO CIRCUMSTANCES SHOULD YOU ATTEMPT TO REPAIR ANY HARDWARE DEVICES. Such repairs must be performed by trained personnel.

11.3.2 Disk data problems

If you encounter the message CATOPN: ACCESS DENIED TO DISK *n* FOR USER *mmm*, it means that user *mmm* has not been given access to write (or read) on limited-access disk *n*. The access rights for all disks can be checked by typing FREESPAC in the AIPS session. In the list of mounted disks, the Access column can say Alluser, Scratch (scratch files only), Resrved (limited access including you), and Not you (limited access not including you). If you feel that you should have access to that particular disk, resume using your correct user number or see your *AIPS* Manager about enabling your user number.

If your data set seems to have disappeared, consider

1. Have you set INDISK *et al.* (especially INTYPE) correctly before running CAT? Type INP CAT ↵ to check. Is USERID not set to 0 or your user number?

2. Are you connected to the right *AIPS* computer, if your site has more than one?

3. Are the desired disks mounted for your AIPS session? Type FREE ↵ to see which disks are currently running and which numbers they are assigned in this session. When you attach disks from other computers (using the da= option of the aips command — § 2.2.3), they are assigned numbers which depend on the list of computers and which may thus vary from session to session.

4. Did you leave your file untouched for a "long" time on a public disk? System managers may have had to delete "old" files to make room for new ones. In this case your data are gone and we hope you made a backup FITS file on disk.

The message write failed, file system is full will appear when the search for scratch space encounters a disk or disks without enough space. (*AIPS* usually emits messages at this time as well.) This is only a problem when none of the disks available for scratch files has enough space, at which point

the task will "die of unnatural causes." Run the verb FREESPAC to see how much disk is available and then review the inputs to the task to make sure that OUTDISK and BADDISK are set properly. Change them to include disks with space. Check the other adverbs to make sure that you have not requested something silly, such as a 2000-channel cube 8096 on a side. Then try again.

If there simply is not enough space, try some of the things suggested in § 3.6, such as SCRDEST to delete orphan scratch files, DISKU to find the disk hogs, and, if all else fails, ZAP to delete some of your own files. DISKU may be run with DOALL = n to list catalog entries that occupy more than n Megabytes. This will help identify those files which will yield the most new space when deleted. Your *AIPS* Manager may help you by removing non-*AIPS* files from the *AIPS* data disks. Do not do this yourself unless they are your files.

11.3.3 Printer problems

All *AIPS* print operations now function by writing the output to a disk text file, then queuing the file to a printer, and then sometime later, deleting the file. After the job is queued, the *AIPS* task or verb will display information about the state of the queue. Read this carefully to be sure that the operation was successful and to find out the job number assigned to your print out. If you are concerned that your print job may be lengthy, or expect that you will only need a few numbers from the job, please consider using the DOCRT option to look at the display on your terminal or the OUTPRINT option to send the display to a file of your choosing without the automatic printing. See § Z.1.4.3 for information about printing such files later.

To find out what jobs are in the spooling queue for the relevant printer, type, at the monitor level:

`$ lpq -P`*ppp* ↵ to show printer *ppp*.

`$ lpstat` *ppp* ↵ to show printer *ppp* under Solaris, HP, SGI (Sys V systems).

where *ppp* is the name of the printer assigned to you when you began AIPS. If the file is still in the queue as job number *nn*, you can type simply

`$ lprm -P`*ppp nn* ↵ to remove the job.

`$ cancel` *nn* ↵ to remove the job under Sys V systems.

`lprm` and `cancel` will announce the names of any files that they remove and are silent if there are no jobs in the queue which match the request.

Since modern printers are capable of swallowing large amounts of input, your job may still be printing even though it is no longer visible in the queue. If you turn off the printer at this stage, you are likely to kill the remainder of your print job and quite possibly one or more other print jobs that followed yours. Use discretion. Do not turn the printer back on if the job is still in the queue. Most systems will start the print job over again after you turn the power back on without doing a `lprm` or `cancel`.

If your printout fails to appear

1. Did the print queuing actually work? Review the messages at the end of the verb or task.
2. Did the printout go to a printer other than the one you expected? Was it diverted to a printer used for especially long print jobs or one used for color plots? The messages at the end of the verb or task should show this.
3. Was the printer not working or backed up for so long that the file was deleted before it could be printed? The delay time for deletion is shown at the end of the verb or task. It can be changed by your *AIPS* Manager for future jobs.
4. Was your print job, or that of a user in the queue ahead of you, a large plot? These can take a long time in some PostScript printers (usually indicated by a blinking green light), so be patient.

11.3.4 Tape problems

The main problem with tapes these days is that no one has any tape devices. The following applies if you are one of the "lucky" few.

When AIPS does a software MOUNT of a magnetic tape, it actually reads the device on most systems. An error message along the lines of ZMOUN2: Couldn't open tape device ... usually means that you have attempted the MOUNT before the device was ready. Wait for all whirring noises and blinking lights to subside and try again. Remote tape mounts are more fragile. If you get a message such as ZVTPO2 connect (INET): Connection refused, then the tape dæmon TPMON is probably not running on the remote host. EXIT and restart AIPS, specifying the remote host in the tp= option (see § 2.2.3). If you are told AMOUNT: TAPE IS ALREADY MOUNTED BY TPMON, then there is a chance that you are trying to mount the wrong tape or that someone left the tape device in a mounted state. See § Z.1.4.6 for advice on curing this stand-off between AIPS, which knows that the tape is not mounted, and TPMON which knows that it is.

If you are having problems reading and writing a tape, consider

1. Did you actually mount the tape in software from the *AIPS* level with the MOUNT verb. A message like ZTPOPN: NO SUCH LOGICAL DEVICE = AMTO*n*: indicates that you have not.
2. Have you specified the INTAPE or OUTTAPE number to correspond with the drive you mounted the tape on?
3. Does your computer have access to tapes on the remote host? The message AIPS TAPE PERMISSION DENIED ON REMOTE HOST suggests not. See the *AIPS* Manager for the remote host.
4. Is the tape correctly loaded in the drive and is the drive "on line" (check the ON LINE light)?
5. Have you set the density correctly? Some drives need the density to be set by a switch, others have software control. Some try to read the tape and sense the density automatically. Be aware that some drives do not set the density until you actually read or write the tape. Under these circumstances, the density indication on the drive can be misleading. If in doubt, consult your local *AIPS* Manager about the meaning of the tape density indicator lights on the drive you are using.
6. Are you using the correct program to read the tape? If you are unsure of the format of a tape, use the task PRTTP to diagnose it for you. It will recognize any format that *AIPS* is able to read.
7. Are you writing to a completely blank tape? This fails sometimes. Or are you writing to an old tape which is new to you? In both cases, try specifying DOEOT FALSE C_R and then rerunning the tape-writing program.
8. Has the drive been cleaned recently? Do *not* attempt to clean a drive yourself. Using the wrong cleaning fluid or cleaning the wrong parts of a drive can do serious damage. If you have any doubts, use another drive.
9. Is your tape defective? Tapes can lose oxide or become stretched, creased, or dirty, all of which will cause problems. Try using another tape, if possible.

12 *AIPS* for the More Sophisticated User

The program AIPS uses the *POPS* language to communicate with you. *POPS* has many capabilities that have been hidden or taken for granted in the previous sections. Once you start to become familiar with *AIPS*, you will need to know more about *POPS* to take full advantage of the powerful features which are available. The first section below describes some of the shortcuts and conventions of *AIPS* while the second section describes program flow control options.. The third section describes multiple features of the *POPS* language including the constructions of procedures. The fourth section addresses the needs of "remote" users of *AIPS*, while the last section discusses how to begin writing your own *AIPS* tasks.

12.1 *AIPS* conventions

12.1.1 AIPS shortcuts

Some niceties of using *POPS* syntax in *AIPS* are:

1. More than one expression can be put on a line. These expressions must be separated by a semicolon (;). Exceptions are RESTORE, GET, RUN and a few other "pseudo-verbs" which, with their arguments, must stand alone. For example, GET MYIMAGE ; INDISK = 1 ↵ will ignore the INDISK = 1. When in doubt, see the HELP files for the pseudo-verb to find the restrictions on its use.

2. As in many other systems, recognized keywords in AIPS do not need to be typed in full. You must type only enough of the leading characters to get a unique match. This "minimum-matching" has been exploited throughout this *CookBook*. If in doubt, hit the Tab key. The keyword will be completed if it is unique or the screen will blink. Hit the Tab key a second time to see your choices.

3. The parameter variables in AIPS are called "adverbs." They are assigned values by the equals verb (=), *e.g.*, INDISK = 2 ↵. The equals sign may usually be replaced by a space. The exception arises when the variable on the left is a sub-scripted array element and the expression on the right involves a unary minus or other function reference (*e.g.*, APARM(3) = -1 ; APARM(4) = SIN(X) ↵).

4. Array adverbs are set to a constant value by putting a single value on the right hand side of the equals sign, *e.g.*, CELLSIZE = 1.5 ↵. A list of values may be put in the array by putting the list on the right hand side of the = sign separated by commas (,), *e.g.*, LEVS = -1, 1, 2, 3, 6, 9 ↵. The commas may be replaced by spaces in most cases. An exception occurs if an element is negative or some other arithmetic expression. Thus, SHIFT = -19 -2 ↵ will produce SHIFT = -21, -21.

5. A list of values may also be put in a sequence of scalar adverbs with the ∼ (tilde sign) adverb. The main use of this is to overcome the 128-character limit for input lines to *POPS* in assigning values to an array. Thus, for example,
   RASHIFT = -3000, -2500, -2000, -1500, -1000, -500, 0, 500, 1000 ↵
   RASHIFT(10) ∼ 1500, 2000, 2500, 3000 ↵

6. Adverbs can be used in arithmetical expressions or set equal to other adverbs, *e.g.*, OUTNAME = INNAME ; OUTSEQ = 2.5 * INSEQ + 3.

7. Both upper and lower case letters may be entered by the user; with one exception, *AIPS* is case insensitive. That exception is in adverb character string values, which are converted to upper case by the trailing quote sign. If you omit the trailing quote — and make that the last command on the input line — then case is preserved (and used) in the string.

As an example, these shortcuts allow the following AIPS command sequence:

```
> INNAME = '3C138' CR
> INCLASS = 'IMAGE' CR
> INSEQ = 0 CR
> BLC = 200 , 200 CR
> TRC = 300 , 300 CR
> OUTNAME = '3C138' CR
> OUTSEQ = 5 CR
> GO PRTIM CR
```

to be shortened to:

```
> Inn '3c138' ; INC 'image' ; blc 200 ; Trc 300 CR     (Note use of upper and lower case.)
> OUTN INN ; OUTS INS + 5 CR
> go prti CR                                           (Task name can be in either case, too.)
```

12.1.2 Data-file names and formats

The physical name of the data file is generated internally, depends on the type of computer, and will not often concern you as a user. You will refer to an image by specifying its disk number, the type of image (`'MA'` for images, `'UV'` for *uv* data), and the following three parts of the image designation:

1. Name — A string of up to 12 legal characters.
2. Class — A string of up to 6 legal characters.
3. Seq — A number between 0 and 46655.

Each of these parts corresponds to separate input adverbs called INNAME, INCLASS, and INSEQ (and their variations). You can choose the image name arbitrarily and sensible choices will reduce other book-keeping. Many programs will choose a reasonable image name if you do not specify one.

A common set of conventions for the name adverbs is used throughout *AIPS*. INNAME ' ' CR means "accept any image name with the specified class and sequence." INSEQ 0 CR means "accept any image with the specified name and class" or, if only one image is to be used, "accept the image with the specified name and class having the highest sequence number." OUTNAME ' ' CR means "use the actual INNAME." OUTCLASS ' ' CR means "use the task name," except for tasks that write more than one output image, in which case task-based defaults will be used. OUTSEQ 0 CR means "use a sequence number that is one higher than that of any files currently on disk with the same name and class as the requested output file." The name and class strings also support "wild-card" characters for input and output. This feature is especially powerful in tasks, such as FITTP, that can be told to operate on *all* images that match the specified name parameters. Type HELP INNAME CR and HELP OUTNAME CR for details. The verb NAMEGET will convert the INNAME set of adverbs to the values that would be used by a task. This is especially useful in complicated procedures.

Only the array data, in the form of 4-byte floating-point numbers, are stored in the image or *uv* data file. The header information is stored separately for each image or *uv* data set. Directory information is stored in a special file, called the Catalog File. Each disk has one such file for each user and it contains directory information for all images and *uv* data sets belonging to that user.

Extension files may be associated with any image data file. Each image can have (in principle) up to fifty types of extension files and up to 46655 "versions" of each type. These subordinate files contain additional information associated with the image and are designated by a two-letter type code. `'HI'` is a history file,

'CC' is a Clean components file, 'PL' is a plot file, 'AN' is an antennas file, 'SL' is a slice file. In *AIPS*, an extension file associated with an image is uniquely specified by the usual file-naming adverbs plus the extension file type (adverb INEXT) and the version number (adverb INVERS). The default convention for INVERS is reasonable — on input, zero means the highest (*i.e.*, most recent) version and, on output, the highest plus one.

Array elements in an image are designated by their pixel coordinates (counts). If $M(i,j,k,l,m,n,o)$ is a seven-dimensional array, the (1,1,1,1,1,1,1) pixel will be associated with the lower left hand corner of the image. The $i^{\rm th}$ (first) coordinate increments fastest and is associated with a column in each plane of the image. The $j^{\rm th}$ (second) coordinate is associated with a row in each plane. The other coordinates allow the image to be generalized to cover up to seven dimensions, *i.e.*, "cubes" and the like. The two adverbs BLC for bottom left corner and TRC for top right corner let you specify the desired sub-array in up to seven dimensions. When a sub-image is taken from an image, the pixel designation of any image element will usually change.

12.2 Process control features of *AIPS*

There are a number of tools available to assist you in scheduling and controlling the execution of *AIPS* tasks. They include taking the input from a text file rather than an interactive terminal, special adverbs to the verb GO, and even one or more detached, batch processing queues.

12.2.1 RUN files

If you have some lengthy sequence of commands to give to AIPS, especially if you will have to give essentially the same sequence more than once, you may find it helpful to prepare the sequence in a text file using your favorite and flexible text editor. In particular, it is cumbersome to write and edit procedures more than about five lines long in *AIPS* because of its primitive internal editor. Long procedures are best written as text files at your computer's monitor level, where the editing facilities will usually be much better. These text files can be transferred to AIPS easily using the RUN file facility.

Any commands that can be typed on the terminal to AIPS can be stored in your text file. The text files may be stored in a disk area of your choosing or in a public area identified by the logical name $RUNFIL. The name of the file must be all upper case letters, followed by a period, followed by your user number as a three-digit "extended-hexadecimal" number with leading zeros. (To translate between decimal and extended hexadecimal, use the *AIPS* procedures or the AIPS verbs called EHEX and REHEX.) To use the RUN file from your own disk area, define a logical name before starting AIPS to point to the area (see § 3.10.1). Then start up AIPS under your user number and enter

> VERSION = 'MYAREA' ↵ where MYAREA is your disk area, or
> VERSION = ' ' ↵ if $RUNFIL is to be used
> RUN *FILE* ↵ to execute the file named *FILE.uuu*

where *uuu* is your user number in extended hexadecimal with leading zeros to make three digits. The file *FILE.uuu* or, if it does not exist, *FILE.001* will be executed by the above command. Note that minimum match also applies to RUN file names.

The first line of a RUN file is ignored by AIPS. You should type comments into your RUN files to remind you what they are doing. The first line, any line which begins with an * in column one, and all text following a $ sign (in any line) are treated as comments which will not be compiled or executed by AIPS. All facilities in AIPS such as GET, SAVE, TGET and TUGET can be used in RUN files. You may even nest RUN files within RUN

files up to 20 deep, but remember that the pseudo-verb COMPRESS makes use of the RUN process. The full contents of a SAVE area may be put in a RUN file using SG2RUN.

AIPS programmers also provide a few RUN files for general use in each release *e.g.*, VLACALIB, VLACLCAL and VLARESET. They are normally used to create procedures helpful to some, but not all, *AIPS* users. These are stored under user number 1, but are available automatically to everyone. To use a procedure from one of them, type, for example:

> RUN VLAPROCS ↵ to read and execute file VLAPROCS in AIPS. The text will be listed as it is read.

This file contains a number of procedure definitions including those named above. To execute the procedure, prepare the input adverbs as needed, and then type

> VLACALIB ↵ to execute the calibration sequence of the simplified procedure.

AIPS programmers will provide a file with the inputs and help information for all canned procedures. Thus you can do

> HELP VLACAL ↵ to see help information on the procedure.

> INP VLACAL ↵ to see the current inputs to the procedure.

The help and inputs functions may require the RUN if special adverbs are used by the procedure, but you must do the RUN to compile the procedure before you can use it.

12.2.2 More about GO

The verb GO is shown in examples throughout this *CookBook*. Don't overlook the fact that GO, like all other *AIPS* verbs, has its own inputs. You will be familiar with the ability to specify which task you want to execute either by an immediate argument, *e.g.*, GO UVSRT ↵, or by the parameter TASK, *e.g.*, TASK 'UVSRT' ; GO ↵. GO has three further parameters, DOWAIT,VERSION, and QCREATE. The value of DOWAIT is passed to the task and instructs it to resume AIPS as soon as possible (DOWAIT FALSE ↵) or to resume AIPS only after completing its operations (DOWAIT TRUE ↵). The latter option lets the task return a meaningful error code to which AIPS may respond by aborting the current input line, procedure, FOR loop, etc. Note that the verb WAIT '*taskname*' also forces AIPS to wait for a task to complete, but it cannot respond to some failure in that task. For example, the line:

> GO UVSRT ; WAIT UVSRT ; GO UVMAP ↵

may cause unwanted images to be generated by UVMAP if UVSRT fails for lack of disk space or some other reason. However, the line:

> DOWAIT TRUE ; GO UVSRT ; GO UVMAP ↵

will not attempt to execute UVMAP if UVSRT fails. Note that AIPS will not get hung up when a task aborts even if DOWAIT is true. *This consideration is particularly important when you do multiple runs of* FITTP *to write your output tape. If one fails, the tape may even be rewound. If the next scheduled* FITTP *actually runs, it may write at the beginning of tape, destroying all previous data on the tape!*

VERSION is used to specify which version of the program you wish to execute. You might use this to select the TST or OLD versions of a task from the NEW version of AIPS, for example. VERSION also allows you to execute private versions of programs and even to check a list of areas for versions of your program. Type HELP VERSION ↵ for details. The last adverb, QCREATE = 1, instructs tasks to create files quickly. This can save a lot of time, but does not guarantee that the disk space will be available when the task gets around to needing it.

GO has another useful capability. Normally, in order to invoke a verb, you simply type its name, *e.g.*,:

> PRTMSG ↵ to print the message file contents.

However, if you type, instead:

> GO PRTMSG ↵

having forgotten that PRTMSG is a verb, then AIPS will actually execute:

> TPUT PRTMSG ; PRTMSG ↵ to save the current PRTMSG inputs and then print the contents of the message file.

You can recover those inputs at a later time with TGET PRTMSG ↵.

12.2.3 Batch jobs

The *AIPS* batch processor can be used to run jobs outside interactive *AIPS*. The job consists of a set of *AIPS* instructions which do not need user interaction. This excludes the TV, the Tektronix, and the tape-drive oriented tasks and verbs. RUN files may be used in batch jobs — as the batch editor facility is also primitive, they are particularly attractive to batch users. Older restrictions on which tasks may be run in queue 1 have been removed in the 15JAN96 release.

The instructions to be executed in the batch processor are prepared in a "workfile." The workfile can be made while in AIPS and detailed instructions are given by typing:

> HELP BATCHJOB ↵

A simple example is given here:

> BATQUE = 2 ; BATCLEAR ↵ to select queue 2 and clear its workfile.
> BATCH ↵ to enter batch preparation mode.
< TASK = 'UVSRT' ↵ (Notice < prompt. Begin typing as in AIPS.)
< INN = '3C16' ; INCL = 'UVDATA' ↵
< INSEQ 1 ; OUTN INN ; OUTCL INCL ↵
< OUTSEQ = 0 ; SORT = 'XY' ↵
< GO ↵ (Batch AIPS always waits for a task to finish before continuing.)
< RUN XXXXX ↵ (RUN files are good to use in a batch job.)
< GO ↵
< ENDBATCH ↵ to leave batch preparation mode type in ENDBATCH spelled out in full.
> (Resume normal interactive processing.)

To list a batch file, type:

> BATFLINE = 0 ; BATLIST ↵

To edit line *n* in a batch file, type:

> BATEDIT *n* ↵
< *put text here* ↵ to replace old line *n*.
< *some more text* ↵ to insert more commands between old lines *n* and *n*+1.
< ENDBATCH ↵ (spelled out in full.)
> (Resume normal interactive processing.)

As with procedures (§ 12.3.2), if *n* is an integer, the existing line *n* is overwritten with the line or lines typed before ENDBATCH. If *n* is not an integer, the new lines are simply inserted between lines *n* and *n*+1. BAMODIFY provides, for workfiles, the same functions as MODIFY does for procedures.

Finally, the workfile can be submitted to the batch processor by typing:

> SUBMIT ↵

The instructions are sent to a checking program which checks that the input is free of obvious errors. All RUN files are expanded and checked. If Checker (the task AIPSC*m* where *m* is some extended hexadecimal number < Z) approves, the job goes into the *AIPS* job queue, which is managed by QMNGR*n*. If you change your mind, the job can be removed from the queue and returned to the workfile with the verb UNQUE. The batch job may be submitted to any local computer, not just the one on which the current *AIPS* session is running. Adverbs REMHOST and REMQUE control this option. The remote host must be one sharing the same *AIPS* installation as the current session. Note that the data must also be available to the remote host, but verb ADDDISK may be useful either within the batch job or in preparing the data files for the job. REMOST is supported by all relevant batch-related verbs.

Batch has several limitations. First, devices which require interactive use (TV device, Tektronix device, and the tape drives) cannot be used in batch. Also, batch uses a different set of TPUT and TGET files. Thus, a TGET in batch does not get the adverbs from your last interactive use of the specified task. However, the AIPS facilities GET and SAVE are particularly useful for batch. You can use interactive AIPS to set up and test set(s) of procedures and adverb values and SAVE them in named files. These files may then be recovered by batch for the routine processing of large sets of data. This is considerably more convenient than using the batch editor. Note that SAVE / GET files may become obsolete with new *AIPS* releases, but that improvements to the *POPS* language have made this quite unlikely.

At present, batch jobs are run after a short delay, on a first-come, first-served basis. After your job has been submitted successfully, type:

> QUEUES ↵ to list jobs in the queue.

Note the SUBMIT TIME for your job. It will not start before that time. The messages generated by your batch job will be printed automatically into a text file. They are kept in your message file, however, and can be reprinted or examined later via PRTMSG with PRNUMB set to the *AIPS* number of the batch queue. Printer output for batch jobs is concatenated into a file either specified by the user with OUTPRINT or a file named PRTFIL:BATCH*jjj*.*nnn*, where *jjj* is the job number and *nnn* the user number both in extended hexadecimal. Note, this means batch job printouts are concatenated in (normally) one file and are not automatically printed. An interactive AIPS can interact with a batch job via TELL; see § 5.3.1.

12.3 AIPS language

AIPS contains a basic set of symbols and keywords which are needed to construct a computer language, as well as the symbols needed by the application code. A list of the basic symbols is given in the help file called POPSYM, reproduced below:

```
Type:  Symbols used in the POPS interpretive language

    VERB      USE        COMMENTS

                   ----Arithmetic expressions

     +       A + B       Add the expression A to B
     -       A - B       Subtract the expression B from A
     *       A * B       Multiply the expression A with B
     /       A / B       Divide the expression A by B
     **      A ** B      Calculate A to the power B
     ( )     (A+B)*C     Grouping expressions as desired
     =       A = B       Store the value of B into A
     ,       A = 3,5,4   Separator of elements in an array
     ~       A(i) ~ 1,2,3  Store values in A(i),A(i+1)...
                         (change only as many as on RHS)
```

```
   :       TO          Equivalent to the verb TO
   ;                   Separator between AIPS statements

                ----Logical expressions

   >       A > B       A greater than B
   <       A < B       A less than B
   =       A = B       A equal B (numeric or string)
   >=      A >= B      A equal to or greater than B
   <=      A <= B      A equal to or less than B
   <>      A <> B      A not equal to B (numeric or string)
   !       A ! B       A or B
   &       A & B       A and B
   ^       ^ A         not A

                ----String expressions

!!       A !! B        string = string A followed by string B
SUBSTR   SUBSTR(A,i,j) string = chars i through j of string A
LENGTH   LENGTH(A)     position last non-blank in A
CHAR     CHAR(A)       convert number A to string
VALUE    VALUE(A)      convert string A to number


                ----Looping constructions

(FOR-TO-BY-END)        FOR I=1 TO 7 BY 2
                       <any valid set of AIPS syntax>
                       END

(WHILE-END)            WHILE <any logical expression>
                       <any valid set of AIPS syntax>
                       END

(IF-THEN-ELSE-END)     IF <any logical expression>
                       THEN <any valid set of AIPS syntax>
                       ELSE <any valid set of AIPS syntax>
                       END

                ----Built-in functions

ACOS          Arc-cosine - output in degrees
ASIN          Arc-sine - output in degrees
ATAN          Arc-tangent (one argument) - output in degrees
ATAN2         Arc-tangent (two arguments) - output in degrees
COS           Cosine (degrees)
SIN           Sine (degrees)
TAN           Tangent (degrees)
EXP           Exponential
LN            Log base e
LOG           Log base 10
SQRT          Square-root
MAX           Maximum i.e. X = MAX (A, B)
```

```
MIN           Minimum i.e. X = MIN (A, B)
MODULUS       Root-square sum of two arguments
MOD(A,B)      A - (A/B) * B  i.e. remainder of A/B
CEIL(A)       Lowest integer >= A
FLOOR(A)      Highest integer <= A

                 ----Procedure building verbs

PROC      PV  Begin building a procedure
PROCEDUR  PV  Begin building a procedure
LIST      pV  List a procedure
EDIT      PV  Edit a procedure
ENDEDIT   PV  End editing a procedure
ERASE     PV  Delete line(s) of a procedure
MODIFY    PV  Modify a line in a procedure
RETURN    V   Last statement in a procedure
FINISH    PV  End procedure building

                 ----Variable declarations

SCALAR    pV  Declare scalars
ARRAY     pV  Declare arrays
STRING    pV  Declare strings

                 ----Input/Output functions

PRINT     V   Print the following keyword value(s)
TYPE      V   Print the following keyword value(s)
READ      V   Read value(s) from terminal after # prompt

                 ----Other information

CORE      pV  Amount of core left in POPS
COMPRESS  PV  Compress the core area, recovering lost space
              and acquiring any new vocabulary
CLRTEMP   V   Clear the temp data array
DEBUG     pV  Debug: turns on compiler debug information
DUMP      V   Dump K array on terminal screen
SCRATCH   PV  Remove procedures in POPS
$         PV  Makes rest of input line a comment
```

12.3.1 Using *POPS* outside of procedures

POPS variables are either numeric or character valued and may be multi-dimensional arrays. Once created, all variables are available everywhere, *i.e.*, they are global. You may manipulate these variables on the command line using most of the symbols listed above. In fact, you have been doing this while setting the adverbs for all the tasks and verbs described in preceding chapters. The more advanced user may wish to use some of the language features in order to simplify repetitive data processing. Here are some simple examples of uses of the AIPS language:

```
> TYPE (2 + 5 * 6) CR                    32 is written on the terminal.
> TYPE 'X =', ATAN (1.0) CR              X = 45 is written on the terminal.
```

> TYPE 'MAPNAME ', INNAME, INCLASS, INSEQ ↵ — MAPNAME 3C138 IMAGE 1 is written on the terminal.

The simplest loop capability in AIPS uses the pseudo-verbs FOR, TO, and BY for repetitive operations. Such loops are primarily intended for use in "procedures" (see § 12.3.2). If a FOR loop can be typed fully on one input line, it will also work outside the procedure mode. The following example shows how to delete a series of images with the same name and class and with consecutive sequence numbers 1 through 10:

> INNA 'TEST' ; INCL 'IMAGE' ↵ — to set (fixed) name parts.
> INDI 1 ↵ — to set (fixed) disk number.
> FOR INSEQ = 1 TO 10 ; ZAP ; END ↵ — to delete the files.

FOR loops must be terminated with an END. The following example shows how to delete every other file in a catalog with 20 entries:

> FOR INSEQ = 1 TO 20 BY 2 ; GETN(I) ; ZAP ; END ↵

More extensive examples are shown in the sections below on procedures.

In some cases, you may wish to manipulate character strings to give your files meaningful names — particularly if your RUN file or procedure operates repetitively on many similar files. The verbs for character manipulation are listed above. As an example,

> OUTNAME = 'CLEAN' !! CHAR(BLC(3)) ↵ — to name each output file after the input image plane.

Note that trailing blanks are ignored. If you wanted a space after CLEAN before the plane number, use

> OUTNAME = 'CLEAN' ↵ — to set the basic form.
> SUBSTR (OUTNAME , 7 , 12) = CHAR (BLC(3)) ↵ — to alter only the last six characters of OUTNAME.

12.3.2 Procedures

Procedure building is a way to combine keywords in AIPS in any convenient way to obtain useful constructs. For complicated sequences, it is easier to prepare and debug procedures in RUN files (§ 12.2.1) than to prepare them in interactive *AIPS*. A procedure is given a name, with or without arguments, and then can be treated as an *AIPS* verb. As an example, consider a procedure to load an image on the TV, set the cursor, and fit for the maximum intensity. You could type the following on your terminal:

> PROC MFIT (I) ↵ — to define procedure MFIT with one argument I. (I and J are two dummy adverbs which are already defined in AIPS.)
: GETNAME(I) ↵ — (Notice the prompt symbol : . This means that we are in the procedure-building mode.)
: TVLOD ; IMXY ; MAXFIT ↵ — to load the image, produce and read the cursor, and fit the maximum near the cursor position when a TV button is pressed
: RETURN ↵ — to designate a return point in the procedure — normally not required at the end of a procedure unless a value is to be left on the stack, *i.e.*, a function.
: FINISH ↵ — to designate the end of the procedure-building mode and to get back into the normal (prompt >) mode.
> — Notice the prompt symbol, you are back to interactive input mode.

When you type such a procedure into AIPS, the code is compiled as you type. Most syntax errors are spotted immediately and will unceremoniously dump you out of procedure mode. However, all lines written before the detected error are kept and the procedure editor can be used to continue.

The *AIPS* procedure editing capabilities are quite primitive. If you want to build procedures longer than about five lines, we therefore recommend using permanent storage files in the "RUN" area, as discussed in §12.2.1 above.

To list the procedure MFIT, type:
> LIST MFIT ↵
This will produce the following:

```
1PROC MFIT(I)
2GETNAME(I)
3TVLOD ; IMXY ; MAXFIT
4RETURN
5FINISH
```

The procedure is identical to what you typed, with line numbers added.

Procedures are edited line by line. To edit line 2 in the above procedure, type:

> EDIT MFIT 2 ↵	to enter Procedure editing mode.
; GETNAME(I) ; TVLOD ↵	(Notice prompt symbol ; for procedure-editing mode.) This change replaces the old line 2 adding a TVLOD.
; IMXY ; MAXFIT ↵	to add a line between the changed line 2 and old line 3.
; GETNAME(I+1) ↵	to add yet another line after 2.
; ENDEDIT ↵	to terminate procedure editing.
> LIST MFIT ↵	

Listing the modified procedure will give:

```
1PROC MFIT(I)
2GETNAME(I) ; TVLOD
3IMXY ; MAXFIT
4GETNAME(I+1)
5TVLOD ; IMXY; MAXFIT
6RETURN
7FINISH
```

To delete lines *n* through *m* from a procedure, type:

> ERASE *xxxxxxxx n* : *m* ↵	where *xxxxxxxx* is the name of the procedure.

To insert one or more lines between lines 3 and 4 of a procedure, type:
> EDIT *xxxxxxxx* 3.5 ↵
; *(Type additional lines as needed.)*
; ENDEDIT ↵
Notice that the lines are renumbered after any EDIT or ERASE. Use LIST to determine the new line numbers.

The pseudo-verb MODIFY lets you modify characters within a line of a procedure to correct the line or change its meaning. The grammar is:

> MODIFY *proc-name line-number*	where *proc-name* is the name of the procedure and *line-number* is the line number in the procedure as shown by LIST.

MODIFY begins by showing the existing line with a ? as a prefix. Then it prompts for input with a ? To keep the character of the original line immediately above the cursor, type a blank (space-bar). To delete that character, type a $ (dollar-sign). To replace that character, type the new character (to get a new blank

character, type an @ sign). Insertions complicate things. To insert text prior to the character immediately above the cursor, type a \ followed by the desired text followed by another \. You may continue to MODIFY the remainder of the line, but you must remember that the current character position in the old line is to the left of the current cursor position by the number of inserted characters (including the 2 \'s). MODIFY will display the resulting line of code after you hit a carriage return ($\subset_R$) and does not change the line number. Example:

```
> MODIFY  ED  2 |CR
?TYPE 'THIS IS EDS PROC'
?              MY@\NEW\     @FOR@EXAMPLE' |CR
TYPE 'THIS IS MY NEW PROC FOR EXAMPLE'
> MODIFY  ED  2 |CR
?TYPE 'THIS IS MY NEW PROC FOR EXAMPLE'
?                 $$$$    \EDURE,\ |CR
TYPE 'THIS IS MY PROCEDURE, FOR EXAMPLE'
```

More information about procedure building and editing can be found by typing:

> HELP PROCEDUR $\subset_R$

Procedure creation and editing uses up the limited memory of the *POPS* processor. When the memory is gone, the message BLEW CORE! will appear and you can do no more procedure writing without starting over (*i.e.*, RESTORE 0 $\subset_R$). CORE $\subset_R$ will tell you how much memory is left. If the memory remaining appears small, try COMPRESS to recover the lost memory (in 15JAN96 and later releases). COMPRESS might even work after a BLEW CORE! if you are lucky.

The procedure MFIT can be executed by:

> MFIT(*n*) $\subset_R$ where *n* is the slot number of the appropriate image.

(It is assumed that the correct disk unit number has already been set.) This procedure can also be part of another procedure or put in a loop. For example:

> FOR I= 1 TO 10 BY 2; MFIT(I) ; END $\subset_R$

will load the TV and fit the maximum for the first ten images on the appropriate disk.

All the syntax available in AIPS is available for use inside procedures *except for certain pseudo-verbs*. The "prohibited" pseudo-verbs include SAVE, GET, STORE, RESTORE, PROCEDURE, EDIT, ENDEDIT, MODIFY, LIST, CORE, SCRATCH and COMPRESS. Others do not make much sense in procedures, including MSGKILL, DEBUG, and ABORTASK. Other pseudo-verbs are, however, particularly useful in procedures. These include TGET, TPUT, TUGET, and GO.

Several verbs are extremely useful in procedures. To set the image name adverbs to those visible on the TV, use TVNAME. When GETN accesses an empty slot, an error condition is raised and the procedure dies. To handle this error condition in your procedure, use EGETN *n* instead and test the adverb ERROR which will be "true" if the slot is empty. CHKNAME may be used similarly to check on the existence of files with computed names. Some tasks require image-data dependent inputs. To help handle this in general procedures, the verb GETHEAD allows all header parameters to be fetched into adverbs. Type EXPLAIN GETHEAD $\subset_R$ for details. Verbs GETTHEAD and TABGET perform similar functions for data in table extension files. The verb GETPOPSN fetches the current *POPS* number for use in procedures, for example, to allow the same procedure to run concurrently in multiple *AIPS* sessions. The verb DELAY will cause AIPS to pause for a specified period of time — MFIT above would benefit by pausing to allow the user to see his images. There are numerous arithmetic functions, useful looping constructions, and powerful methods of building arithmetic, logical, and string expressions in *POPS*. See § 12.3 above for a list of these. CLRTEMP may be used in procedures which do a lot of looping. It clears the temporary space used to hold sub-strings and other temporary constants. A procedure that does much string manipulation is likely to overflow this area after a number of iterations. The message BLEW TEMP C! usually accompanies the overflow.

Once a procedure is written and edited, it can be stored in a SAVE file for later use. Procedures are lost when

another GET file is obtained. Procedures can be stored more permanently in RUN files which are described in § 12.2.1 above. To list the names of all procedures currently in your AIPS environment, type:

> HELP PROCS ↵

This will list internal *AIPS* procedures as well as your own.

Several procedures have been built into *AIPS*. In particular, some procedures are defined in the system RUN file VLAPROCS to aid *routine* calibration of VLA data. Currently, these are VLACALIB, VLACLCAL and VLARESET. Similarly, VLBA reductions are aided by the procedures in the file named VLBAUTIL. They may be useful templates for your own procedures. If you are unfamiliar with the use of *AIPS* procedures, looking at these system-supplied ones will help you to understand, and see the power of, this feature of *AIPS*.

12.3.3 Writing your own programs with *POPS*

You may want to add your own programs to *AIPS*. It is not a trivial matter to generate an *AIPS*-standard FORTRAN program (see HELP NEWTASK ↵, the *Going AIPS* manuals, and § 12.6 below). Simple but powerful programs may however be built as procedures that use existing verbs and tasks.

Consider a the following example. (This example is presented as if it were typed into an interactive AIPS. In practice, you will probably prefer to prepare such a complicated procedure as a RUN file.) We wish to determine the average value and rms scatter at any pixel location in a set of n images. We shall demand that the n images all have the same INNAME and INCLASS with sequence numbers between 1 and n. The RENAME verb can be used to name the images appropriately. We could call this procedure:

```
AVGRMS (PIXXY, N, AVG, RMS)
```

where

PIXXY	is the pixel location in the images,
N	is the number of images,
AVG	is the average value at the pixel location, and
RMS	is the rms value at the pixel location.

The array adverb PIXXY is a standard AIPS adverb, but the variables N, AVG, and RMS are unknown to AIPS. These must be defined before we can write the procedure AVGRMS. This is done by a short dummy procedure which we will call DAVGRMS:

> PROC DAVGRMS ↵	to define dummy procedure.
: SCALAR N, AVG, RMS ↵	to define scalar adverbs.
: FINISH ↵	to exit from dummy procedure.

Now begin the procedure AVGRMS:

> PROC AVGRMS (PIXXY, N, AVG, RMS) ↵	to enter procedure building mode.
: SCALAR SUM, SUM2 ↵	to define more variables.
: ARRAY VAL(20) ↵	to define an array.
: RMS = 0 ; SUM = 0 ; SUM2 = 0 ↵	to zero some variables.
: FOR INSEQ =1 TO N ↵	to begin summing loop.
: QIMVAL ↵	to get pixel value at PIXXY in image INNAME INCLASS INSEQ.
: VAL(INSEQ) = PIXVAL ↵	to save pixel value (placed in PIXVAL by IMVAL) in our array.
: SUM = SUM + PIXVAL ↵	to sum for averaging.
: SUM2 = SUM2 + PIXVAL * PIXVAL ↵	to sum for rms.

```
:       END CR                                          to mark end of FOR loop.
:   AVG = SUM / N CR                                    to get average value.
:   IF N > 1.5 THEN CR                                  to check if N > 1.
:       RMS = SQRT((SUM2 - N*AVG*AVG) / (N * (N-1))) CR to calculate rms if N > 1.
:   ELSE ; TYPE 'N TOO SMALL', N CR                     to warn the user.
:       END CR                                          to mark end of IF clause.
:   TYPE 'AVG=',AVG,'RMS=',RMS,'AT PIXEL',PIXXY CR
:   TYPE ' # ',' VAL ',' ERROR ' CR                     to print a heading.
:   FOR INSEQ = 1 TO N CR                               to begin another loop.
:       SUM = AVG - VAL(INSEQ) CR                       to get residual.
:       TYPE INSEQ, VAL(INSEQ), SUM CR                  to print data and residual.
:       END CR                                          to mark end of FOR loop.
:   FINISH CR                                           to return to regular AIPS mode.
>
```

The above procedure could be run as follows. First fill in the adverbs INNAME, INCLASS and PIXXY with the desired values. Then type:

```
> AVGRMS (PIXXY, n, AVG, RMS) CR
```

where n is the number of images to average. The average and rms will be calculated and written on the terminal and in the message file. This procedure could be used by another procedure. Suppose we wanted to determine the average and rms of the pixels within a rectangular area. If we set BLC and TRC in the usual way to define the rectangular boundary, then the procedure:

```
> PROC AVGARRAY (BLC, TRC) CR                         to define new proc.
:   FOR I = BLC(1) TO TRC(1) CR                         to loop over x-coordinate
:       FOR J = BLC(2) TO TRC(2) CR                     to loop over y-coordinate.
:           PIXXY = I , J CR                            to set pixel coordinates for AVGRMS.
:           AVGRMS (PIXXY, N, AVG, RMS) CR
:           END ; END CR                                to end y loop, then x loop.
:   FINISH CR                                           to end the proc., RETURN not needed.
>
```

will calculate the average value and rms at this array of pixel locations. Please note that this is just an example. The verb IMSTAT performs this function much more efficiently.

12.3.3.1 Special facilities for use in procedures

When a task or verb encounters an error condition, it sets an error flag which normally causes *POPS* to terminate the line or procedure it was executing. You can avoid the termination due to task failure by setting DOWAIT FALSE and then doing a WAIT on the task. This does not let your procedure know that an error in the task occurred, but it does let you ignore any possible error. Certain verbs have a second version which sets the ERROR adverb, rather than the automatic termination flag. These include CHKNAME which checks for the existence of an *AIPS* file with specified name parameters, EGETHEAD which returns a keyword value from a header or ERROR if the keyword is not present, EGETNAME which returns ERROR if the catalog slot is empty, COPIXEL which returns ERROR if the coordinate is not in the image, and SYSTEM which returns ERROR if the function fails. Several fitting verbs, IMVAL, MAXFIT, QIMVAL, TVFLUX, and TVMAXFIT, return ERROR if the fit fails. Your procedure can test ERROR after these verbs and take appropriate action.

Numerous verbs return adverb values which may be used in your procedures. These include COPIXEL, COWINDOW, EGETHEAD, EGETNAME, EPOCONV, GAMMASET, the various GETNAMEs, GETHEAD, GETTHEAD, GREAD,

IN2TV, IMDIST, IMPOS, IMSTAT, IMVAL, IMMXY, MAXFIT, QIMVAL, REBOX, SETSLICE, SETXWIN, TABGET, TK1SET, TKBOX, TKNBOXS, TKSET, TKVAL, TKWIN, TV1SET, TVBOX, TVCOLORS, TVDIST, TVFLUX, TVMAXFIT, TVNAME, TVPOS, TVSET, TVSTAT, and TVWINDOW. GETHEAD and the table functions GETTHEAD and TABGET are particularly useful in procedures. The verb NAMEGET will convert the INNAME set of adverbs to the values that would be used by a task.

The RUN file VLBAUTIL provides interesting procedures such as MAXTAB which returns the maximum table number of a specified table type. Other procedures in that RUN file include ones that return antenna number corresponding to a specific name and ones that test whether a data set appears to be new and appears to contain only VLBA antennas. MAXTAB has been made obsolete by the verbs GETVERS and QGETVERS which perform the same function, returning adverb MAXVERS either with or without messages.

Several tasks now return adverb values. These values can then be used for later computation. These tasks include CUBIT, GAL, FINDR, IMEAN, IMFIT, JMFIT, RLDIF, and SETFC. The first two of these return image fit parameters, the next two return uv and image statistics, the next two return the results of Gaussian fitting, RLDIF returns the phase difference between RL and LR polarizations for polarization calibration, and SETFC returns imaging parameters including number of facets, cell size, and image size.

There is also a verb named SYSTEM which allows the *AIPS* user to fork a command to the host operating system from within AIPS including from within procedures. This may be used to delete unwanted text or FITS files, to run ftp to fetch data files from the web, to run another copy of AIPS, or any of a very large number of other commands. The verb FILEZAP will allow you to delete non*AIPS* files without use of the SYSTEM verb.

12.4 Remote use of *AIPS*

AIPS users do not always find themselves seated in front of the main display screen of the computer which they intend to use for their data analysis. *AIPS* provides facilities for such users which depend to some extent on the nature and location of the workstation or terminal at which the user is seated. Nearly seamless function is provided to a user seated at a workstation on the local Ethernet well known to the *AIPS* installation. Substantial capabilities are still available to the user at a more distant workstation capable of X-Windows display, especially if that workstation can also run *AIPS* programs such as the TV server and remote tape server. Even the user at a simple terminal or workstation, capable of emulating a Tektronix 4010, can still get some interactive displays. It is only the users at very simple (and now antique) terminals who will be rather limited in their interactive use of *AIPS*.

12.4.1 Connections via X-Windows

In the following discussion, we will assume that you need to do your computing on a computer called *Server* and that you are sitting in front of a workstation called *MyHost*.

If *Server* and *MyHost* are both on the same local area network and both have the same byte ordering, then they should have *AIPS* installed with both of them shown as being at the same *AIPS* "site." In this case, you simply slogin, rlogin, or telnet into *Server* from a window on *MyHost* and issue the aips command as described in § 2.2.3. The aips command will recognize that you are coming in from *MyHost* only if the $DISPLAY environment variable is correct (*MyHost*:0). In that case, or if you add tv=*MyHost* to the aips command line, the procedure will start the message, graphics, and TV servers on *MyHost* if needed. If you want to share data areas between the two computers, you may add a da=*MyHost* on the command line and AIPS will run with all data areas from both machines. (The disk systems must be auto-mountable between

the two computers.) There are two verbs within AIPS which will allow you the equivalent of da=. They are ADDDISK which will add the disks of a selected computer to the current list of data disks and REMDISK which will remove the disks of a selected computer from the current list. The aips command also lets you select the most convenient printer within your local area network for use in your AIPS session. Other forms of data transfer, including magnetic tapes, will be discussed later.

If *Server* and *MyHost* have different byte orders or are not both on the same local area network, then they cannot be at the same AIPS site. If both machines have AIPS installed and the versions are compatible, then you may run on *Server* with *MyHost* treated as a "guest." Again, slogin, rlogin, or telnet into *Server* from a window on *MyHost*. Make sure that the environment variable DISPLAY on *Server* is set to *MyHost*:0 and that *Server* is mentioned in your .rhosts file on *MyHost*. Then issue the usual aips command. You do not have to give the tv= option, but you may give tv=*MyHost* if you wish.. This will start the message, graphics, and TV servers on *MyHost* if needed. If the servers fail to start and messages such as "Cannot start remote TV servers..." appear, then you must start the servers using the aips command on *MyHost*. The displays will work without restarting AIPS on *Server*. Thereafter, give the aips option tvok to *Server* (rather than tv=) to suppress the annoying messages. There is no drawback to being a guest TV; all catalogs and device information are now maintained by XAS itself. Since the display refreshing is handled locally by programs running in *MyHost*, this level of connection supports nearly full interactivity including such demanding displays as TVBLINK and TVMOVIE.

If *MyHost* does not have AIPS installed, then you may run the message, graphics, and TV servers on *Server*, with the X-Windows $DISPLAY set to *MyHost*:0. You may do this with internet sockets, but this ties up the one instance of the servers allowed to use such sockets. The socially acceptable method uses local Unix sockets so that only current AIPS session(s) on *Server* may talk to the windows in *MyHost*. You must ask for this explicitly, setting tv=local. This mode of operation is not encouraged since the display refreshing has to be transmitted over the network, making some of the interactive displays too slow to be useful. Nonetheless, there are circumstances in which this mode of operation is the only one available. You will have to add *Server* as an allowed X host (xhost +*Server*) to use this option. See HELP AIPS $^{C}_{R}$ for more information on "local" TVs which, among other things, allow for multiple TVs on a single display screen.

12.4.2 Connections to a terminal

You may do some interactive AIPSing if your workstation window is able to emulate a Tektronix 4010 terminal, or you are at a terminal capable of this emulation. (Note that most xterm displays may be switched between a "Tek mode" and the normal "VT mode" by pressing the Control key and the middle mouse button.) To operate in this mode, log in to *Server* from your terminal or workstation window and start AIPS with the command-line option REMOTE. This option will disable all TV functions, will cause all task messages to come to your terminal or window, and will cause any graphics ("TK") displays to be sent to your terminal or window. You may display an "image" by creating a contour drawing with CNTR and then displaying the plot file with TKPL Interactive cursor verbs and procedures such as TKXY, TKPOS, and TKWIN may then be used. The old 4010 Tektronix display mixed plotting and text in a less than elegant fashion which is slavishly honored by most emulations. This is unfortunate, but usually does not prevent the display from being used for simple position selection and the like.

The AIPS task TXPL is a powerful tool for remote users without any, or correct, Tektronix 4010 emulation. It reads an extension file of type PL and translates the graphics commands in that file to an alphanumeric display for a "dumb" terminal. TXPL may be exactly what you need for AIPS applications that depend on scanning the *shape* of a plot rather than its fine detail. Common examples are viewing the shape of visibility functions produced by UVPLT (to guide self-calibration or to diagnose interference) or examining calibration solution plots from SNPLT. TXPL can also usefully interpret simple contour plots or even grey scales(!) for a remote user. It is often much faster to use TXPL to diagnose the state of your AIPS data processing over

a low-bandwidth link than to use TKPL to execute a stream of Tektronix graphics instructions (even if you have full Tektronix 4010 emulation).

12.4.3 File transfer connections

The techniques discussed above apply to many computer configurations and to most tools within AIPS. They do not, however, handle the outputs of printing and plotting tasks. Nor do they provide much support for small computer systems that have no AIPS capability of their own. In such cases, the results of your computations will need to be written onto disk on *Server* and then transferred over the network to *MyHost*.

AIPS does not support remote printers explicitly. However, all AIPS tasks and verbs which generate printer output support the OUTPRINT adverb. With this adverb, you may specify a disk text file to receive the printer text. If you specify the same file for successive printer verbs or tasks, the outputs will be concatenated. (AIPS batch jobs do this automatically to concatenate all printer displays for the job.) You may then copy the text file to *MyHost* for editing, printing, or whatever. If you wish to print the whole file on a PostScript printer on *MyHost*, you may wish to run F2PS on the text file on *Server* and copy the result to *MyHost*. AIPS provides a "filter" program to convert plain (or Fortran) text files to PostScript for printing on PostScript printers. The command

`$ F2PS -`*nn* `<` *file* `>` *outfile*

will convert text file *file* to PostScript format file *outfile*. The parameter *nn* is the number of lines per page used inside AIPS; use 97 for a small font in "portrait" form or 61 for a larger font in "landscape" form.

The AIPS task LWPLA also has an option to write its output to a disk file in encapsulated PostScript form using the OUTFILE parameter. Similarly, TVRGB and TVCPS use the same adverb to write PostScript text files containing their three-color displays. Numerous other AIPS tasks offer the option to write details of the operation to a text file specified with OUTTEXT. These include SLICE (slice), IMEAN (histogram), GAL (fit results), POSSM (spectrum), FRPLT (spectrum), HITEXT (history), UVCRS (*uv*-plane crossings), CONPL (convolving functions), etc.

FITS-disk files may be written with FITTP and FITAB and read with FITLD, UVLOD and IMLOD. For efficiency reasons, these are binary files rather than the text files produced by everything else. AIPS table (and optionally plot) files will have their contents transferred by these tasks along with the main data files. To put an AIPS table in a disk file in text form, use TBOUT to write a simple text file or EXTAB to write a file suitable for database and spreadsheet programs. Files in the form written by TBOUT may be read back into AIPS with TBIN.

To transfer the FITS-disk files and text files between *MyHost* and *Server*, some standard network file transfer must be used. For example, use `ftp` on *MyHost* with

`%` cd *MyArea* ↵	to switch to the disk area on *MyHost* used for your files.
`%` ftp *Server* ↵	to start `ftp` to the remote system.
`Name (`*Server*`:...):` *loginame* ↵	to log in to account *loginame*.
`Password:` *password* ↵	to give the account's password.
`ftp>` cd *directory* ↵	to change to the *directory* name containing the file on *Server*.
`ftp>` binary ↵	to allow reading of a binary file — required for FITS-disk files, okay for text files.
`ftp>` hash ↵	to get progress symbols as the copy proceeds — a good idea for large files.
`ftp>` put *filename* ↵	to send *filename* from *MyHost* to *Server*.
`ftp>` get *anothername* ↵	to send *anothername* from *Server* to *MyHost*.
`ftp>` quit ↵	to exit from `ftp`.

The files should then be in the desired directories. You may have to rename them, however, to a name in all upper-case letters unless you use the "trick" mentioned in § 3.10.1. The secure copy (`scp`) is preferable if you have a secure connection set up. The files may be compressed with `gzip` before copying and then uncompressed with `gunzip` at the other end. This is particularly effective on text files and images written by FITAB with quantization.

12.5 Moving data to a new computer

Lucky users who get a new computer frequently ask "how do I move my data to my new, better machine?" The easiest thing to do is first to install *AIPS* on the new computer. Create at least as many data areas on the new machine as there were on the old. If you are moving between machines of the same byte order (same architecture or Solaris to Mac PPC, Linux to Mac Intel, or vice versa), then do a network `scp` copy (or easier yet a cp from cross-mounted disks) to move all data files from *AIPS* disk 1 on the old machine to *AIPS* disk 1 on the new machine. Repeat for the other *AIPS* disks. Note that *AIPS* disk 1 is different; it contains message, SAVE/GET, and other special files as well as normal data files. The other data areas may be rearranged if you want, but do not copy two old data areas to one new data area. That will cause extreme confusion and loss of data. Now you are ready to resume *AIPS*ing.

But if you are moving from Solaris or Mac PPC to Linux or Mac Intel, or vice versa, then you have more to do. Mount the *AIPS* disks of the old machine from the new machine if possible. Otherwise copy the data from the old machine to new, temporary data areas on the new machine (not the new *AIPS* data areas) being careful to keep each data area separate. Outside of AIPS, setup the *AIPS* environment:

% cd $AIPS_ROOT ↵	to move to the root of all *AIPS*.
% source LOGIN.CSH ↵	to set the basic environment under C-shell, or
% . LOGIN.SH ↵	to set the basic environment under bash — note the dot.
% $CDTST ↵	to set the full environment.
% RUN REBYTE ↵	to run a format converter once for each *AIPS* data area.

Answer the question about range of user number (`0 0` ↵ will get you all possible numbers) and then enter the full pathname of the input (old) data area and the corresponding new *AIPS* data area. The program will convert the format of all files. The program — barring software error — will produce correct output for almost all files. There is a small danger that the SAVE/GET and TPUT/TGET files may have some errors since it is way beyond the scope of REBYTE to understand the format of these files.

The traditional way to move between computer architectures is more time consuming. On the old computer, use procedure WRTDISK to write a data area out as FITS files. The FITS file names encode the old *AIPS* disk and catalog numbers so that more than one disk can be written into the same FITS area. Copy the files to the new computer (or better yet, cross mount the disks) and then use procedure READISK on the new computer to read them into the new data area(s). These procedures are obtained via a RUN WRTPROCS. Plot and slice files no longer are lost in this process if you set DOPLOT to true in FITTP and FITAB starting in 31DEC25. You can move the SAVE/GET files with the verb SG2RUN which converts an SG file to a text file which can be RUN on the new computer and then SAVEd. This traditional method works fairly well and can deal with the new computer having fewer *AIPS* data areas than the old one. It also offers the opportunity to do a probably overdue backup of your data. However, the loss of message files and the added overhead in disk space may matter to you.

12.6 Adding your own tasks to AIPS

This Section is a brief guide for the user who wants to modify an existing task in AIPS or to write a new task. While it is difficult to write an AIPS task from scratch, AIPS contains several template tasks that are designed to hide most of the work from the "occasional" programmer. Anyone familiar with FORTRAN and a little of the system services of their local computer should be able to add convenient tasks to their local version of AIPS with a little practice and patience. The growing use of binary installations makes this more difficult since local sites are unlikely to have the relevant compilers.

12.6.1 Initial choices to make

The simplest way to write an AIPS task is to modify one of the four template tasks, TAFFY, CANDY, FUDGE, OFUDG, or UVFIL. These tasks handle the AIPS I/O, contain extensive documentation, need to be modified in only a few well-defined places, and can be easily interfaced to user subroutines. They are limited in their ability to handle many input and output images, however, and generally operate on one image row or one visibility point at a time. Softening of these limitations will be discussed in § 12.6.5.

TAFFY — This template task reads an existing image in AIPS line by line, modifies the line of data as desired and then creates and writes the modified image in AIPS. This template task might, for example, be used to blank pixels in an input image in some specified manner.

CANDY — This task is similar to TAFFY except the input data are contained in an auxiliary file which is FORTRAN readable and outside AIPS. The task can transfer an image in any reasonable format outside AIPS into the AIPS data structure.

FUDGE — This template task reads an existing AIPS *uv* database optionally applying calibration, editing, and data selection, modifies the results point by point, and then creates and writes this modified output *uv* database in the AIPS catalog. This template task might, for example, be used to modify a *uv* data base for which the time parameter has been incorrectly written. OFUDG is an older version of FUDGE which omits all the complications of the calibration system.

UVFIL — This task is similar to FUDGE without calibration and editing, except that the input *uv* data are contained in one or two auxiliary files, which are FORTRAN readable and outside AIPS. The task is useful for translating visibility data in an arbitrary format into an AIPS cataloged *uv* data set, or for computing a *uv* data set from model sources.

If you wish to make a minor change in an existing AIPS task, it will be simpler to copy the AIPS task itself and modify it as needed. For example, if you wanted to add a option in COMB to combine two input images in a new way to obtain a resultant image, it would be better to start with COMB itself than with one of the templates. However, non-trivial changes to major tasks will require a careful look at the code, which is generally well documented and segmented.

Changes to tasks that make plots and/or use the TV and other graphics devices can be tricky, but are useful sometimes. Changes to imaging and deconvolution software should not be attempted unless the changes are almost trivial. Modifying existing verbs or creating new ones require you to find which AU*xx*.FOR routine is involved and then require the entire AIPS program to be re-linked. *Going AIPS* describes some of the considerations. The new pseudo-verbs VERB and PSEUDOVB allow you to update your AIPS vocabulary using your own procedure rather than by changing the supplied vocabulary files and executing POPSGN as described in *Going AIPS*. In all cases, do not put your modified code into the standard AIPS code areas unless your local AIPS Manager agrees and has made a backup copy of the original code and executables. Such modified files may well disturb the AIPS updates done by the so-called "Midnight Job."

12.6.2 Getting started

Even if you have special privileges in AIPS, it is wise to generate and link new code in your own user directory rather than in your installation's designated local AIPS directory. After the code has been written and tested, check with your local AIPS Manager about installing it in a public area. Decide which template task or other AIPS task you want to modify, then follow these instructions after logging into your private area.

Under Unix

% source /AIPS/LOGIN.CSH ↵	to get the basic AIPS system logicals under a c-shell, or
% . /AIPS/LOGIN.SH ↵	to get the basic AIPS system logicals under a korn, bourne, or bash shell. (These assume that your AIPS system home directory is called `/AIPS`).
% $CDTST (or $CDNEW) ↵	to set up the logical assignments for programming.
% PROG *TASK* ↵	to locate the task called *TASK*.
% LIBS $*AREA* > *NTASK*.OPT ↵	to create a file of linking information for a task found in area *AREA*, *e.g.*, `APLPGM` for `OFUDG` and `TAFFY`, `APGNOT` for `FUDGE`, `CANDY` and `UVFIL`, `YPGM` for `IMEAN`, etc.
% cp $*AREA*/*TASK*.FOR *NTASK*.FOR ↵	to copy code for *TASK* into your area and to give it a new name (≤ 5 letters).
% cp $HLPFIL/*TASK*.HLP *NTASK*.HLP ↵	to copy and rename the inputs/help file for the task.
% setenv MYAIPS 'pwd' ↵	to define `MYAIPS` (must be uppercase) as your disk area for AIPS programs, or
% export MYAIPS='pwd' ↵	to define the `MYAIPS` environment variable under korn, bash, and bourne shells.

12.6.3 Initial check of code and procedures

Before modifying the task in any way, it is wise to compile, link and execute the unchanged task to check that the original code is sound and that all of the AIPS logical assignments have been properly set. The duplicated AIPS task should run identically to the original; the template tasks are set up to duplicate the input data set with no changes and default parameters. In this way errors or other problems associated with the generation of new tasks can be found and corrected before getting into the quagmire of bugs that you are about to add. Note that `CANDY` and `UVFIL` require external data files and are therefore not so easily checked.

One change is needed in the FORTRAN program before checking. In the main program, about 60 lines down, there is a data statement which identifies the task.

`DATA PRGM /'`*TASK*` '/`

Change this to

`DATA PRGM /'`*NTASK*`'/`

The maximum number of characters in a task name is five and the `DATA` statement must be in the above form. There is no need to change the help file yet, as long as it is named `NTASK.HLP`.

To compile and link the task, type:

% COMLNK *NTASK* *NTASK*.OPT ↵

To compile the task for debugging, add the `DEBUG NOOPT` options to the `COMLNK` line. There should be no error messages and no significant warnings.

After successful compilation and linking, try executing the task. Under Unix, stay logged in to your area. To initiate the AIPS program, type:

% aips ↵	or
% aips debug ↵	to use the debugger (*e.g.*, dbx) on tasks compiled with the DEBUG option. You will have to tell the procedure which debugger you want and that you do not want to start AIPS itself under that debugger.

Your *AIPS* Manager may have to make some arrangements for you to activate *AIPS* from your login. Once you have started *AIPS*, type

> VERSION 'MYAIPS' ↵	to specify the location of inputs and help information and of the task executable.

Set up the input parameters, then

> INP ↵	to review the input parameters. You should be told AIPS 1: Found in Version=MYAIPS at the start of the inputs display. If not, there is something wrong with the logical or task name or the location of the inputs/help file.
> GO ↵	to run the task. The Found ... line should appear again.

12.6.4 Modifying an *AIPS* task

If you are modifying an existing task, the only ground rules are to read the task carefully and note the locations where changes must be made. Unlike the template tasks which are organized so that additional code need be inserted in only one or two places, an *AIPS* task may need revisions in a variety of places. Some guidelines are:

1. Change the data statement DATA PRGM ... near the beginning of the program and the PROGRAM statement at the very beginning, if you have not already done this.
2. Make changes in the introductory text which describes the task. This is particularly important if you are adding or removing adverbs.
3. The subroutine GTPARM obtains the adverb values from the input table in order. If you add or subtract adverbs, change INPRMS, the amount of input information. Be careful in the translation of the adverbs, which are usually listed in a COMMON named /INPARM/.
4. Change code as desired. Try to modify HIstory file entries as well.
5. Liberally sprinkle PRINT statements in crucial places to help debug the program. If much of the new code can be put in a subroutine, write and debug the subroutine outside *AIPS*. Then, add it to the task.
6. Revise the file NTASK.HLP. At least, change all references to TASK to NTASK! If there are any changes in the adverb list, look at these changes carefully. Further changes in the HELP and EXPLAIN portions of the file may be needed to document your work for for others and for yourself at a later date.
7. Compile and link the modified task following the instructions in the previous section. When it compiles and links without errors, try it out in *AIPS*.

12.6.5 Modifying an AIPS template task.

The template tasks are:

1. TAFFY — modifies an existing image file and writes a new image.
2. CANDY — writes a new image. Input data from outside AIPS.
3. FUDGE — modifies an existing *uv* database and writes a new database with calibration, flagging, source selection, etc.
4. OFUDG — modifies an existing *uv* database and writes a new database.
5. UVFIL — writes a new *uv* database taking input data from outside AIPS.

These tasks are described in detail in Chapter 2 of *Going AIPS*. The template tasks and the code for each are extensively documented there, so only the major points are discussed here.

TAFFY reads a selected subset of an image, one row at a time, to a user interface subroutine DIDDLE. An output image is created, cataloged and filled with values calculated in DIDDLE (or attached subroutines). The dimensionality of the output image need not be the same as the input image.

1. If the output image has identical dimensions to the input image, and an output image row can be generated from each input image row, then TAFFY is relatively straight-forward to use and the accompanying documentation should suffice.
2. TAFFY has the option of deferring the writing of an output row until some number of input rows have been read. This would allow the output to depend on some function of several, or even all, input rows. Examples would include smoothing in x and y and other sorts of spatial filtering.
3. It is possible to handle several input images and several output images with TAFFY by using multi-dimensional images and the axis transposing task, TRANS. Suppose you want to calculate the spectral index from a set of four images. First, use FQUBE to put the four images into one data cube. (FQUBE lets you avoid re-defining the frequency axis as MCUBE would require.) Then, transpose the cube so that the frequency axis is first with right ascension and declination as the second and third axes. Use this image as the input to TAFFY and use CPARM to specify the frequency values. Each input row will then be the intensity at a pixel for the four frequencies — from which the spectral index and other parameters can be calculated. The output dimension can be specified arbitrarily. For example, you might want to write out the spectral index, the error, the curvature and the flux intercept at some fiducial frequency. When the task has completed, transpose the output cube so that the celestial coordinates are again the first two axes and the spectral index and friends are the third axis.
4. Subroutine NEWHED may be modified to require certain axis types, operation codes, etc. It must be used to change the output image dimensions if they are not to be the same as the input. An option to omit from the output image the first input axis is available and must be selected in NEWHED.
5. If you can write and debug outside AIPS any subroutines that DIDDLE will call in calculating the output image, you may speed up the debugging of your algorithms.

CANDY lets you create an image one row at a time. Input information is obtained through a file which is external to AIPS.

1. Subroutine NEWHED shows an example of how the external file may be defined and how it can be used. The first few records of the external file should contain information for defining the header of the output image and updating the appropriate catalog blocks. The pointers to the AIPS catalog are given in Chapter 5 of *Going AIPS*.

2. The subroutine MAKMAP reads further records from the external file until a full row of the output image is obtained.

3. Many adverbs are built into CANDY. If you need more or wish to change them, read the information at the beginning of subroutine CANIN.

OFUDGE reads an existing *uv* database point by point and creates a new *uv* database with the modified data. It can easily be used to make simple changes in a *uv* database; for example, if the u, v, or w terms are to be recalculated, if all phases are to be changed in sign, etc. FUDGE does the same but with the full set of calibration, flagging, source selection, etc. adverbs.

1. If the output data set has different dimensions than the input data set (*e.g.*, by combining several spectral-line channels), changes must be made in the subroutine FUDGIN.

2. To combine several input data points, use the task UVSRT to put the data points adjacent in the file.

3. If you wish to calculate quantities from an input *uv* data set without creating an output file, use this task. In the subroutine DIDDLE set IRET = -1 to avoid writing an the output *uv* data set.

UVFIL reads input from one or two FORTRAN-readable files which are outside *AIPS* to create, catalog, and fill a *uv* database into *AIPS*. This task is useful for transcribing *uv* data in an arbitrary format into an *AIPS* database.

1. The task code has copious notes to help the user. The first auxiliary file should contain header information about the observations and the telescopes. This file is read in the subroutine NEWHED.

2. The second auxiliary file contains the actual *uv* data in some format and it is converted into an *AIPS* *uv* database, point by point, in the subroutine FIDDLE. The comments are extensive and an example is given in the code.

12.6.6 Further remarks

1. Try to use the *AIPS* coding standards as described in *Going AIPS*. This will make your code more readable and more portable. It will also save other *AIPS* programmers lots of work if your new task comes into general use.

2. Declare all variables that you use.

3. While debugging, use the FORTRAN PRINT *, ... statement in your code to obtain temporary output on your terminal during execution. If you want more permanent output, use the *AIPS* message file facilities with the appropriate message level (4 is recommended for information, 6 for warnings, 8 for errors).

4. You may use a debugger (such as dbx) on *AIPS* tasks by specifying DEBUG to the COMLNK and aips procedures. Inside AIPS, set the "hidden" adverb SETDEBUG to 0 for normal operation and to 20 to run tasks with your specified debugger.

13 Current *AIPS* Software

The complete lists of software in *AIPS* are kept up-to-date in certain special files which may then be accessed with the AIPS verb ABOUT. Semi-automatic software makes these listing files using the primary and secondary keywords entered by *AIPS* programmers in the numerous help files. The explanations given for each symbol are also those entered by the programmers as the "one-liner" descriptions of the symbol for which the help file is written. The lists of primary and secondary keywords may be viewed directly by typing:

> HELP CATEGORY ↵ to view primary keywords

> HELP SECONDARY ↵ to view secondary keywords

The help file for ABOUT is more explanatory, however. The general help file (for HELP itself) lists a number of general help files which will be of interest. These are also mentioned in the section called INFORMATION below.

The following sections are verbatim reproductions of the various listing files used by ABOUT and are roughly current to the current release of *AIPS*. Each section title is the name of the keyword which is used as the ABOUT topic. Each line within a section lists a task, verb, pseudoverb, procedure, adverb, or RUN file with a very brief description of its function. Pseudoverbs come in two flavors: those that act roughly like verbs and those that must be treated specially, *i.e.*, that must appear alone on a line or only in certain contexts. The help file for each pseudoverb should clarify the its grammatical limits. Typing

> HELP *name* ↵

where *name* is one of the entries in the left-hand column, will give more useful information about that *AIPS* symbol.

13.1 ADVERB

```
ADVERB
Type:  General type of POPS symbol
Use:   Adverbs are the symbols used to address values.  They
       may be REAL (single-precision floating point), ARRAY
       (multiply-dimensioned REALs), or STRING (character
       strings with or without subscripts).  The user may
       create new adverbs by defining them while typing or
       editing procedures.
Grammar:   Adverb names may be used either in compile mode
       or in regular execute mode.  In the former, their
       pointers are compiled with the procedure and their
       values, at the time the procedure is invoked, are
       used during the execution of the procedure.
Usage examples:
       ARRAY2 = ARRAY1
       ARRAY3(I,J) = 23.6
       CHAN = 4
       STRARRY = 'STR1','STR2','STR3',STRAR2
       DUM3 (24, I, STRARRY)
    =====>  Character string data must be enclosed in quotes!
    =====>  This allows the compiler to tell data from adverb
            names and allows embedded special characters.
************************************************************
List of ADVERBs
```

ALIAS	adverb to alias antenna numbers to one another
ALLOKAY	specifies that initial conditions have been met.
ANTENNAS	Antennas to include/exclude from the task or verb
ANTNAME	A list of antenna (station) names
ANTUSE	Antennas to include/exclude from the task or verb
ANTWT	Antenna Weights for UV data correction in Calibration
APARM	General numeric array adverb used many places
ARRAY1	General scratch array adverb
ARRAY2	General scratch array adverb
ARRAY3	General scratch array adverb
ASDMFILE	Full path to EVLA ASDM/BDF directory
ASPMM	Plot scaling parameter - arc seconds per millimeter on plot
AVERSION	Specify a version of the ALBUS container
AVGCHAN	Controls averaging of spectral channels
AVGIF	Controls averaging of IF channels
AVOPTION	Controls type or range of averaging done by a task
AX2REF	Second reference pixel number
AXINC	Axis increment - change in coordinate between pixels
AXREF	Reference pixel number
AXTYPE	Type of coordinate axis
AXVAL	Value of axis coordinate at reference pixel
BADDISK	specifies which disks are to be avoided for scratch files
BAND	specifies the approximate frequency of UV data to be selected
BANDPOL	specifies polarizations of individual IFs
BASELINE	specifies which antenna pairs are to be selected/deselected
BATFLINE	specifies starting line in a batch work file
BATNLINE	specifies the number of lines to process in a batch work file
BATQUE	specifies the desired batch queue
BB	spare adverb for use in procedures
BCHAN	sets the beginning channel number
BCOMP	gives beginning component number for multiple fields
BCOUNT	gives beginning location for start of a process
BDROP	gives number of pooints dropped at the beginning
BIF	gives first IF (spectral window) to be included
BITER	gives beginning point for some iterative process
BLC	gives lower-left-corner of selected subimage
BLOCKING	specifies blocking factor to use on e.g. tape records
BLVER	specifies the version of the baseline-calibration table used
BMAJ	gives major axis size of beam or component
BMIN	gives minor axis size of beam or component
BOX	specifies pixel coordinates of subarrays of an image
BOXFILE	specifies name of Clean box text file
BPA	gives position angle of major axis of beam or component
BPARM	general numeric array adverb used too many places
BPASS5	Normalization control in BPASS for VLBARUN
BPASSPRM	Control adverb array for bandpass calibration
BPRINT	gives beginning location for start of a printing process
BPSOLINT	Sets SOLINT in BPASS for VLBARUN
BPVER	specifies the version of the bandpass table to be applied
BWSMEAR	amount of bandwidth smearing correction to use
CALCODE	specifies the type of calibrator to be selected
CALIN	specifies name of input disk file usually with calibration data
CALSOUR	specifies source names to be included in calibration
CALTASK	Controls which calibration task is used when by VLBARUN
CATNO	Specifies AIPS catalog slot number range
CBPLOT	selects a display of a Clean beam full width at half maximum
CCBOX	specifies pixel coordinates of subarrays of an image
CELLSIZE	gives the pixel size in physical coordinates
CHANNEL	sets the spectral channel number
CHANSEL	Array of start, stop, increment channel numbers to average
CHINC	the increment between selected channels
CLBOX	specifies subarrays of an image for Clean to search
CLCORPRM	Parameter adverb array for task CLCOR

CLEV	Contour level multiplier in physical units
CLINT	CL table entry interval
CMETHOD	specifies the method by which the uv model is computed
CMODEL	specifies the method by which the uv model is computed
CODETYPE	specifies the desired operation or data type
COLORS	specifies the desired TV colors
COMMENT	64-character comment string
CON3COL	Controls use of full 3-color graphics for contouring
CONFIG	Configuration ID number within an EVLA ASDM/BDF data set
COOINC	Celestial axes increment: change in coordinate between pixels
COORDINA	Array to hold coordinate values
COOREF	Reference pixel number for two coordinate axes
COOTYPE	Celestial axes projection type
COPIES	sets the number of copies to be made
CPARM	general numeric array adverb used many places
CROWDED	allows a task to perform its function in a crowded fashion
CTYPE	specifies type of component
CUTOFF	specifies a limit below or above which the operation ends
DARKLINE	The level at which vectors are switched from light to dark
DATA2IN	specifies name of input FITS disk file
DATAIN	specifies name of input FITS disk file
DATAOUT	specifies name of output FITS disk file
DCHANSEL	Array of start, stop, increment channel #S + IF to avoid
DCODE	General string adverb
DDISK	Deterimins where input Y2K data are found
DECIMAL	specifies if something is in decimal format
DECSHIFT	gives Y-coordinate shift of an image center from reference
DEFER	Controls when file creation takes place
DELCORR	Controls task actions on calibration tables.
DELTAX	Increment or size in X direction
DELTAY	Increment or size in Y direction
DENSITY	gives the desired tape density
DENUMB	a scalar decimal number
DETIME	specifies a time interval for an operation (destroy, batch)
DIGICOR	specifies whether VLBA digital corrections are to be applied
DIST	gives a distance
DO3COLOR	Controls whether full 3-color graphics are used in a plot
DO3DIMAG	specifies whether uvw's are reprojected to each field center
DOACOR	specifies whether autocorrelation data are included
DOALIGN	specifies how two or more images are aligned in computations
DOALL	specifies if an operation is done once or for all matching
DOALPHA	specifies whether some list is alphabetized
DOAPPLY	Flag to indicate whether an operation is applied to the data
DOARRAY	spcifies if subarrays are ignored or the information used
DOBAND	specifies if/how bandpass calibration is applied
DOBLANK	controls handling of blanking
DOBTWEEN	Controls smoothing between sources in calibration tables
DOCALIB	specifies whether a gain table is to be applied or not
DOCAT	specifies whether the output is saved (cataloged) or not
DOCELL	selects units of cells over angular unit
DOCENTER	selects something related to centering
DOCIRCLE	select a "circular" display (i.e. trace coordinates, ...)
DOCLIP	Controls whether and how a CLIP operation is performed
DOCOLOR	specifies whether coloring is done
DOCONCAT	selects concatenated or indivudual output files
DOCONFRM	selects user confirmation modes of repetitive operation
DOCONT	selects a display of contour lines
DOCRT	selects printer display or CRT display (giving width)
DODARK	specifies whether "dark" vectors are plotted dark or light
DODATE	specifies if the dat and time are to be used in some way

DODELAY	selects solution for phase/amplitude or delay rate/phase
DOEBAR	Controls display of estimates of the uncertainty in the data
DOEOF	selects end-of-file writing or reading until
DOEOT	selects tape positioning before operation: present or EOI
DOFIT	Controls which antennas are fit by what methods
DOFLAG	Controls closure cutoff in gain solutions and flagging
DOFRACT	Tells whether to compute a fraction or ratio
DOGREY	selects a display of a grey-scale image
DOGRIDCR	selects correction for gridding convolution function
DOHIST	selects a histogram display or controls writing of history file
DOHMS	selects sexagesimal (hours-mins-secs) display format
DOIFS	controls functions done across IFs
DOINVERS	selects opposite of normal function
DOKEEP	specifies if something is kept or deleted
DOMAX	selects solutions for maxima of models
DOMODEL	selects display of model function
DONEWTAB	do we make new tables, use a new table format, etc.
DOOUTPUT	selects whether output image or whatever is saved / discarded
DOPLOT	Controls plotting of something
DOPOL	selects application of any polarization calibration
DOPOS	selects solutions for positions of model components
DOPRINT	selects printer display or CRT display (giving width)
DOPROFIL	selects display of average across rows, columns, planes of an image‘
DOREAL	Say to do operation with real/imaginary rather than amplitude/phase
DORESID	selects display of differences between model and data
DOROBUST	Controls method of averaging - simple mean/rms or robust
DOSCALE	specifies if a scaling operation of some sort is to be performed
DOSCAN	specifies if a scan-related operation is to be done
DOSLICE	selects display of slice data
DOSPIX	selects solutions for spectral index of model components
DOSTOKES	selects options related to polarizations
DOTABLE	selects use of table-format for data
DOTV	selects use of TV display option in operation
DOTWO	do we make two of something
DOUVCOMP	selects use of compression in writing UV data to disk
DOVECT	selects display of polarization vectors
DOWAIT	selects wait-for-completion mode for running tasks
DOWEDGE	selects display of intensity step wedge
DOWEIGHT	selects operations with data weights
DOWIDTH	selects solution for widths of model components
DPARM	General numeric array adverb used many places
ECHAN	define an end for a range of channel numbers
ECOUNT	give the highest count or iteration for some process
EDGSKP	Deterimins border excluded from comparision or use
EDROP	number of points/iterations to be omitted from end of process
EFACTOR	scales some error analysis process
EHNUMB	an extended hexadecimal ”number”
EIF	last IF (spectral window) number to be included in operation
EPRINT	gives location for end of a printing process
ERROR	was there an error
EXPERT	specifies an user experience level or mode
FACTOR	scales some display or CLEANing process
FGAUSS	Minimum flux to Clean to by widths of Gaussian models
FITOUT	specifies name of output text file for results of fitting
FLAGVER	selects version of the flagging table to be applied
FLDSIZE	specifies size(s) of images to be processed
FLMCOMM	Comment for film recorder image.
FLUX	gives a total intensity value for image/component or to limit
FMAX	specifies peak values of model components - results of fits
FORMAT	gives a format code number: e.g. FITS accuracy required
FOV	Specifies the field of view
FPARM	General numeric array adverb used in modeling

FPOS	specifies pixel positions of fit model components
FQCENTER	specifies that the frequency axis should be centered
FQTOL	Frequency tolerance with which FQ or frequency entries are accepted.
FREQID	Frequency Identifier for frequency, bandwidth combination
FREQSEL	List of frequencies desired
FSHIFT	specifies a position shift - output from fitting routines
FSIZE	file size in Megabytes
FUNCTYPE	specifies type of intensity transfer function
FWIDTH	gives widths of model components - results of fitting
GAIN	specifies loop gain for deconvolutions
GAINERR	gives estimate of gain uncertainty for each antenna
GAINUSE	specifies output gain table or gain table applied to data
GAINVER	specifies the input gain table
GCVER	specifies the version of the gain curve table used
GG	spare scalar adverb for use in procedures
GMAX	specifies peak values of model components
GPOS	specifies pixel positions of model components
GR2CHAN	specifies a second TV graphics channel to be used
GRADDRES	specifies user's home address for replies to gripes
GRCHAN	specifies the TV graphics channel to be used
GREMAIL	gives user's e-mail address name for reply to gripe entry
GRNAME	gives user's name for reply to gripe entry
GRPHONE	specifies phone number to call for questions about a gripe
GUARD	portion of UV plane to receive no data in gridding
GWIDTH	gives widths of model components
HIEND	End record number in a history-file operation
HISTART	Start record number in a history-file operation
I	spare scalar adverb for use in procedures
ICHANSEL	Array of start, stop, increment channel #S + IF to average
ICUT	specifies a cutoff level in units of the image
IM2PARM	Specifes enhancement parameters for OOP-based imaging: 2nd set
IMAGRPRM	Specifes enhancement parameters for OOP-based imaging
IMSIZE	specifies number of pixels on X and Y axis of an image
IN2CLASS	specifies the "class" of the 2nd input image or data base
IN2DISK	specifies the disk drive of the 2nd input image or data base
IN2EXT	specifies the type of the 2nd input extension file
IN2FILE	specifies name of a disk file, outside the regular catalog
IN2NAME	specifies the "name" of the 2nd input image or data base
IN2SEQ	specifies the sequence # of the 2nd input image or data base
IN2TYPE	specifies the type of the 2nd input image or data base
IN2VERS	specifies the version number of the 2nd input extension file
IN3CLASS	specifies the "class" of the 3rd input image or data base
IN3DISK	specifies the disk drive of the 3rd input image or data base
IN3EXT	specifies the type of the 3rd input extension file
IN3NAME	specifies the "name" of the 3rd input image or data base
IN3SEQ	specifies the sequence # of the 3rd input image or data base
IN3TYPE	specifies the type of the 3rd input image or data base
IN3VERS	specifies the version number of the 3rd input extension file
IN4CLASS	specifies the "class" of the 4th input image or data base
IN4DISK	specifies the disk drive of the 4th input image or data base
IN4NAME	specifies the "name" of the 4th input image or data base
IN4SEQ	specifies the sequence # of the 4th input image or data base
IN4TYPE	specifies the type of the 4th input image or data base
IN4VERS	specifies the version number of the 4th input extension file
IN5CLASS	specifies the "class" of the 5th input image or data base
IN5DISK	specifies the disk drive of the 5th input image or data base
IN5NAME	specifies the "name" of the 5th input image or data base
IN5SEQ	specifies the sequence # of the 5th input image or data base
IN5TYPE	specifies the type of the 5th input image or data base
IN5VERS	specifies the version number of the 5th input extension file
INCLASS	specifies the "class" of the 1st input image or data base
INDISK	specifies the disk drive of the 1st input image or data base
INEXT	specifies the type of the 1st input extension file

INFILE	specifies name of a disk file, outside the regular catalog
INLIST	specifies name of input disk file, usually a source list
INNAME	specifies the "name" of the 1st input image or data base
INSEQ	specifies the sequence # of the 1st input image or data base
INTAPE	specifies the input tape drive number
INTERPOL	specifies the type of averaging done on the complex gains
INTEXT	specifies name of input text file, not in regular catalog
INTPARM	specifies the parameters of the frequency interpolation function
INTTIME	adds user-generated lines to the history extension file
INTYPE	specifies the type of the 1st input image or data base
INVERS	specifies the version number of the 1st input extension file
IOTAPE	Deterimins which tape drive is used during a Y2K RUN
ISCALIB	states that the current source is a point-source calibrator
J	spare scalar adverb for use in procedures
JOBNUM	specifies the batch job number
KEYSTRNG	gives contents of character-valued keyword parameter
KEYTYPE	Adverb giving the keyword data type code
KEYVALUE	gives contents of numeric-valued keyword parameter
KEYWORD	gives name of keyword parameter - i.e. name of header field
KK	Spare adverb for use in procedures
LABEL	selects a type of extra labeling for a plot
LEVS	list of multiples of the basic level to be contoured
LPEN	specifies the "pen width" code # =¿ width of plotted lines
LTYPE	specifies the type and degree of axis labels on plots
MAPSHIFT	used by MAPPR to specify the shift of the image from the phase center
MAXPIXEL	maximum pixels searched for components in Clark CLEAN
MAXVERS	The maximum version number of a specified extension file type
MDISK	Determins where input Y2K data are found
MINAMPER	specifies the minimum amplitude error prior to some action
MINANTEN	states minimum number of antennas for a solution
MINPATCH	specifies the minimum size allowed for the center of the beam
MINPHSER	specifies the minimum phase error prior to some action
MPDIFS	Retains comparison differences between current images and master ones
NAXIS	Axis number
NBOXES	Number of boxes
NCCBOX	Number of clean component boxes
NCHAN	Number of spectral channels in each spectral window
NCHAV	Number of channels averaged in an operation
NCOMP	Number of CLEAN components
NCOUNT	General adverb, usually a count of something
NDIG	Number of digits to display
NFIELD	The number of fields imaged
NFILES	The number of files to skip, usually on a tape.
NGAUSS	Number of Gaussians to fit
NIF	Number of IFs (spectral windows) in a data set
NITER	The number of iterations of a process.
NMAPS	Number of maps (images) in an operation
NOISE	estimates the noise in images, noise level cutoff
NORMALIZ	specifies the type of gain normaliztion if any
NPIECE	The number of pieces to make
NPLOTS	gives number of plots per page or per job
NPOINTS	General adverb giving the number of something
NPRINT	gives number of items to be printed
NSKIP	Number of objects to skip
NTHREAD	Controls number of threads used by multi-threaded processes in OBIT
NUMTELL	selects POPS number of task which is the target of a TELL or ABORT
NX	General adverb referring to a number of things in the X direction
NY	General adverb referring to a number of things in the Y direction
OBJECT	The name of an object
OBOXFILE	specifies name of output Clean box text file
OFFSET	General adverb, the offset of something.
OFMFILE	specifies the name of a text file containing OFM values

OLDVOL	used by READISK to specify the disk number when WRITDISK was run
ONEBEAM	specifies whether one beam is made for all facets or one for each
ONEFREQ	states that the current CC model was made with one frequency
OPCODE	General adverb, defines an operation
OPTELL	The operation to be passed to a task by TELL
OPTYPE	General adverb, defines a type of operation.
ORDER	Adverb used usually to specify the order of polynomial fit
OTFMODE	Specifies whether the data are in on-the-fly mode or not
OUT2CLAS	The class of a secondary output file
OUT2DISK	The disk number of a secondary output file.
OUT2NAME	The name of a secondary output file.
OUT2SEQ	The sequence of a secondary output file.
OUTCLASS	The class of an output file
OUTDISK	The disk number of an output file.
OUTFGVER	selects version of the flagging table to be written
OUTFILE	specifies name of output disk file, not in regular catalog
OUTNAME	The name of an output file.
OUTPRINT	Specifies name of disk file to keep the printer output
OUTSEQ	The sequence of an output file.
OUTTAPE	The output tape drive number.
OUTTEXT	specifies name of output text file, not in regular catalog
OUTVERS	The output version number of an extension file (table).
OVERLAP	specifies how overlaps are to be handled
OVRSWTCH	specifies when IMAGR switches from OVERLAP ¿= 2 to OVERLAP = 1 mode
PBPARM	Primary beam parameters
PBSIZE	estimates the primary beam size in interferometer images
PCUT	Cutoff in polarized intensity
PDVER	specifies the version of the spetral polarization table to use
PHASPRM	Phase data array, by antenna number.
PHAT	Prussian hat size
PHSLIMIT	gives a phase value in degrees
PIX2VAL	An image value in the units specified in the header.
PIX2XY	Specifies a pixel in an image
PIXAVG	Average image value
PIXRANGE	Range of pixel values to display
PIXSTD	RMS pixel deviation
PIXVAL	Value of a pixel
PIXXY	Specifies a pixel location.
PLCOLORS	specifies the colors to be used
PLEV	Percentage of peak to use for contour levels
PLVER	specifies the version number of a PL extension file
PMODEL	Polarization model parameters
POL3COL	Controls use of full 3-color graphics for polarization lines
POLANGLE	Intrinsic polarization angles for up to 30 sources
POLPLOT	specifies the desired polarization ratio before plotting.
PPARM	used by PEELR to list the facets to be ”peeled”
PRIORITY	Limits prioroty of messages printed
PRNUMBER	POPS number of messages
PROGRAM	Specify a program to be used
PRSTART	First record number in a print operation
PRTASK	Task name selected for printed information
PRTIME	Time limit
PRTLEV	Specifies a code for the amount of information requested.
PRTLIMIT	specifies limits to printing functions
PVPLOT	selects a display of a polarization vector in a box in a corner
QCREATE	adverb controlling the way large files are created
QUAL	Source qualifier
QUANTIZE	Quantization level to use
RADIUS	Specify a radius in an image
RASHIFT	Shift in RA
REASON	The reason for an operation
REFANT	Reference antenna number

REFDATE	To specify the initial or reference date of a data set
REFREQ	Specifies the frequency to be uased as the reference
REMHOST	gives the name of another computer which will provide service
REMQUE	specifies the desired batch queue on a remote computer
REMTAPE	gives the number of another computer's tape device
RESTFREQ	Rest frequency of a transition
REWEIGHT	Reweighting factors for UV data weights.
RGBCOLOR	specifies the desired TV graphics color
RGBGAMMA	specifies the desired color gamma corrections
RGBLEVS	colors to be applied to the contour levels
RMSLIMIT	selects things with RMS above this limit
ROBUST	Uniform weighting "robustness" parameter
ROMODE	Specified roam mode
ROTATE	Specifies a rotation
ROTMEAS	Tells the rotation measure of a source
RPARM	General numeric array adverb used in modeling
SAMPTYPE	Specifies sampling type
SCALR1	General adverb
SCALR2	General adverb
SCALR3	General adverb
SCANLENG	specify length of "scan"
SCUTOFF	noise level cutoff
SEARCH	Ordered list of antennas for fring searches
SELBAND	SpecifieS bandwidth of the desired frequency ID
SELFREQ	Specifies frequency of desired frequency ID
SHIFT	specifies a position shift
SKEW	Specifies a skew angle
SLOT	Specifies AIPS catalog slot number
SMODEL	Source model
SMOOTH	Specifies spectral smoothing
SMOTYPE	Specifies smoothing
SNCORPRM	Task-specific parameters for SNCOR.
SNVER	specifies the output solution table
SOLCON	Gain solution constraint factor
SOLINT	Solution interval
SOLMIN	Minimum number of solution sub-intervals in a solution
SOLMODE	Solution mode
SOLSUB	Solution sub-interval
SOLTYPE	Solution type
SORT	Specified desired sort order
SOUCODE	Calibrator code for source, not calibrator, selection
SOURCES	A list of source names
SPARM	General string array adverb
SPECFLUX	Specifies the flux at the reference frequency
SPECINDX	Spectral index used to correct calibrations
SPECPARM	Spectral index per polarization per source
SPECTRAL	Flag to indicate whether an operation is spectral or continuum
SPECURVE	Spectral index survature used to correct calibrations
SRCALIAS	A list of source names, used as an alias
SRCNAME	A single source name
STATIONS	Station names to be included
STFACTOR	scales star display or SDI CLEANing process
STOKES	Stokes parameter
STORE	Store current POPS environment
STRA1	General string adverb
STRA2	General string adverb
STRA3	General string adverb
STRB1	General string adverb
STRB2	General string adverb
STRB3	General string adverb
STRC1	General string adverb
STRC2	General string adverb
STRC3	General string adverb

STVERS	star display table version number
SUBARRAY	Subarray number
SYMBOL	General adverb, probably defines a plotting symbol type
SYS2COM	specifies a command to be sent to the operating system
SYSCOM	specifies a command to be sent to the operating system
SYSOUT	specifies the output device or file used by the system
SYSVEL	Systemic velocity
TASK	Name of a task
TAUO	Opacities by antenna number
TBLC	Gives the bottom left corner of an image to be displayed
TCODE	Deterimins which type of Y2K is RUN.
TDISK	Deterimins where output Y2K data is placed
TERSE	Controls the amount of information printed by the Y2K tests
THEDATE	contains the date and time in a string form
TIME2	used by VLBAMPCL as the time range for a second calibrator scan
TIMERANG	Specifies a time range
TIMETOL	used by MERGECAL to control accuracy when comparing times
TIMSMO	Specifiessmoothing times
TMASK	Deterimins which tasks are executed when a Y2K is RUN.
TMODE	Deterimins which input is used when a Y2K is RUN.
TNAMF	Internal variable for Y2K test suite
TRANSCOD	Specified desired transposition of an image
TRC	Specified the top right corner of a subimage
TRECVR	Receiver temperatures by polarization and antenna
TRIANGLE	specifies closure triangles to be selected/deselected
TT	spare scalar adverb for use in procedures
TTRC	Specifies the top right corner of a subimage to be displayed
TV2CHAN	Specified a second TV channel (plane) used in some operations
TVBUT	Tells which AIPS TV button was pushed
TVCHAN	Specified a TV channel (plane)
TVCORN	Specified the TV pixel for the bottom left corner of an image
TVLEVS	Gives the peak intensity to be displayed in levels
TVXY	Pixel position on the TV screen
TXINC	TV X coordinate increment
TYINC	TV Y coordinate increment
TYVER	specifies the version of the system temperature table used
TZINC	TV Z coordinate increment
USERID	User number
UVBOX	radius of the smoothing box used for uniform weighting
UVBXFN	type of function used when counting for uniform weighting
UVCOPPRM	Parameter adverb array for task UVCOP
UVFIXPRM	Parameter adverb array for task UVFIX
UVRANGE	Specify range of projected baselines
UVSIZE	specifies number of pixels on X and Y axes of a UV image
UVTAPER	Widths in U and V of gaussian weighting taper function
UVWTFN	Specify weighting function, Uniform or Natural
VCODE	General string adverb
VECTOR	selects method of averaging UV data
VELDEF	Specifies velocity definition
VELTYP	Velocity frame of reference
VERSION	Specify AIPS version or local task area
VLAMODE	VLA observing mode
VLAOBS	Observing program or part of observer's name
VLBINPRM	Control parameters to read data from NRAO/MPI MkII correlators
VNUMBER	Specifies the task parameter (VGET/VPUT) save area
VPARM	General numeric array adverb used in modeling
WEIGHTIT	Controls modification of weights before gain/fringe solutions
WGAUSS	Widths of Gaussian models (FWHM)
WTTHRESH	defines the weight threshold for data acceptance
WTUV	Specifies the weight to use for UV data outside UVRANGE
X	spare scalar adverb for use in procedures
XAXIS	Which parameter is plotted on the horizontal axis.
XINC	increment associated with an array of numbers

XYRATIO	Ratio of X to Y units per pixel
Y	spare scalar adverb for use in procedures
Y2KDISK	Specifies logical name of disk area for Y2K to read or write
Y2KFILE	File name of the current Y2K disk file
Y2KSIZE	Specifies the size of the Y2K test to be run
YINC	Y axis increment
YPARM	Specifies Y axis convolving function
YTYPE	Y axis (V) convolving function type
ZEROSP	Specify how to include zero spacing fluxes in FT of UV data
ZINC	Set the increment of the third axis
ZXRATIO	Ratio between Z axis (pixel value) and X axis

13.2 ANALYSIS

ACIMG	Makes image of autocorrelation data showing time vs frequency
ACTNOISE	puts estimate of actual image uncertainty and zero in header
AFARS	Is used after FARS to determine Position and Value of the maximum
AGAUS	Fits 1-dimensional Gaussians to absorption-line spectra
AHIST	Task to convert image intensities by adaptive histogram
AVOPTION	Controls type or range of averaging done by a task
BDEPO	computes depolarization due to rotation measure gradients
BLANK	blanks out selected, e.g. non-signal, portions of an image
BLSUM	sums images over irregular sub-images, displays spectra
BSCOR	Combines two beam-switched images
BSTST	Graphical display of solutions to frequency-switched data
BWSMEAR	amount of bandwidth smearing correction to use
CC2IM	Make model image from a CC file
CMPLX	Combines complex images by a variety of mathematical methods.
COMB	combines two images by a variety of mathematical methods
CTYPE	specifies type of component
CUBIT	Model a galaxy's density and velocity distribution from full cube
DFTIM	Makes image of DFT at arbitrary point showing time vs frequency
DOALIGN	specifies how two or more images are aligned in computations
DOFARS	Procedure to aid in Faraday rotation synthesis using the FARS task
DOINVERS	selects opposite of normal function
DOMAX	selects solutions for maxima of models
DOOUTPUT	selects whether output image or whatever is saved / discarded
DOPOS	selects solutions for positions of model components
DOSPIX	selects solutions for spectral index of model components
DOWIDTH	selects solution for widths of model components
ECOUNT	give the highest count or iteration for some process
FARS	Faraday rotation synthesis based on the brightness vs wavelength
FLUX	gives a total intensity value for image/component or to limit
FMAX	specifies peak values of model components - results of fits
FPOS	specifies pixel positions of fit model components
FQUBE	collects n-dimensional images into n+1-dimensional FREQID image
FSHIFT	specifies a position shift - output from fitting routines
FTARS	Fits 1-dimensional polarization spectrum to Q/U text file
FWIDTH	gives widths of model components - results of fitting
GAL	Determine parameters from a velocity field
GMAX	specifies peak values of model components
GPOS	specifies pixel positions of model components
GRBLINK	Verb which blinks 2 TV graphics planes
GWIDTH	gives widths of model components
HGEOM	interpolates image to different gridding and/or geometry
HLPTVHLD	Interactive image display with histogram equalization - run-time help

HLPTVIEW	Interactive display of cube in 3 transpositions - run-time help
HLPTVSAD	Find & fit Gaussians to an image with interaction - run-time help
HLPTVSPC	Interactive display of spectra from a cube - run-time help
HOCLN	Task to do a clean of holography images
HOLGR	Read & process holography visibility data to telescope images
HOLOG	Read & process holography visibility data to telescope images
IM2TX	Sums an image region into a 1-dimensional text file
IMCENTER	returns pixel position of sub-image centroid
IMDIST	determines spherical distance between two pixels
IMEAN	displays the mean & extrema and plots histogram of an image
IMERG	merges images of different spatial resolutions
IMFIT	Fits Gaussians to portions of an image
IMLIN	Fits and removes continuum emission from cube
IMMOD	adds images of model objects to an image
IMSTAT	Returns statistics of a sub-image
IMVAL	returns image intensity and coordinate at specified pixel
IMVIM	plots one image's values against another's
IRING	integrates intensity / flux in rings / ellipses
JMFIT	Fits Gaussians to portions of an image
LAYER	Task to create an RGB image from multiple images
LGEOM	regrids images with rotation, shift using interpolation
LISPX	Fits spectral indexes to a list of spectra, allows interactive edit
MATHS	operates on an image with a choice of mathematical functions
MAXFIT	returns pixel position and image intensity at a maximum
MCUBE	collects n-dimensional images into n+1-dimensional image
MEDI	combines four images by a variety of mathematical methods
MFITSET	gets adverbs for running IMFIT and JMFIT
MFPRT	prints MF tables in a format needed by modelling software
MINPATCH	specifies the minimum size allowed for the center of the beam
MODAB	Makes simple absorption/emission spectral-line image in I/V
MODIM	adds images of model objects to image cubes in IQU polarization
MODSP	adds images of model objects to image cubes in I/V polarization
MOMFT	calculates images of moments of a sub-image
MOMNT	calculates images of moments along x-axis (vel, freq, ch)
MTARS	Compute model inputs to TARS
MWFLT	applies linear & non-linear filters to images
NGAUSS	Number of Gaussians to fit
NINER	Applies various 3x3 area operaters to an image.
NNLSQ	Non-Negative-Least-Squares decomposition of spectrum
OMFIT	Fits sources and, optionally, a self-cal model to uv data
OUTTEXT	specifies name of output text file, not in regular catalog
PANEL	Convert HOLOG output to panel adjustment table
PBCOR	Task to apply or correct an image for a primary beam
PRTIM	prints image intensities from an MA catalog entry
QIMVAL	returns image intensity and coordinate at specified pixel
QUXTR	extracts text files from Q,U cubes for input to TARS
REFREQ	Specifies the frequency to be uased as the reference
RFARS	Correct Q/U cubes for Faraday rotation synthesis results
RM	Task to calculate rotation measure and magnetic field
RM2PL	Plots spectrum of a pixel with RMFIT fit
RMFIT	Fits 1-dimensional polarization spectrum to Q/U cube
RMSD	Calculate rms for each pixel using data in a box around the pixel
RMSLIMIT	selects things with RMS above this limit
RMTFC	Convolves rotation measure transfer function with slab models.
SAD	Finds and fits Gaussians to portions of an image
SCLIM	operates on an image with a choice of mathematical functions
SERCH	Finds line signals in transposed data cube
SHADO	Calculate the shadowing of antennas at the array
SLCOL	Task to collate slice data and models.
SLFIT	Task to fit gaussians to slice data.
SLICE	Task to make a slice file from an image
SMOTH	Task to smooth a subimage from upto a 7-dim. image

SPCOR	Task to correct an image for a primary beam and spectral index
SPECFLUX	Specifies the flux at the reference frequency
SPFIX	Makes cube from input to and output from SPIXR spectral index
SPIXR	Fits spectral indexes to each row of an image incl curvature
SPMOD	Modify UV database by adding a model with spectral lines
SPXMD	adds images of model objects to image cubes in spectral index
STFUN	Task to calculate a structure function image
STVERS	star display table version number
SUMSQ	Task to sum the squared pixel values of overlapping images
TABGET	returns table entry for specified row, column and subscript.
TABPUT	replaces table entry for specified row, column and subscript.
TARPL	Plot input and output text files of the TARS task
TARS	Simulation of Faraday rotation synthesis (mainly task FARS)
TK1SET	Verb to reset 1D gaussian fitting initial guess.
TKAGUESS	Verb to add a plot of the slice model guess directly on TEK
TKAMODEL	Verb to add slice model display directly on TEK
TKASLICE	Verb to add a slice display on TEK from slice file
TKGUESS	Verb to display slice model guess directly on TEK
TKMODEL	Verb to display slice model directly on TEK
TKSET	Verb to set 1D gaussian fitting initial guesses.
TKSLICE	Verb to display slice file directly on TEK
TKVAL	Verb to obtain value under cursor from a slice
TKXY	Verb to obtain pixel value under cursor
TV1SET	Verb to reset 1D gaussian fitting initial guess on TV plot.
TVACOMPS	Verb to add slice model components directly on TV graphics
TVAGUESS	Verb to re-plot slice model guess directly on TV graphics
TVAMODEL	Verb to add slice model display directly on TV graphics
TVARESID	Verb to add slice model residuals directly on TV graphics
TVASLICE	Verb to add a slice display on TV graphics from slice file
TVBLINK	Verb which blinks 2 TV planes, can do enhancement also
TVCOMPS	Verb to display slice model components directly on TV graphics
TVCUBE	Verb to load a cube into tv channel(s) & run a movie
TVDIST	determines spherical distance between two pixels on TV screen
TVGUESS	Verb to display slice model guess directly on TV graphics
TVHLD	Task to load an image to the TV with histogram equalization
TVIEW	Display image cube in all 3 transpositions with rotation & reference
TVMAXFIT	displays fit pixel positions and intensity at maxima on TV
TVMODEL	Verb to display slice model directly on TV graphics
TVRESID	Verb to display slice model residuals directly on TV graphics
TVSAD	Finds and fits Gaussians to portions of an image with interaction
TVSET	Verb to set slice Gaussian fitting initial guesses from TV plot
TVSLICE	Verb to display slice file directly on TV
TVSPC	Display images and spectra from a cube including fitting of spectra
TVSPX	Fits spectral indexes to each image row, allows interactive edit
TX2IM	Converts a 1-dimensional text file into cataloged image
UNSPX	Removes spectral index model from a data cube
UVADC	Fourier transforms and corrects a model and adds to uv data.
UVCON	Generate sample UV coverage given a user defined array layout
UVFIT	Fits source models to uv data.
UVGIT	Fits source models to uv data.
UVHIM	Makes image of the histogram on two user-chosen axes
UVMOD	Modify UV database by adding a model incl spectral index
UVSEN	Determine RMS sidelobe level and brightness sensitivity
UVSIM	Generate sample UV coverage given a user defined array layout
WARP	Model warps in Galaxies
XBASL	Fits and subtracts nth-order baselines from cube (x axis)
XG2PL	Plots spectrum of a pixel with XGAUS/AGAUS and ZEMAN/ZAMAN fits
XG2XG	Changes number of components in an XG or ZE table creating new one
XGAUS	Fits 1-dimensional Gaussians to images: restartable
XMOM	Fits one-dimensional moments to each row of an image
ZAMAN	Fits 1-dimensional Zeeman model to absorption-line data
ZEMAN	Fits 1-dimensional Zeeman model to data

13.3 AP

APCLN	Deconvolves images with CLEAN algorithm
APGS	deconvolves image with Gerchberg-Saxton algorithm
APVC	Deconvolves images with van Cittert algorithm
BLING	find residual rate and delay on individual baselines
BPASS	computes spectral bandpass correction table
BSGRD	Task to image beam-switched single-dish data
CALIB	determines antenna calibration: complex gain
CCRES	Removes or restores a CC file to a map with a gaussian beam.
CONPL	Plots AIPS gridding convolution functions
CONVL	convolves an image with a gaussian or another image
CPASS	computes polynomial spectral bandpass correction table
EDITA	Interactive TV task to edit uv data based on TY/SY/SN/CL tables
EDITR	Interactive baseline-oriented visibility editor using the TV
FFT	Takes Fourier Transform of an image or images
FRCAL	Faraday rotation self calibration task
FRING	fringe fit data to determine antenna calibration, delay, rate
GUARD	portion of UV plane to receive no data in gridding
HLPCLEAN	Cleaning tasks - run-time help
HLPSCIMG	Full-featured image plus self-cal loops, editing - run-time help
HLPSCMAP	Imaging plus self-cal and editing SCMAP - run-time help
IM2UV	converts an image to a visibility data set
IMAGR	Wide-field and/or wide-frequency Cleaning / imaging task.
IMFRING	large image delay fitting with IM2CC and OOFRING
IMSCAL	large image self-cal with IM2CC and OOCAL
KRING	fringe fit data to determine antenna calibration, delay, rate
MAXPIXEL	maximum pixels searched for components in Clark CLEAN
NOBAT	Task to lock lower priority users out of the AP
OOCAL	determines antenna complex gain with frequency-dependent models
OOFRING	Fringe fit data to determine calibration with spectral index options
RLDLY	fringe fit data to determine antenna R-L delay difference
RSTOR	Restores a Clean component file to a map with a Gaussian beam.
SCIMG	Full-featured imaging plus self-calibration loop with editing
SCMAP	Imaging plus self-calibration loop with editing
SDGRD	Task to select and image random-position single-dish data
SDIMG	Task to select and image random-position single-dish data
SNEDT	Interactive SN/CL/TY/SY table editor using the TV
TDEPEND	Time-dependent imaging procedure sequence
TDSCANS	Time-dependent imaging procedure sequence: find intervals
TDSSCAN	Time-dependent imaging procedure sequence: find intervals
TDSTEP3	Time-dependent imaging procedure ”step 3”
TDSTEP5	Time-dependent imaging procedure sequence: later steps
UVADC	Fourier transforms and corrects a model and adds to uv data.
UVMAP	makes images from calibrated UV data.
VLBABPSS	computes spectral bandpass correction table

13.4 ASTROMET

ASTROMET	Describes the process of astrometric/geodetic reduction in AIPS
FRMAP	Task to build a map using fringe rate spectra
HF2SV	convert HF tables from FRING/MBDLY to form used by Calc/Solve
HFPRT	write HF tables from CL2HF
XTRAN	Create an image with transformed coordinates

13.5 BATCH

```
Type:  Operations to prepare, submit, and monitor batch jobs
Use:   There are two batch streams of AIPS, each capable of
   processing a queue of jobs.  To run a batch job, one must
   first prepare the text of the job in a work file.  This text
   may contain any normal AIPS/POPS statement including RUN,
   except for verbs and tasks related to batch preparation, the
   TV, the TEK4012 green screen, and the tape drives.  When the
   text is ready, it may be submitted to the batch AIPS.  On
   the way, it is tested for errors and is submitted only if
   none are found.  After successful submission, the work file
   and any RUN files involved may be altered without affecting
   the job.  Array processor tasks are allowed only in queue
   #2 and only at night.  They may be submitted at any time,
   however.  Line printer output should be directed to a user
   chosen file (via adverb OUTPRINT).  If OUTPRINT = ' ', all
   tasks and AIPS itself will write to a file named
   PRTFIL:BATCHjjj.nnn, where jjj is the job number in hex and
   nnn is the user number in hex.  Note that all print jobs
   are concatenated into the specified file(s).
Adverbs:
   BATQUE     Number of queue to be used ( 1 or 2 or more )
   JOBNUM     Job number involved (101 - 164, 201 -264, ...)
   BATFLINE   First line number to be editted or listed
   BATNLINE   Number of lines to be listed
Verbs:
   BATCH      Add text to BATQUE work file
   BATCLEAR   Initiate and clear BATQUE work file
   BATLIST    List BATNLINE starting with BATFLINE from BATQUE
              work file
   BATEDIT    Edit text in BATQUE work file starting with line
              BATFLINE (or immediate argument)
   BAMODIFY   Edit text in BATQUE work file in line BATFLINE (or
              immediate argument), character-mode editing.
   SUBMIT     Submit text in BATQUE work file as job for queue
              BATQUE
   JOBLIST    List BATNLINE starting with BATFLINE from text file
              of job JOBNUM
   QUEUES     List jobs submitted, running, and completed in
              queue BATQUE
   UNQUE      Remove JOBNUM from queue, copy text of job to work
              file BATQUE

Batch jobs may also be prepared and submitted outside of AIPS,
using the program BATER.  See HELP BATER.
************************************************************
```

AIPSB	AIPS main program for executing batch jobs
AIPSC	AIPS main program for testing and queuing batch jobs
BAMODIFY	edits characters in a line of a batch work file
BATCH	starts entry of commands into batch-job work file
BATCLEAR	removes all text from a batch work file
BATEDIT	starts an edit (replace, insert) session on a batch work file
BATER	stand-alone program to prepare and submit batch jobs
BATFLINE	specifies starting line in a batch work file
BATLIST	lists the contents of a batch work file
BATNLINE	specifies the number of lines to process in a batch work file
BATQUE	specifies the desired batch queue

ENDBATCH	terminates input to batch work file
JOBLIST	lists contents of a submitted and pending batch job
JOBNUM	specifies the batch job number
QUEUES	Verb to list all submitted jobs in the job queue
REMQUE	specifies the desired batch queue on a remote computer
UNQUE	remove a given job from the job queue

13.6 CALIBRAT

```
For a lengthy description of the calibration of interferometric
data (VLA and VLB line and continuum) enter:
             HELP CALIBRAT\\
************************************************************
```

ACCOR	Corrects cross amplitudes using auto correlation measurements
ACFIT	Determine antenna gains from autocorrelations
ACLIP	edits suto-corr data for amplitudes, phases, and weights out of range
ACSCL	Corrects cross amplitudes using auto correlation measurements
ALBUS	Calculate ionospheric delay and Faraday rotation using ALBUS
ANCAL	Places antenna-based Tsys and gain corrections in CL table
ANCHECK	Checks By sign in Antenna files
ANTAB	Read amplitude calibration information into AIPS
ANTENNAS	Antennas to include/exclude from the task or verb
ANTUSE	Antennas to include/exclude from the task or verb
ANTWT	Antenna Weights for UV data correction in Calibration
APCAL	Apply TY and GC tables to generate an SN table
APGPS	Apply GPS-derived ionospheric corrections
ATMCA	Determines delay/phase gradient from calibrator observations
AVERSION	Specify a version of the ALBUS container
BASELINE	specifies which antenna pairs are to be selected/deselected
BASFIT	fits antenna locations from SN-table data
BDAPL	Applies a BD table to another data set
BLAPP	applies baseline-based fringe solutions a la BLAPP
BLAVG	Average cross-polarized UV data over baselines.
BLCAL	Compute closure offset corrections
BLCHN	Compute closure offset corrections on a channel-by-channel basis
BLING	find residual rate and delay on individual baselines
BLPCL	Do divide by I,Q,U models for circular polarization BLCHN
BLPLT	Plots BLCAL tables in 2 dimensions as function of baseline and IF
BLPOLCAL	Do BLCHN for I, Q, U, V models
BLVER	specifies the version of the baseline-calibration table used
BPASS	computes spectral bandpass correction table
BPASS5	Normalization control in BPASS for VLBARUN
BPASSPRM	Control adverb array for bandpass calibration
BPCOR	Correct BP table.
BPEDT	Interactive TV task to edit uv data based on BP tables
BPEPL	Plots bandpass tables overlaying multiple times like BPEDT
BPERR	Print and plot BPASS closure outputs
BPFLG	Flgas uv data based on bad BP values
BPLOT	Plots bandpass tables in 2 dimensions as function of time
BPPLT	Plots selected contents of bandpass (BP) files versus time
BPSMO	Smooths or interpolates bandpass tables to regular times
BPSOLINT	Sets SOLINT in BPASS for VLBARUN
BPVER	specifies the version of the bandpass table to be applied
BPWAY	Determines channel-dependent relative weights
BPWGT	Calibrates data and scales weights by bandpass correction
BSPRT	print BS tables
BSROT	modifies SD beam-switch continuum data for error in throw

CALCODE	specifies the type of calibrator to be selected
CALDIR	lists calibrator source models available as AIPS FITS files
CALIB	determines antenna calibration: complex gain
CALIBRAT	describes the process of data calibration in AIPS
CALIN	specifies name of input disk file usually with calibration data
CALMODEL	Applies SPLIT & UVSUB to make a model data set matching a calibrator
CALRD	Reads calibrator source model-image FITS file
CALSOUR	specifies source names to be included in calibration
CALTASK	Controls which calibration task is used when by VLBARUN
CC2IM	Make model image from a CC file
CENTR	modifies UV data to center the reference channel
CHANSEL	Array of start, stop, increment channel numbers to average
CHREFANT	Name of reference antenna
CLCAL	merges and smooths SN tables, applies them to CL tables
CLCOP	copy CL/SN file calibration between polarizations or IFs
CLCOR	applies user-selected corrections to the calibration CL table
CLCORPRM	Parameter adverb array for task CLCOR
CLINT	CL table entry interval
CLINV	copy CL/SN file inverting the calibration
CLIP	edits data based on amplitudes, phases, and weights out of range
CLSMO	smooths a calibration CL table
CLVLB	Corrects CL table gains for pointing offsets in VLBI data
CMETHOD	specifies the method by which the uv model is computed
CMODEL	specifies the method by which the uv model is computed
CODETYPE	specifies the desired operation or data type
CONFI	Optimize array configuration by minimum side lobes
CPASS	computes polynomial spectral bandpass correction table
CSCOR	applies specified corrections to CS tables
CVEL	shifts spectral-line UV data to a given velocity
DCHANSEL	Array of start, stop, increment channel #S + IF to avoid
DECOR	Measures the decorrelation between channels and IF of uv data
DEFLG	edits data based on decorrelation over channels and time
DELCORR	Controls task actions on calibration tables.
DELZN	Determines residual atmosphere depth at zenith and clock errors
DFCOR	Applies ATMO differential corrections to the calibration CL table
DGAUS	Correct UV model for convolution by Clean beam
DIGICOR	specifies whether VLBA digital corrections are to be applied
DOACOR	specifies whether autocorrelation data are included
DOAPPLY	Flag to indicate whether an operation is applied to the data
DOBAND	specifies if/how bandpass calibration is applied
DOBTWEEN	Controls smoothing between sources in calibration tables
DOCALIB	specifies whether a gain table is to be applied or not
DODELAY	selects solution for phase/amplitude or delay rate/phase
DOFIT	Controls which antennas are fit by what methods
DOFLAG	Controls closure cutoff in gain solutions and flagging
DOOSRO	calibrating amplitude and phase, and imaging VLA data
DOPOL	selects application of any polarization calibration
DOVLAMP	Produces amp calibration file for phased-VLA VLBI data
DT2PD	Write PD table from a PD table text file
DTSIM	Generate fake UV data
EDITA	Interactive TV task to edit uv data based on TY/SY/SN/CL tables
EDITR	Interactive baseline-oriented visibility editor using the TV
ELEVLIM	Lower limit on elevation
ELFIT	Plots/fits selected contents of SN, TY, SY, PC or CL files
ELINT	Determines and removes gain dependence on elevation
EOPSFILE	Name of text file containing EOP information
EVACL	Evaluates statistics in closure phase and amplitude
EVASN	Evaluates statistics in SN/CL tables
EVAUV	Subtracts & divides a model into UV data, does statistics on results
FACES	makes images of catalog sources for initial calibration
FARAD	add ionospheric Faraday rotation to CL table
FGPLT	Plots selected contents of FG table

FGPRT	Prints counts of selected portions of an FG table.
FGTAB	writes frequency-range flags to a text file to be read by UVFLG
FINDR	Find normal values for a uv data set
FIXWT	Modify weights to reflect amplitude scatter of data
FLAGR	Edit data based on internal RMS, amplitudes, weights
FLAGVER	selects version of the flagging table to be applied
FLGIT	flags data based on the rms of the spectrum
FQTOL	Frequency tolerance with which FQ or frequency entries are accepted.
FRCAL	Faraday rotation self calibration task
FREQID	Frequency Identifier for frequency, bandwidth combination
FRING	fringe fit data to determine antenna calibration, delay, rate
FTFLG	interactive flagging of UV data in channel-time using the TV
GAINERR	gives estimate of gain uncertainty for each antenna
GAINUSE	specifies output gain table or gain table applied to data
GAINVER	specifies the input gain table
GCPLT	Plots gain curves from text files
GCVER	specifies the version of the gain curve table used
GETJY	determines calibrator flux densities
GPSDL	Calculate ionospheric delay and Faraday rotation corrections
HLPCLEAN	Cleaning tasks - run-time help
HLPEDIBP	Interactive BP table uv-data editor BPEDT - run-time help
HLPEDICL	Interactive SN/CL table uv-data editor - run-time help
HLPEDIPC	Interactive PC table editor PCEDT - run-time help
HLPEDIPD	Interactive PD table editor PDEDT - run-time help
HLPEDISN	Interactive SN/CL table (not UV) editor - run-time help
HLPEDISS	Interactive SY table (not UV) editor - run-time help
HLPEDISY	Interactive SY table uv-data editor EDITA - run-time help
HLPEDITS	Interactive TY table (not UV) editor - run-time help
HLPEDITY	Interactive TY table uv-data editor EDITA - run-time help
HLPEDIUV	Interactive uv-data editor EDITR - run-time help
HLPFTFLG	Interactive time-channel visibility Editor - run-time help
HLPIBLED	Interactive Baseline based visibility Editor - run-time help
HLPPCFLG	Interactive time-channel PC table Editor PCFLG - run-time help
HLPSCIMG	Full-featured image plus self-cal loops, editing - run-time help
HLPSCMAP	Imaging plus self-cal and editing SCMAP - run-time help
HLPSPFLG	Interactive time-channel visibility Editor SPFLG - run-time help
HLPTVFLG	Interactive time-baseline visibility Editor TVFLG - run-time help
HOLIS	Prints information about holography sampling
IANTB	Writes out GC and TY data in ANTAB format
IBLED	Interactive BaseLine based visibility EDitor
ICHANSEL	Array of start, stop, increment channel #S + IF to average
IM2CC	Task to convert an image to multi-facet Clean Components
IMAGR	Wide-field and/or wide-frequency Cleaning / imaging task.
IMFRING	large image delay fitting with IM2CC and OOFRING
IMSCAL	large image self-cal with IM2CC and OOCAL
INDXR	writes index file describing contents of UV data base
INTERPOL	specifies the type of averaging done on the complex gains
INTPARM	specifies the parameters of the frequency interpolation function
KRING	fringe fit data to determine antenna calibration, delay, rate
LDGPS	load GPS data from an ASCII file
LISTR	prints contents of UV data sets and assoc. calibration tables
LISUN	prints contents of UV data sets with Sun angles
LOCIT	fits antenna locations from SN-table data
LPCAL	Determines instrumental polarization for UV data
MAPBM	Map VLA beam polarization
MBDLY	Fits multiband delays from IF phases, updates SN table
MFIMG	Image the Earth magnetic field model or TEC data used by TECOR
MINAMPER	specifies the minimum amplitude error prior to some action
MINANTEN	states minimum number of antennas for a solution
MINPHSER	specifies the minimum phase error prior to some action
MOD3D	Computes a 3D CC model from a set of facets

MSORT	Sort a UV dataset into a specified order
MULTI	Task to convert single-source to multi-source UV data
NORMALIZ	specifies the type of gain normaliztion if any
OMFIT	Fits sources and, optionally, a self-cal model to uv data
ONEFREQ	states that the current CC model was made with one frequency
OOCAL	determines antenna complex gain with frequency-dependent models
OOFRING	Fringe fit data to determine calibration with spectral index options
OOSRT	Sort a UV dataset into a specified order
OOSUB	Subtracts/divides a model from/into a uv data base
PBEAM	Fits the analytic function to the measured values of the beam
PCABL	Copies the Cable cal values from PC table 1 to PC table n
PCAL	Determines instrumental polarization for UV data
PCASS	Finds amplitude bandpass shape from pulse-cal table data
PCAVG	Averages pulse-cal (PC) tables over time
PCCOR	Corrects phases using PCAL tones data from PC table
PCDIV	Scales pulse-cal amplitudes by Tsys or Psum
PCEDT	Interactive TV task to edit pulse-cal (PC) tables
PCFIT	Finds delays and phases using a pulse-cal (PC) table
PCFLG	interactive flagging of Pulse-cal data in channel-TB using the TV
PCLOD	Reads ascii file containing pulse-cal info to PC table.
PCPLT	Plots pulse-cal tables in 2 dimensions as function of time
PCRMS	Finds statistics of a pulse-cal table; flags bad times and channels
PCVEL	shifts spectral-line UV data to a given velocity: planet version
PDEDT	Interactive TV task to edit polarization D-term (PD) tables
PDPLT	Plots selected contents of PD tables
PDVER	specifies the version of the spetral polarization table to use
PEEK	fits pointing model function to output from the VLA
PEELR	calibrates interfering sources in multi-facet imges
PHASPRM	Phase data array, by antenna number.
PIPEAIPS	calibrating amplitude and phase, and imaging VLA data
POLANGLE	Intrinsic polarization angles for up to 30 sources
PPAPP	Applies Right minus Left phase difference to a 2nd data set
PPARM	used by PEELR to list the facets to be "peeled"
PROGRAM	Specify a program to be used
PRTSY	Task to print statistics from the SY table
QBEAM	Plot holography data as contour image and outputs interpolated image
QUFIX	determines Right minus Left phase difference, corrects cal files
QUOUT	writes text file of Q, U versus frequency to be used by RLDIF
REAMP	Scale visibility amplitudes by IF- or subarray-dependent factor
REFANT	Reference antenna number
RESEQ	Renumber antennas
REWGT	modifies UV data re-scaling the weights only
RFI	Look for RFI in uv data
RIFRM	Task to modify a CL and TE table with IFRM values from ALBUS files
RLCAL	Determines instrumental right-left phase versus time (a self-cal)
RLDIF	determines Right minus Left phase difference, corrects cal files
RLDLY	fringe fit data to determine antenna R-L delay difference
ROTMEAS	Tells the rotation measure of a source
SAMPTYPE	Specifies sampling type
SCIMG	Full-featured imaging plus self-calibration loop with editing
SCMAP	Imaging plus self-calibration loop with editing
SDCAL	Task to apply single dish calibration
SDVEL	shifts spectral-line single-dish data to a given velocity
SEARCH	Ordered list of antennas for fring searches
SELBAND	SpecifieS bandwidth of the desired frequency ID
SELFCAL	Requests self-calibration of calibration sources
SELFREQ	Specifies frequency of desired frequency ID
SETJY	Task to enter source info into source (SU) table.
SHAHO	Flags Holography data for shadowing
SHOUV	displays uv data in various ways.
SMODEL	Source model

SMOTYPE	Specifies smoothing
SMRATES	Requests smoothing and clipping of rates after FRING in pipeline
SNBLP	Plots selected contents of SN or CL files on a baseline basis
SNCOR	applies user-selected corrections to the calibration SN table
SNCORPRM	Task-specific parameters for SNCOR.
SNDUP	copies and duplicates SN table from single pol file to dual pol
SNEDT	Interactive SN/CL/TY/SY table editor using the TV
SNFIT	Fits parabola to SN amplitudes and plots result
SNFLG	Writes flagging info based on the contents of SN files
SNIFS	Plots selected contents of SN, TY, SY, PC or CL files by IF
SNP2D	Task to convert SN table single-channel phase to delay
SNPLT	Plots selected contents of SN, SY, TY, PC or CL files versus time
SNREF	Chooses best reference antenna to minimize R-L differences
SNRMS	Print statistics for selected contents of SN, SY, TY, PC or CL files
SNSMO	smooths and filters a calibration SN table
SNVER	specifies the output solution table
SOLCL	adjust gains for solar data according to nominal sensitivity
SOLCON	Gain solution constraint factor
SOLINT	Solution interval
SOLMIN	Minimum number of solution sub-intervals in a solution
SOLMODE	Solution mode
SOLSUB	Solution sub-interval
SOLTYPE	Solution type
SOUCODE	Calibrator code for source, not calibrator, selection
SOUSP	fits source spectral index from SU table or adverbs
SPCAL	Determines instrumental polzn. for spec. line UV data
SPECINDX	Spectral index used to correct calibrations
SPECPARM	Spectral index per polarization per source
SPECTRAL	Flag to indicate whether an operation is spectral or continuum
SPECURVE	Spectral index survature used to correct calibrations
SPFLG	interactive flagging of UV data in channel-TB using the TV
SPLAT	Applies calibration and splits or assemble selected sources.
SPLIT	converts multi-source to single-source UV files w calibration
STATIONS	Station names to be included
SY2TY	Task to generate a TY extension file from an EVLA SY table
SYHIS	Task to clip, filter, smooth SY tables and plot histograms
SYPRT	Task to print values from the SY table
SYSOL	undoes and re-does nominal sensitivity application for Solar data
SYVSN	Plots selected SY/TY data with SN solution values versus time
TASAV	Task to copy all extension tables to a dummy uv or map file
TAUO	Opacities by antenna number
TDEPEND	Time-dependent imaging procedure sequence
TECOR	Calculate ionospheric delay and Faraday rotation corrections
TECRFILE	Name of file containg TEC data for use in TECOR
TECRTYPE	Type of IONEX file to download in VLA and VLBA TECR procedures
TEPLT	Plots selected contents of TE (TECOR) parameter tables
TIME2	used by VLBAMPCL as the time range for a second calibrator scan
TIMSMO	Specifiessmoothing times
TLCAL	Converts JVLA telcal files to an SN file
TRECVR	Receiver temperatures by polarization and antenna
TRIANGLE	specifies closure triangles to be selected/deselected
TRUEP	determines true antenna polarization from special data sets
TVFLG	interactive flagging of UV data using the TV
TYAPL	undoes and re-does nominal sensitivity application
TYCOP	copy TY or SY table calibration between IFs
TYSMO	smooths and filters a calibration TY or SY table
TYSMOLEV	Clip level to use in running TYSMO on a VLB TY table
TYVER	specifies the version of the system temperature table used
UNCAL	sets up tables for uncalibrating Australia Telescope data
USUBA	Assign subarrays within a uv-data file
UVCRS	Finds the crossing points of UV-ellipses.
UVFIT	Fits source models to uv data.

UVFLG	Flags UV-data
UVGIT	Fits source models to uv data.
UVHOL	Prints holography data from a UV data base with calibration
UVMLN	edits data based on the rms of the spectrum
UVPRT	prints data from a UV data base with calibration
UVSRT	Sort a UV dataset into a specified order
UVSUB	Subtracts/divides a model from/into a uv data base
VBCAL	Scale visibility amplitudes by antenna based constants
VHCAL	Do divide by I,Q,U models for linear polarization data
VHCALIB	Do CALIB for linear polarization with I, Q, U models
VHDIF	find/apply V minus H linear polarization phase difference
VHMOD	Compute linear polarization point source model, applies to data
VLABP	VLA antenna beam polarization correction for snapshot images
VLACALIB	Runs CALIB and LISTR for VLA observation
VLACLCAL	Runs CLCAL and prints the results with LISTR
VLALIST	Runs LISTR for VLA observation
VLAMODE	VLA observing mode
VLAMP	Makes ANTAB file for phased VLA used in VLBI observations
VLANT	applies VLA/EVLA antenna position corrections from OPs files
VLAOBS	Observing program or part of observer's name
VLAPROCS	Procedures to simplify the reduction of VLBA data
VLARESET	Reset calibration tables to a virginal state
VLARUN	Calibrating amplitude and phase, and imaging VLA data
VLBAAMP	applies a-priori amplitude corrections to VLBA data
VLBABPSS	computes spectral bandpass correction table
VLBACALA	applies a-priori amplitude corrections to VLBA data
VLBACCOR	applies a-priori amplitude corrections to VLBA data
VLBAEOPS	Corrects Earth orientation parameters
VLBAFQS	Copies different FQIDS to separate files
VLBAFRGP	Fringe fit phase referenced data and apply calibration
VLBAFRNG	Fringe fit data and apply calibration
VLBAKRGP	Fringe fit phase referenced data and apply calibration
VLBAKRNG	Fringe fit data and apply calibration
VLBAMCAL	Merges redundant calibration data
VLBAMPCL	Calculates and applies manual instrumental phase calibration
VLBAMPHC	applies VLBARUN calibration to additional phase stopping centers
VLBAPANG	Corrects for parallactic angle
VLBAPCOR	Calculates and applies instrumental phase calibration
VLBAPIPE	Applies amplitude and phase calibration procs to VLBA data
VLBARFI	Calibrates VLBA autocorrelations, writes out statistics
VLBARUN	applies amplitude and phase calibration procs to VLBA data
VLBATECR	Calculate ionospheric delay and Faraday rotation corrections
VLBAUTIL	Procedures to simplify the reduction of VLBA data
VLBAWIDE	Calibrates all VLA wide-band autocorrelations with VLBARFI
VLBCD	Reads ascii file containing VLBA calibration info into a CD table
VLBSY	Reads ascii file containing VLBA total power info into an SY table
VLOG	Pre-process external VLBA calibration files
WEIGHTIT	Controls modification of weights before gain/fringe solutions
WETHR	Plots selected contents of WX tables, flags data based on WX
WRTPROCS	Procedures to simplify the reduction of VLBA data
WTTHRESH	defines the weight threshold for data acceptance
WTUV	Specifies the weight to use for UV data outside UVRANGE
WXLOD	Reads KEYIN-format weather information making WX table
XYDIF	find/apply X minus Y linear polarization phase difference
XYMOD	Compute linear polarization point source model, applies to data

13.7 CATALOG

ACTNOISE	puts estimate of actual image uncertainty and zero in header
ADDBEAM	Inserts clean beam parameters in image header
ADDDISK	makes a computer's disks available to the current AIPS session
ALLDEST	Delete a group or all of a users data files
ALTDEF	Sets frequency vs velocity relationship into image header
ALTSWTCH	Switches between frequency and velocity in image header
ARESTORE	Restores back up tapes of users data
AX2REF	Second reference pixel number
AXDEFINE	Define or modify an image axis description
AXINC	Axis increment - change in coordinate between pixels
AXREF	Reference pixel number
AXTYPE	Type of coordinate axis
AXVAL	Value of axis coordinate at reference pixel
BAKLD	reads all files of a catalog entry from BAKTP tape
BAKTP	writes all files of a catalog entry to tape in host format
CAT2LOG	list one or more entries in the user's data directory: #2 file name
CAT3LOG	list one or more entries in the user's data directory: #3 file name
CAT4LOG	list one or more entries in the user's data directory: #4 file name
CAT5LOG	list one or more entries in the user's data directory: #5 file name
CATALOG	list one or more entries in the user's data directory
CATNO	Specifies AIPS catalog slot number range
CATOLOG	list one or more entries in the user's data directory: out file name
CELGAL	switches header between celestial and galactic coordinates
CHKNAME	Checks for existence of the specified image name
CLNUP	removes old MS, RL, SG, TG and ?? files from AIPS disks
CLR2NAME	clears adverbs specifying the second input image
CLR3NAME	clears adverbs specifying the third input image
CLR4NAME	clears adverbs specifying the fourth input image
CLR5NAME	clears adverbs specifying the fifth input image
CLRNAME	clears adverbs specifying the first input image
CLRONAME	clears adverbs specifying the first output image
CLRSTAT	remove any read or write status flags on a directory entry
COODEFIN	Define or modify an image axis coordinate description
COOINC	Celestial axes increment: change in coordinate between pixels
COOREF	Reference pixel number for two coordinate axes
COOTYPE	Celestial axes projection type
DAYNUMBR	finds day nuumber of an image or uv data set
DISKU	shows disk use by one or all users
DOALPHA	specifies whether some list is alphabetized
DOCAT	specifies whether the output is saved (cataloged) or not
DOCC	Adverb to control application to Clean component files
DOHIST	selects a histogram display or controls writing of history file
DOOUTPUT	selects whether output image or whatever is saved / discarded
EGETHEAD	returns parameter value from image header and error code
EGETNAME	fills in input name adverbs by catalog slot number, w error
EPOSWTCH	Switches between B1950 and J2000 coordinates in header
ERROR	was there an error
EXTDEST	deletes one or more extension files
EXTLIST	lists detailed information about contents of extension files
GET2NAME	fills 2nd input image name parameters by catalog slot number
GET3NAME	fills 3rd input image name parameters by catalog slot number
GET4NAME	fills 4th input image name parameters by catalog slot number
GET5NAME	fills 5th input image name parameters by catalog slot number
GETHEAD	returns parameter value from image header
GETNAME	fills 1st input image name parameters by catalog slot number
GETONAME	fills 1st output image name parameters by catalog slot number

GETVERS	finds maximum version number of an extension file
HGEOM	interpolates image to different gridding and/or geometry
HIEND	End record number in a history-file operation
HINOTE	adds user-generated lines to the history extension file
HISTART	Start record number in a history-file operation
HITEXT	writes lines from history extension file to text file
IM2HEAD	displays the image 2 header contents to terminal, message file
IM3HEAD	displays the image 3 header contents to terminal, message file
IM4HEAD	displays the image 4 header contents to terminal, message file
IM5HEAD	displays the image 5 header contents to terminal, message file
IMDIST	determines spherical distance between two pixels
IMHEADER	displays the image header contents to terminal, message file
IMOHEAD	displays the output image header contents
IMPOS	displays celestial coordinates selected by the TV cursor
IMVAL	returns image intensity and coordinate at specified pixel
IN2CLASS	specifies the ”class” of the 2nd input image or data base
IN2DISK	specifies the disk drive of the 2nd input image or data base
IN2EXT	specifies the type of the 2nd input extension file
IN2NAME	specifies the ”name” of the 2nd input image or data base
IN2SEQ	specifies the sequence # of the 2nd input image or data base
IN2TYPE	specifies the type of the 2nd input image or data base
IN2VERS	specifies the version number of the 2nd input extension file
IN3CLASS	specifies the ”class” of the 3rd input image or data base
IN3DISK	specifies the disk drive of the 3rd input image or data base
IN3EXT	specifies the type of the 3rd input extension file
IN3NAME	specifies the ”name” of the 3rd input image or data base
IN3SEQ	specifies the sequence # of the 3rd input image or data base
IN3TYPE	specifies the type of the 3rd input image or data base
IN3VERS	specifies the version number of the 3rd input extension file
IN4CLASS	specifies the ”class” of the 4th input image or data base
IN4DISK	specifies the disk drive of the 4th input image or data base
IN4NAME	specifies the ”name” of the 4th input image or data base
IN4SEQ	specifies the sequence # of the 4th input image or data base
IN4TYPE	specifies the type of the 4th input image or data base
IN4VERS	specifies the version number of the 4th input extension file
IN5CLASS	specifies the ”class” of the 5th input image or data base
IN5DISK	specifies the disk drive of the 5th input image or data base
IN5NAME	specifies the ”name” of the 5th input image or data base
IN5SEQ	specifies the sequence # of the 5th input image or data base
IN5TYPE	specifies the type of the 5th input image or data base
IN5VERS	specifies the version number of the 5th input extension file
INCLASS	specifies the ”class” of the 1st input image or data base
INDISK	specifies the disk drive of the 1st input image or data base
INEXT	specifies the type of the 1st input extension file
INNAME	specifies the ”name” of the 1st input image or data base
INSEQ	specifies the sequence # of the 1st input image or data base
INTYPE	specifies the type of the 1st input image or data base
INVERS	specifies the version number of the 1st input extension file
KEYSTRNG	gives contents of character-valued keyword parameter
KEYTYPE	Adverb giving the keyword data type code
KEYVALUE	gives contents of numeric-valued keyword parameter
KEYWORD	gives name of keyword parameter - i.e. name of header field
LGEOM	regrids images with rotation, shift using interpolation
M2CAT	displays images in the user’s catalog directory for IN2DISK
M3CAT	displays images in the user’s catalog directory for IN3DISK
M4CAT	displays images in the user’s catalog directory for IN4DISK
M5CAT	displays images in the user’s catalog directory for IN5DISK
MAXVERS	The maximum version number of a specified extension file type
MCAT	lists images in the user’s catalog directory on disk INDISK
MOCAT	displays images in the user’s catalog directory for OUTDISK

MOVE	Task to copy or move data from one user or disk to another
NAMEGET	fills 1st input image name parameters by default matching
OUT2CLAS	The class of a secondary output file
OUT2DISK	The disk number of a secondary output file.
OUT2NAME	The name of a secondary output file.
OUT2SEQ	The sequence of a secondary output file.
OUTCLASS	The class of an output file
OUTDISK	The disk number of an output file.
OUTNAME	The name of an output file.
OUTSEQ	The sequence of an output file.
OUTVERS	The output version number of an extension file (table).
PCAT	Verb to list entries in the user's catalog (no log file).
PLVER	specifies the version number of a PL extension file
PRTHI	prints selected contents of the history extension file
PUTHEAD	Verb to modify image header parameters.
Q2HEADER	Verb to summarize the image 2 header: positions at center
Q3HEADER	Verb to summarize the image 3 header: positions at center
Q4HEADER	Verb to summarize the image 4 header: positions at center
Q5HEADER	Verb to summarize the image 5 header: positions at center
QGETVERS	finds maximum version number of an extension file quietly
QHEADER	Verb to summarize the image header: positions at center
QOHEADER	Verb to summarize the output image header: center positions
QUAL	Source qualifier
RECAT	Verb to compress the entries in a catalog file
REMDISK	removes a computer's disks from the current AIPS session
RENAME	Rename a file (UV or Image)
RENUMBER	Verb to change the catalog number of an image.
RESCALE	Verb to modify image scale factor and offset
REVERSN	checks disk for presence of extension files
SCAT	lists scratch files in the user's catalog directory on all disks
SCRDEST	Verb to destroy scratch files left by bombed tasks.
SLOT	Specifies AIPS catalog slot number
STALIN	revises history by deleting lines from history extension file
STARTTV	Restarts XAS (TV), MS, TK servers that have disappered
TVDIST	determines spherical distance between two pixels on TV screen
TVRESTOR	Restarts XAS (TV), MS, TK servers that have disappered
U2CAT	list a user's UV and scratch files on disk IN2DISK
U3CAT	list a user's UV and scratch files on disk IN3DISK
U4CAT	list a user's UV and scratch files on disk IN4DISK
U5CAT	list a user's UV and scratch files on disk IN5DISK
UCAT	list a user's UV and scratch files on disk INDISK
UOCAT	list a user's UV and scratch files on disk OUTDISK
USERID	User number
ZAP	Delete a catalog entry and its extension files

13.8 COORDINA

ALTDEF	Sets frequency vs velocity relationship into image header
ALTSWTCH	Switches between frequency and velocity in image header
CODECIML	Convert between decimal and sexagesimal coordinate values
COODEFIN	Define or modify an image axis coordinate description
COORDINA	Array to hold coordinate values
COPIXEL	Convert between physical and pixel coordinate values
COSTAR	Verb to plot a symbol at given position on top of a TV image
COTVLOD	Proc to load an image into a TV channel about a coordinate
COWINDOW	Set a window based on coordinates
EPOCONV	Convert between J2000 and B1950 coordinates
EPOSWTCH	Switches between B1950 and J2000 coordinates in header

FRMAP	Task to build a map using fringe rate spectra
FSHIFT	specifies a position shift - output from fitting routines
NAXIS	Axis number
OBEDT	Task to flag data of orbiting antennas
OBTAB	Recalculate orbit parameters and other spacecraft info
PIX2VAL	An image value in the units specified in the header.
PIX2XY	Specifies a pixel in an image
RASHIFT	Shift in RA
REGRD	Regrids an image from one co-ordinate frame to another
RESTFREQ	Rest frequency of a transition
ROTATE	Specifies a rotation
SHIFT	specifies a position shift
SKYVE	Regrids a DSS image from one co-ordinate frame to another
SYSVEL	Systemic velocity
VELDEF	Specifies velocity definition
VELTYP	Velocity frame of reference
XINC	increment associated with an array of numbers
XPARM	General adverb for up to 10 parameters, may refer to X coord
XTRAN	Create an image with transformed coordinates
XTYPE	X axis (U) convolving function type
XYRATIO	Ratio of X to Y units per pixel
ZINC	Set the increment of the third axis
ZXRATIO	Ratio between Z axis (pixel value) and X axis

13.9 EDITING

ACLIP	edits suto-corr data for amplitudes, phases, and weights out of range
BPEDT	Interactive TV task to edit uv data based on BP tables
BPFLG	Flgas uv data based on bad BP values
CLIP	edits data based on amplitudes, phases, and weights out of range
CROWDED	allows a task to perform its function in a crowded fashion
DEFLG	edits data based on decorrelation over channels and time
DOCLIP	Controls whether and how a CLIP operation is performed
DOFLAG	Controls closure cutoff in gain solutions and flagging
DOPLOT	Controls plotting of something
DOROBUST	Controls method of averaging - simple mean/rms or robust
EDITA	Interactive TV task to edit uv data based on TY/SY/SN/CL tables
EDITR	Interactive baseline-oriented visibility editor using the TV
EXPERT	specifies an user experience level or mode
FGCNT	Counts samples comparing two flag tables or detailed count from ¡= 1
FGDIF	Compares affect of 2 FG tables
FGSPW	Flags bad spectral windows
FGTAB	writes frequency-range flags to a text file to be read by UVFLG
FINDR	Find normal values for a uv data set
FLAGR	Edit data based on internal RMS, amplitudes, weights
FLGIT	flags data based on the rms of the spectrum
FTFLG	interactive flagging of UV data in channel-time using the TV
GETITIME	Verb to return the UV data integration time.
HLPEDIBP	Interactive BP table uv-data editor BPEDT - run-time help
HLPEDICL	Interactive SN/CL table uv-data editor - run-time help
HLPEDIPC	Interactive PC table editor PCEDT - run-time help
HLPEDIPD	Interactive PD table editor PDEDT - run-time help
HLPEDISN	Interactive SN/CL table (not UV) editor - run-time help
HLPEDISS	Interactive SY table (not UV) editor - run-time help
HLPEDISY	Interactive SY table uv-data editor EDITA - run-time help
HLPEDITS	Interactive TY table (not UV) editor - run-time help
HLPEDITY	Interactive TY table uv-data editor EDITA - run-time help
HLPEDIUV	Interactive uv-data editor EDITR - run-time help

HLPFTFLG	Interactive time-channel visibility Editor - run-time help
HLPUFLAG	Edit uv-data on a grid UFLAG - run-time help
HLPWIPER	edit uv data from UVPLT-like plot WIPER - run-time help
IBLED	Interactive BaseLine based visibility EDitor
INTTIME	adds user-generated lines to the history extension file
ISCALIB	states that the current source is a point-source calibrator
NOISE	estimates the noise in images, noise level cutoff
OFLAG	uses on-line flag table information to write a flag table
OUTFGVER	selects version of the flagging table to be written
PCEDT	Interactive TV task to edit pulse-cal (PC) tables
PCFLG	interactive flagging of Pulse-cal data in channel-TB using the TV
PDEDT	Interactive TV task to edit polarization D-term (PD) tables
REASON	The reason for an operation
REFLG	Attempts to compress a flag table
RFI	Look for RFI in uv data
RFLAG	Flags data set based on time and freq rms in fringe visibilities
SCANLENG	specify length of "scan"
SCIMG	Full-featured imaging plus self-calibration loop with editing
SCMAP	Imaging plus self-calibration loop with editing
SCUTOFF	noise level cutoff
SDLSF	least squares fit to channels and subtracts from SD uv data
SHAHO	Flags Holography data for shadowing
SNEDT	Interactive SN/CL/TY/SY table editor using the TV
SNFLG	Writes flagging info based on the contents of SN files
SPFLG	interactive flagging of UV data in channel-TB using the TV
TABED	Task to edit tables
TAFLG	Flags data in a Table extension file
TVFLG	interactive flagging of UV data using the TV
UFLAG	Plots and edits data using a uv-plane grid and the TV
UVFLG	Flags UV-data
UVFND	prints selected data from UV data set to search for problems
UVLIN	Fits and removes continuum visibility spectrum, also can flag
UVLSD	least squares fit to channels and divides the uv data.
UVLSF	least squares fit to channels and subtracts from uv data.
UVMLN	edits data based on the rms of the spectrum
VPFLG	Resets flagging to all or all corss-hand whenever some are flagged
WETHR	Plots selected contents of WX tables, flags data based on WX
WIPER	plots and edits data from a UV data base using the TV

13.10 EXT-APPL

BSPRT	print BS tables
GETTHEAD	Returns keyword values from a table header, also column #, # rows
MF2ST	Task to generate an ST ext. file from Model Fit ext. file
PRTSY	Task to print statistics from the SY table
PUTTHEAD	inserts a given value into a table keyword/value pair
SYHIS	Task to clip, filter, smooth SY tables and plot histograms
SY2TY	Task to generate a TY extension file from an EVLA SY table
SYPRT	Task to print values from the SY table
TABGET	returns table entry for specified row, column and subscript.
TABPUT	replaces table entry for specified row, column and subscript.

13.11 FITS

CALRD	Reads calibrator source model-image FITS file
CALWR	writes calibrator source model images w CC files to FITS disk files
DATAIN	specifies name of input FITS disk file
DATAOUT	specifies name of output FITS disk file
FILEZAP	Delete an external file
FIT2A	reads the fits input file and records it to the output ascii file
FITAB	writes images / uv data w extensions to tape in FITS format
FITDISK	writes images / uv data w extensions to disk in FITS format
FITLD	Reads FITS files to load images or UV (IDI or UVFITS) data to disk
FITTP	writes images / uv data w extensions to tape in FITS format
IMLOD	Reads tape or disk files to load images to disk
MPDIFS	Retains comparison differences between current images and master ones
OLDVOL	used by READISK to specify the disk number when WRITDISK was run
READISK	Reads images / uv data w extensions in FITS format into AIPS
TCOPY	Tape to tape copy with some disk FITS support
TERSE	Controls the amount of information printed by the Y2K tests
UVLOD	Read export or FITS data from a tape or disk
VLBALOAD	Loads VLBA data
WRTDISK	writes images / uv data w extensions to disk in FITS format
WRTPROCS	Procedures to simplify the reduction of VLBA data
Y2KDISK	Specifies logical name of disk area for Y2K to read or write
Y2KFILE	File name of the current Y2K disk file
Y2KSIZE	Specifies the size of the Y2K test to be run

13.12 GENERAL

ABORTASK	stops a running task
ABOUT	displays lists and information on tasks, verbs, adverbs
AIPS	AIPS main program for interactive use
AIPSB	AIPS main program for executing batch jobs
ANTNAME	A list of antenna (station) names
APARM	General numeric array adverb used many places
BADDISK	specifies which disks are to be avoided for scratch files
BATCHJOB	Information about BATCH
BCOUNT	gives beginning location for start of a process
BITER	gives beginning point for some iterative process
BLC	gives lower-left-corner of selected subimage
BPARM	general numeric array adverb used too many places
CATEGORY	List of allowed primary keywords in HELP files
CLNUP	removes old MS, RL, SG, TG and ?? files from AIPS disks
CLRMSG	deletes messages from the user's message file
COMMENT	64-character comment string
CPARM	general numeric array adverb used many places
CPUTIME	displays curren tcpu and real time usage of the AIPS task
CROWDED	allows a task to perform its function in a crowded fashion
DCODE	General string adverb
DDISK	Deterimins where input Y2K data are found
DECIMAL	specifies if something is in decimal format
DETIME	specifies a time interval for an operation (destroy, batch)
DISKU	shows disk use by one or all users
DOALL	specifies if an operation is done once or for all matching
DOCONFRM	selects user confirmation modes of repetitive operation
DOKEEP	specifies if something is kept or deleted
DOSCALE	specifies if a scaling operation of some sort is to be performed
DOSCAN	specifies if a scan-related operation is to be done

DOWAIT	selects wait-for-completion mode for running tasks
DOWEIGHT	selects operations with data weights
DPARM	General numeric array adverb used many places
DRCHK	stand-alone program checks system setup files for consistency
ECOUNT	give the highest count or iteration for some process
EDGSKP	Deterimins border excluded from comparision or use
EHEX	converts decimal to extended hex
EXPERT	specifies an user experience level or mode
EXPLAIN	displays help + extended information describing a task/symbol
FILEZAP	Delete an external file
FPARM	General numeric array adverb used in modeling
FREESPAC	displays available disk space for AIPS in local system
GET	restores previously SAVEd full POPS environment
GETDATE	Convert the current date and time to a string
GNUGPL	Information about GNU General Public License for AIPS
GO	starts a task, detaching it from AIPS or AIPSB
GRADDRES	specifies user's home address for replies to gripes
GRDROP	deletes the specified gripe entry
GREMAIL	gives user's e-mail address name for reply to gripe entry
GRINDEX	lists users and time of all gripe entries
GRIPE	enter a suggestion or bug report for the AIPS programmers
GRIPR	standalone program to enter suggestions/complaints to AIPS
GRLIST	lists contents of specified gripe entry
GRNAME	gives user's name for reply to gripe entry
GRPHONE	specifies phone number to call for questions about a gripe
HELP	displays information on tasks, verbs, adverbs
HINOTE	adds user-generated lines to the history extension file
HITEXT	writes lines from history extension file to text file
IN2FILE	specifies name of a disk file, outside the regular catalog
INFILE	specifies name of a disk file, outside the regular catalog
INTEXT	specifies name of input text file, not in regular catalog
IOTAPE	Deterimins which tape drive is used during a Y2K RUN
LSAPROPO	Data input to APROPO to find what uses what words
MDISK	Determins where input Y2K data are found
MSGKILL	turns on/off the recording of messages in the message file
MSGSERVER	Information about the X11-based message server
MSGSRV	Information about the X11-based message server
NBOXES	Number of boxes
NCCBOX	Number of clean component boxes
NCOUNT	General adverb, usually a count of something
NITER	The number of iterations of a process.
NPOINTS	General adverb giving the number of something
NTHREAD	Controls number of threads used by multi-threaded processes in OBIT
OBJECT	The name of an object
OFFSET	General adverb, the offset of something.
OPCODE	General adverb, defines an operation
OPTELL	The operation to be passed to a task by TELL
OPTYPE	General adverb, defines a type of operation.
ORDER	Adverb used usually to specify the order of polynomial fit
OTFBS	Translates on-the-fly continuum SDD format to AIPS UV file
OTFUV	Translates on-the-fly single-dish SDD format to AIPS UV file
OUTFILE	specifies name of output disk file, not in regular catalog
OUTTEXT	specifies name of output text file, not in regular catalog
OUTVERS	The output version number of an extension file (table).
PANIC	Instructions for what to do when things go wrong
PIX2VAL	An image value in the units specified in the header.
PIXRANGE	Range of pixel values to display
PIXVAL	Value of a pixel
POSTSCRIP	General comments about AIPS use of PostScript incl macros

PRTAC	prints contents and summaries of the accounting file
PRTASK	Task name selected for printed information
PRTHI	prints selected contents of the history extension file
PRTLIMIT	specifies limits to printing functions
PRTMSG	prints selected contents of the user's message file
QUAL	Source qualifier
READLINE	Information about AIPS use of the GNU readline library.
REBYTE	service program to transform byte order of full data sets
REHEX	converts extended hex string to decimal
ROTATE	Specifies a rotation
RPARM	General numeric array adverb used in modeling
RTIME	Task to test compute times
SCALR1	General adverb
SCALR2	General adverb
SCALR3	General adverb
SECONDARY	List of allowed secondary keywords in HELP files
SECONDRY	List of allowed secondary keywords in HELP files
SOURCES	A list of source names
SPARM	General string array adverb
SRCALIAS	A list of source names, used as an alias
SRCNAME	A single source name
STALIN	revises history by deleting lines from history extension file
STRA1	General string adverb
STRA2	General string adverb
STRA3	General string adverb
STRB1	General string adverb
STRB2	General string adverb
STRB3	General string adverb
STRC1	General string adverb
STRC2	General string adverb
STRC3	General string adverb
SUBARRAY	Subarray number
SYMBOL	General adverb, probably defines a plotting symbol type
SYS2COM	specifies a command to be sent to the operating system
SYSCOM	specifies a command to be sent to the operating system
SYSOUT	specifies the output device or file used by the system
SYSTEM	Verb to send a command to the operating system
TCODE	Deterimins which type of Y2K is RUN.
TDISK	Deterimins where output Y2K data is placed
TELL	Send parameters to tasks that know to read them on the fly
TGREP	finds all user numbers which have used specified task
THEDATE	contains the date and time in a string form
TIMERANG	Specifies a time range
TMASK	Deterimins which tasks are executed when a Y2K is RUN.
TMODE	Deterimins which input is used when a Y2K is RUN.
TNAMF	Internal variable for Y2K test suite
TPMON	Information about the TPMON "Daemon"
VCODE	General string adverb
VLBDDT	Verification tests using simulated data
VPARM	General numeric array adverb used in modeling
WHATSNEW	lists changes and new code in the last several AIPS releases
XHELP	Accesses hypertext help system
XPARM	General adverb for up to 10 parameters, may refer to X coord
XTYPE	X axis (U) convolving function type
Y2K	verifies correctness and performance using standard problems
Y2KSAVE	verifies correctness and performance using standard problems
YINC	Y axis increment
YTYPE	Y axis (V) convolving function type

13.13 HARDCOPY

BPRINT	gives beginning location for start of a printing process
BSPRT	print BS tables
DOCELL	selects units of cells over angular unit
EPRINT	gives location for end of a printing process
FACTOR	scales some display or CLEANing process
FGPRT	Prints counts of selected portions of an FG table.
FLMCOMM	Comment for film recorder image.
HIEND	End record number in a history-file operation
HISTART	Start record number in a history-file operation
HOLIS	Prints information about holography sampling
INTEXT	specifies name of input text file, not in regular catalog
ISPEC	Plots and prints spectrum of region of a cube
NPLOTS	gives number of plots per page or per job
NPRINT	gives number of items to be printed
OTFIN	Lists on-the-fly single-dish SDD format data files
OUTFILE	specifies name of output disk file, not in regular catalog
OUTPRINT	Specifies name of disk file to keep the printer output
OUTTEXT	specifies name of output text file, not in regular catalog
PLRFI	Plots spectral statistics from output of VBRFI & VLBRF
POSTSCRIP	General comments about AIPS use of PostScript incl macros
PRINTER	Verb to set or show the printer(s) used
PRIORITY	Limits prioroty of messages printed
PRNUMBER	POPS number of messages
PRRFI	Prints excess spectral statistics from output of VBRFI & VLBRF
PRSTART	First record number in a print operation
PRTASK	Task name selected for printed information
PRTIME	Time limit
PRTLEV	Specifies a code for the amount of information requested.
PRTSD	prints contents of AIPS single-dish data sets
PRTUV	prints contents of a visibility (UV) data set
REASON	The reason for an operation
RGBGAMMA	specifies the desired color gamma corrections
RSPEC	Plots and prints spectrum of rms of a cube
TVCPS	Task to copy a TV screen-image to a PostScript file.
TVDIC	Task to copy a TV screen-image to a Dicomed film recorder.
UVFND	prints selected data from UV data set to search for problems
UVHOL	Prints holography data from a UV data base with calibration
UVPRT	prints data from a UV data base with calibration
VBRFI	Plots spectral statistics of VLBA autocorrelation data for RFI
VLBRF	Plots spectral statistics of VLBA autocorrelation data for RFI

13.14 IMAGE-UT

BDROP	gives number of pooints dropped at the beginning
BSGEO	Beam-switched Az-El image to RA-Dec image translation
CALWR	writes calibrator source model images w CC files to FITS disk files
DOMODEL	selects display of model function
DORESID	selects display of differences between model and data
DOSLICE	selects display of slice data
DSKEW	Geometric interpolation correction for skew
EDROP	number of points/iterations to be omitted from end of process
FIT2A	reads the fits input file and records it to the output ascii file
FITAB	writes images / uv data w extensions to tape in FITS format
FITDISK	writes images / uv data w extensions to disk in FITS format

FITTP	writes images / uv data w extensions to tape in FITS format
HLPTVHUI	Interactive intensity-hue-saturation display - run-time help
HLPTVIEW	Interactive display of cube in 3 transpositions - run-time help
HLPTVRGB	Interactive red-green-blue display - run-time help
HLPTVSPC	Interactive display of spectra from a cube - run-time help
IMCLP	Clip an image to a specified range.
IMLOD	Reads tape or disk files to load images to disk
OGEOM	Simple image rotation, scaling, and translation
OHGEO	Geometric interpolation with correction for 3-D effects
READISK	Reads images / uv data w extensions in FITS format into AIPS
TVIEW	Display image cube in all 3 transpositions with rotation & reference
TVSPC	Display images and spectra from a cube including fitting of spectra
WRTDISK	writes images / uv data w extensions to disk in FITS format
WRTPROCS	Procedures to simplify the reduction of VLBA data

13.15 IMAGE

CPYRT	replaces history with readme file, inserts copyright
IMRMS	Plot IMEAN rms answers
MAPBM	Map VLA beam polarization
NANS	reads an image or a UV data set and looks for NaNs
PROFL	Generates plot file for a profile display.
STRAN	Task compares ST tables, find image coordinates (e.g. guide star)
VLABP	VLA antenna beam polarization correction for snapshot images
XSMTH	Smooth data along the x axis
XSUM	Sum or average images on the x axis
XTRAN	Create an image with transformed coordinates

13.16 IMAGING

AHIST	Task to convert image intensities by adaptive histogram
ALLOKAY	specifies that initial conditions have been met.
APCLN	Deconvolves images with CLEAN algorithm
APGS	deconvolves image with Gerchberg-Saxton algorithm
APVC	Deconvolves images with van Cittert algorithm
AVOPTION	Controls type or range of averaging done by a task
BCOMP	gives beginning component number for multiple fields
BLC	gives lower-left-corner of selected subimage
BLWUP	Blow up an image by any positive integer factor.
BMAJ	gives major axis size of beam or component
BMIN	gives minor axis size of beam or component
BOX	specifies pixel coordinates of subarrays of an image
BOX2CC	Converts CLBOX in pixels to CCBOX in arc seconds
BOXES	Adds Clean boxes to BOXFILE around sources.
BOXFILE	specifies name of Clean box text file
BPA	gives position angle of major axis of beam or component
BSAVG	Task to do an FFT-weighted sum of beam-switched images
BSCLN	Hogbom Clean on beam-switched difference image
BSGRD	Task to image beam-switched single-dish data
BSMAP	Images weak sources with closure phases
CANDY	user-definable (paraform) task to create an AIPS image
CCBOX	specifies pixel coordinates of subarrays of an image
CCEDT	Select CC components in BOXes and above mininum flux.
CCFND	prints the contents of a Clean Components extension file.
CCGAU	Converts point CLEAN components to Gaussians
CCMOD	generates clean components to fit specified source model

CCMRG	sums all clean components at the same pixel
CCRES	Removes or restores a CC file to a map with a gaussian beam.
CELLSIZE	gives the pixel size in physical coordinates
CHKFC	makes images of Clean boxes from Boxfile
CLBOX	specifies subarrays of an image for Clean to search
CMETHOD	specifies the method by which the uv model is computed
CMODEL	specifies the method by which the uv model is computed
COHER	Baseline Phase coherence measurement
CONPL	Plots AIPS gridding convolution functions
CONVL	convolves an image with a gaussian or another image
CUTOFF	specifies a limit below or above which the operation ends
CXCLN	Complex Hogbom CLEAN
CXIMAGR	Grid UV data into a complex image, Fourier transform, Clean
DCONV	deconvolves a gaussian from an image
DECSHIFT	gives Y-coordinate shift of an image center from reference
DELBOX	Verb to delete boxes with TV cursor & graphics display.
DFILEBOX	Verb to delete Clean boxes with TV cursor & write to file
DO3DIMAG	specifies whether uvw's are reprojected to each field center
DOGRIDCR	selects correction for gridding convolution function
DOOSRO	calibrating amplitude and phase, and imaging VLA data
DRAWBOX	Verb to draw Clean boxes on the display
DTSUM	Task to provide a summary of the contents of a dataset
EVAUV	Subtracts & divides a model into UV data, does statistics on results
FACES	makes images of catalog sources for initial calibration
FACTOR	scales some display or CLEANing process
FETCH	Reads an image from an external text file.
FFT	Takes Fourier Transform of an image or images
FGAUSS	Minimum flux to Clean to by widths of Gaussian models
FILEBOX	Verb to reset Clean boxes with TV cursor & write to file
FILIT	Interactive BOXFILE editing with facet images
FIXBX	converts a BOXFILE to another for input to IMAGR
FLATN	Re-grid multiple fields into one image incl sensitivity
FLDSIZE	specifies size(s) of images to be processed
FLUX	gives a total intensity value for image/component or to limit
FOV	Specifies the field of view
FSHIFT	specifies a position shift - output from fitting routines
FSIZE	file size in Megabytes
GAIN	specifies loop gain for deconvolutions
GUARD	portion of UV plane to receive no data in gridding
HA2TI	Converts data processed by TI2HA (STUFFER) back to real times
HISEQ	task to translate image by histogram equalization
HLPCLEAN	Cleaning tasks - run-time help
HLPSCIMG	Full-featured image plus self-cal loops, editing - run-time help
HLPSCMAP	Imaging plus self-cal and editing SCMAP - run-time help
HOCLN	Task to do a clean of holography images
HOLGR	Read & process holography visibility data to telescope images
HOLOG	Read & process holography visibility data to telescope images
IM2CC	Task to convert an image to multi-facet Clean Components
IM2PARM	Specifes enhancement parameters for OOP-based imaging: 2nd set
IM2UV	converts an image to a visibility data set
IMAGR	Wide-field and/or wide-frequency Cleaning / imaging task.
IMAGRPRM	Specifes enhancement parameters for OOP-based imaging
IMERG	merges images of different spatial resolutions
IMSIZE	specifies number of pixels on X and Y axis of an image
IMTXT	Write an image to an external text file.
INLIST	specifies name of input disk file, usually a source list
LINIMAGE	Build image cube from multi-IF data set
LISPX	Fits spectral indexes to a list of spectra, allows interactive edit
LTESS	makes mosaic images by linear combination
MANDL	creates an image of a subset of the Mandlebrot Set

MAPPR	Simplified access to IMAGR
MAPSHIFT	used by MAPPR to specify the shift of the image from the phase center
MASKS	Makes mask image of Clean boxes from boxfile
MAXPIXEL	maximum pixels searched for components in Clark CLEAN
MOD3D	Computes a 3D CC model from a set of facets
MODVF	task to create a warped velocity field
MWFLT	applies linear & non-linear filters to images
NBOXES	Number of boxes
NCCBOX	Number of clean component boxes
NCOMP	Number of CLEAN components
NDIG	Number of digits to display
NFIELD	The number of fields imaged
NMAPS	Number of maps (images) in an operation
NOIFS	makes all IFs into single spectrum
NOISE	estimates the noise in images, noise level cutoff
OBITIMAG	Access to OBIT task Imager without self-cal or peeling
OBITMAP	Simplified access to OBIT task Imager
OBITPEEL	Access to OBIT task Imager with self-cal and peeling
OBITSCAL	Access to OBIT task Imager with self-cal, NOT peeling
OBITVERS	Python version to use with BDF2AIPS and OBIT imaging verbs
OBOXFILE	specifies name of output Clean box text file
ONEBEAM	specifies whether one beam is made for all facets or one for each
ONEFREQ	states that the current CC model was made with one frequency
OOSUB	Subtracts/divides a model from/into a uv data base
OUT2CLAS	The class of a secondary output file
OUT2DISK	The disk number of a secondary output file.
OUT2NAME	The name of a secondary output file.
OUT2SEQ	The sequence of a secondary output file.
OVERLAP	specifies how overlaps are to be handled
OVRSWTCH	specifies when IMAGR switches from OVERLAP ¿= 2 to OVERLAP = 1 mode
PADIM	Task to increase image size by padding with some value
PASTE	Pastes a selected subimage of one image into another.
PATGN	Task to create a user specified test or primary-beam pattern
PBCOR	Task to apply or correct an image for a primary beam
PBPARM	Primary beam parameters
PBSIZE	estimates the primary beam size in interferometer images
PGEOM	Task to transform an image into polar coordinates.
PHASE	Baseline Phase coherence measurement
PHAT	Prussian hat size
PIPEAIPS	calibrating amplitude and phase, and imaging VLA data
PIX2VAL	An image value in the units specified in the header.
PIX2XY	Specifies a pixel in an image
PIXAVG	Average image value
PIXRANGE	Range of pixel values to display
PIXSTD	RMS pixel deviation
PIXVAL	Value of a pixel
PIXXY	Specifies a pixel location.
PPARM	used by PEELR to list the facets to be "peeled"
PRTCC	prints the contents of a Clean Components extension file.
PUTVALUE	Verb to store a pixel value at specified position
QCREATE	adverb controlling the way large files are created
QUANTIZE	Quantization level to use
REBOX	Verb to reset boxes with TV cursor & graphics display.
REGRD	Regrids an image from one co-ordinate frame to another
REMAG	Task to replace magic blanks with a user specified value
RMSD	Calculate rms for each pixel using data in a box around the pixel
ROBUST	Uniform weighting "robustness" parameter
RSTOR	Restores a Clean component file to a map with a Gaussian beam.
SABOX	create box file from source islands in facet images

SCIMG	Full-featured imaging plus self-calibration loop with editing
SCMAP	Imaging plus self-calibration loop with editing
SDCLN	deconvolves image by Clark and then "SDI" cleaning methods
SDGRD	Task to select and image random-position single-dish data
SDIMG	Task to select and image random-position single-dish data
SETFC	Makes a BOXFILE for input to IMAGR, recommends cell and image sizes.
SHADW	Generates the "shadowed" representation of an image
SHIFT	specifies a position shift
SIZEFILE	return file size plus estimate of IMAGR work file size
SKEW	Specifies a skew angle
SKYVE	Regrids a DSS image from one co-ordinate frame to another
SMODEL	Source model
SPCOR	Task to correct an image for a primary beam and spectral index
SPECR	Spectral regridding task for UV data
SPFIX	Makes cube from input to and output from SPIXR spectral index
SPIXR	Fits spectral indexes to each row of an image incl curvature
STACK	Task to co-add a set of 2-dimensional images with weighting
STEER	Task which deconvolves the David Steer way.
STESS	Task which finds sensitivity in mosaicing
STFACTOR	scales star display or SDI CLEANing process
STUFFR	averages together data sets in hour angle
SUBIM	Task to select a subimage from up to a 7-dim. image
SUMIM	Task to sum overlapping, sequentially-numbered images
TDEPEND	Time-dependent imaging procedure sequence
TDSCANS	Time-dependent imaging procedure sequence: find intervals
TDSSCAN	Time-dependent imaging procedure sequence: find intervals
TDSTEP3	Time-dependent imaging procedure "step 3"
TDSTEP5	Time-dependent imaging procedure sequence: later steps
TKBOX	Procedure to set a Clean box with the TK cursor
TKNBOXS	Procedure to set Clean boxes 1 - n with the TK cursor
TRANS	Task to transpose a subimage of an up to 7-dim. image
TRANSCOD	Specified desired transposition of an image
TRC	Specified the top right corner of a subimage
TVBOX	Verb to set boxes with TV cursor & graphics display.
TVSPX	Fits spectral indexes to each image row, allows interactive edit
UBAVG	Baseline dependent time averaging of uv data
UNSPX	Removes spectral index model from a data cube
UTESS	deconvolves images by maximizing emptiness
UVBOX	radius of the smoothing box used for uniform weighting
UVBXFN	type of function used when counting for uniform weighting
UVFRE	Makes one data set have the spectral structure of another
UVIMG	Grid UV data into an "image"
UVMAP	makes images from calibrated UV data.
UVPOL	modifies UV data to make complex image and beam
UVSIZE	specifies number of pixels on X and Y axes of a UV image
UVSUB	Subtracts/divides a model from/into a uv data base
UVWTFN	Specify weighting function, Uniform or Natural
VLARUN	Calibrating amplitude and phase, and imaging VLA data
VTESS	Deconvolves sets of images by the Maximum Entropy Method
WGAUSS	Widths of Gaussian models (FWHM)
WTSUM	Task to do a a sum of images weighted by other images
XMOM	Fits one-dimensional moments to each row of an image
YPARM	Specifies Y axis convolving function
ZEROSP	Specify how to include zero spacing fluxes in FT of UV data

13.17 INFORMAT

ASTROMET	Describes the process of astrometric/geodetic reduction in AIPS
BATCHJOB	Information about BATCH
CALIBRAT	describes the process of data calibration in AIPS
CATEGORY	List of allowed primary keywords in HELP files
GNUGPL	Information about GNU General Public License for AIPS
LSAPROPO	Data input to APROPO to find what uses what words
MSGSERVER	Information about the X11-based message server
MSGSRV	Information about the X11-based message server
NEWTASK	Information about installing a new task
NOADVERB	Information about the lack of a defined adverb or verb
PANIC	Instructions for what to do when things go wrong
POPSDAT	lists all POPS symbols to create them for MEmory files
POPSYM	Describes the symbols used in POPS
POSTSCRIP	General comments about AIPS use of PostScript incl macros
READLINE	Information about AIPS use of the GNU readline library.
REBYTE	service program to transform byte order of full data sets
SECONDARY	List of allowed secondary keywords in HELP files
SECONDRY	List of allowed secondary keywords in HELP files
TEKSERVER	Information about the X-11 Tektronix emulation server
TEKSRV	Information about the X-11 Tektronix emulation server
TPMON	Information about the TPMON "Daemon"
USERLIST	Alphabetic and numeric list of VLA users, points to real list
UV1TYPE	Convolving function type 1, pillbox or square wave
UV2TYPE	Convolving function type 2, exponential function
UV3TYPE	Convolving function type 3, sinc function
UV4TYPE	Convolving function type 4, exponent times sinc function
UV5TYPE	Convolving function type 5, spheroidal function
UV6TYPE	Convolving function type 6, exponent times BessJ1(x) / x
WHATSNEW	lists changes and new code in the last several AIPS releases
XAS	Information about TV-Servers

13.18 INTERACT

AIPS	AIPS main program for interactive use
BPEDT	Interactive TV task to edit uv data based on BP tables
DELBOX	Verb to delete boxes with TV cursor & graphics display.
DFILEBOX	Verb to delete Clean boxes with TV cursor & write to file
EDITA	Interactive TV task to edit uv data based on TY/SY/SN/CL tables
EDITR	Interactive baseline-oriented visibility editor using the TV
FILEBOX	Verb to reset Clean boxes with TV cursor & write to file
FTFLG	interactive flagging of UV data in channel-time using the TV
HLPAGAUS	Interactive Gaussian absorption fitting task AGAUS - run-time help
HLPCLEAN	Cleaning tasks - run-time help
HLPEDIBP	Interactive BP table uv-data editor BPEDT - run-time help
HLPEDICL	Interactive SN/CL table uv-data editor - run-time help
HLPEDIPC	Interactive PC table editor PCEDT - run-time help
HLPEDIPD	Interactive PD table editor PDEDT - run-time help
HLPEDISN	Interactive SN/CL table (not UV) editor - run-time help
HLPEDISS	Interactive SY table (not UV) editor - run-time help
HLPEDISY	Interactive SY table uv-data editor EDITA - run-time help
HLPEDITS	Interactive TY table (not UV) editor - run-time help
HLPEDITY	Interactive TY table uv-data editor EDITA - run-time help
HLPEDIUV	Interactive uv-data editor EDITR - run-time help
HLPFILIT	Interactive Clean box file editing with image display - run-time help
HLPFTFLG	Interactive time-channel visibility Editor - run-time help

HLPIBLED	Interactive Baseline based visibility Editor - run-time help
HLPLISPX	Fits spectral index model to data in text file - run-time help
HLPPCFLG	Interactive time-channel PC table Editor PCFLG - run-time help
HLPPLAYR	OOP TV class demonstration task - run-time help
HLPRMFIT	Polarization fitting task RMFIT - run-time help
HLPSCIMG	Full-featured image plus self-cal loops, editing - run-time help
HLPSCMAP	Imaging plus self-cal and editing SCMAP - run-time help
HLPSPFLG	Interactive time-channel visibility Editor SPFLG - run-time help
HLPTVFLG	Interactive time-baseline visibility Editor TVFLG - run-time help
HLPTVHLD	Interactive image display with histogram equalization - run-time help
HLPTVHUI	Interactive intensity-hue-saturation display - run-time help
HLPTVIEW	Interactive display of cube in 3 transpositions - run-time help
HLPTVRGB	Interactive red-green-blue display - run-time help
HLPTVSAD	Find & fit Gaussians to an image with interaction - run-time help
HLPTVSPC	Interactive display of spectra from a cube - run-time help
HLPTVSPX	Fits 1-dimensional spectral index model to data - run-time help
HLPUFLAG	Edit uv-data on a grid UFLAG - run-time help
HLPWIPER	edit uv data from UVPLT-like plot WIPER - run-time help
HLPXGAUS	Interactive Gaussian fitting task XGAUS - run-time help
HLPZAMAN	Fits 1-dimensional Zeeman model to absorption data - run-time help
HLPZEMAN	Fits 1-dimensional Zeeman model to data - run-time help
IBLED	Interactive BaseLine based visibility EDitor
IMAGR	Wide-field and/or wide-frequency Cleaning / imaging task.
LISPX	Fits spectral indexes to a list of spectra, allows interactive edit
MAPPR	Simplified access to IMAGR
MFITSET	gets adverbs for running IMFIT and JMFIT
OFMPL	Task to display and contrast system and user OFM color tables.
OPTELL	The operation to be passed to a task by TELL
PCEDT	Interactive TV task to edit pulse-cal (PC) tables
PCFLG	interactive flagging of Pulse-cal data in channel-TB using the TV
PDEDT	Interactive TV task to edit polarization D-term (PD) tables
PLAYR	Task to load one or two images and play with them on the TV
READ	Read a value from the users terminal
READLINE	Information about AIPS use of the GNU readline library.
REBOX	Verb to reset boxes with TV cursor & graphics display.
SCIMG	Full-featured imaging plus self-calibration loop with editing
SCMAP	Imaging plus self-calibration loop with editing
SETSLICE	Set slice endpoints on the TV interactively
SNEDT	Interactive SN/CL/TY/SY table editor using the TV
SPFLG	interactive flagging of UV data in channel-TB using the TV
TDEPEND	Time-dependent imaging procedure sequence
TDSCANS	Time-dependent imaging procedure sequence: find intervals
TDSSCAN	Time-dependent imaging procedure sequence: find intervals
TDSTEP3	Time-dependent imaging procedure "step 3"
TDSTEP5	Time-dependent imaging procedure sequence: later steps
TK1SET	Verb to reset 1D gaussian fitting initial guess.
TKBOX	Procedure to set a Clean box with the TK cursor
TKNBOXS	Procedure to set Clean boxes 1 - n with the TK cursor
TKPOS	Read a position from the graphics screen or window
TKSET	Verb to set 1D gaussian fitting initial guesses.
TKWIN	Procedure to set BLC and TRC with Graphics cursor
TV1SET	Verb to reset 1D gaussian fitting initial guess on TV plot.
TVBOX	Verb to set boxes with TV cursor & graphics display.
TVFLG	interactive flagging of UV data using the TV
TVHELIX	Verb to activate a helical hue-intensity TV pseudo-coloring
TVIEW	Display image cube in all 3 transpositions with rotation & reference
TVSAD	Finds and fits Gaussians to portions of an image with interaction
TVSCROL	Shift position of image on the TV screen
TVSET	Verb to set slice Gaussian fitting initial guesses from TV plot
TVSPC	Display images and spectra from a cube including fitting of spectra
TVSPLIT	Compare two TV image planes, showing halves

TVSPX	Fits spectral indexes to each image row, allows interactive edit
TVSTAT	Find the mean and RMS in a blotch region on the TV
TVTRANSF	Interactively alters the TV image plane transfer function
TVWINDOW	Set a window on the TV with the cursor
TVZOOM	Activate the TV zoom
UFLAG	Plots and edits data using a uv-plane grid and the TV
WEDERASE	Load a wedge portion of the TV with zeros
WIPER	plots and edits data from a UV data base using the TV

13.19 MODELING

ACTNOISE	puts estimate of actual image uncertainty and zero in header
AGAUS	Fits 1-dimensional Gaussians to absorption-line spectra
BLPCL	Do divide by I,Q,U models for circular polarization BLCHN
BLPOLCAL	Do BLCHN for I, Q, U, V models
BOXES	Adds Clean boxes to BOXFILE around sources.
BSMOD	creates single-dish UV beam-switched data with model sources
BSTST	Graphical display of solutions to frequency-switched data
CHEBI	Make text file to plot Chebyshev polynomials
CMETHOD	specifies the method by which the uv model is computed
CMODEL	specifies the method by which the uv model is computed
CONFI	Optimize array configuration by minimum side lobes
CUBIT	Model a galaxy's density and velocity distribution from full cube
DGAUS	Correct UV model for convolution by Clean beam
DIFUV	Difference two matching uv data sets, write out or print statistics
DOMODEL	selects display of model function
DORESID	selects display of differences between model and data
DOSPIX	selects solutions for spectral index of model components
EFACTOR	scales some error analysis process
EVAUV	Subtracts & divides a model into UV data, does statistics on results
FACES	makes images of catalog sources for initial calibration
FITOUT	specifies name of output text file for results of fitting
FSHIFT	specifies a position shift - output from fitting routines
FTARS	Fits 1-dimensional polarization spectrum to Q/U text file
GLENS	models galaxy gravitational lens acting on 3 component source
HLPAGAUS	Interactive Gaussian absorption fitting task AGAUS - run-time help
HLPLISPX	Fits spectral index model to data in text file - run-time help
HLPRMFIT	Polarization fitting task RMFIT - run-time help
HLPTVSAD	Find & fit Gaussians to an image with interaction - run-time help
HLPTVSPX	Fits 1-dimensional spectral index model to data - run-time help
HLPXGAUS	Interactive Gaussian fitting task XGAUS - run-time help
HLPZAMAN	Fits 1-dimensional Zeeman model to absorption data - run-time help
HLPZEMAN	Fits 1-dimensional Zeeman model to data - run-time help
IM2TX	Sums an image region into a 1-dimensional text file
IMFIT	Fits Gaussians to portions of an image
IMMOD	adds images of model objects to an image
INLIST	specifies name of input disk file, usually a source list
JMFIT	Fits Gaussians to portions of an image
MFITSET	gets adverbs for running IMFIT and JMFIT
MFPRT	prints MF tables in a format needed by modelling software
MOD3D	Computes a 3D CC model from a set of facets
MODAB	Makes simple absorption/emission spectral-line image in I/V
MODIM	adds images of model objects to image cubes in IQU polarization
MODSP	adds images of model objects to image cubes in I/V polarization
MODVF	task to create a warped velocity field
OMFIT	Fits sources and, optionally, a self-cal model to uv data
ONEFREQ	states that the current CC model was made with one frequency
OOSUB	Subtracts/divides a model from/into a uv data base

RADIUS	Specify a radius in an image
RM2PL	Plots spectrum of a pixel with RMFIT fit
RMFIT	Fits 1-dimensional polarization spectrum to Q/U cube
RMSLIMIT	selects things with RMS above this limit
SAD	Finds and fits Gaussians to portions of an image
SDMOD	modifies single-dish UV data with model sources
SLFIT	Task to fit gaussians to slice data.
SPMOD	Modify UV database by adding a model with spectral lines
SPXMD	adds images of model objects to image cubes in spectral index
STVERS	star display table version number
TK1SET	Verb to reset 1D gaussian fitting initial guess.
TKAMODEL	Verb to add slice model display directly on TEK
TKARESID	Verb to add slice model residuals directly on TEK
TKGUESS	Verb to display slice model guess directly on TEK
TKMODEL	Verb to display slice model directly on TEK
TKRESID	Verb to display slice model residuals directly on TEK
TKSET	Verb to set 1D gaussian fitting initial guesses.
TKSLICE	Verb to display slice file directly on TEK
TV1SET	Verb to reset 1D gaussian fitting initial guess on TV plot.
TVACOMPS	Verb to add slice model components directly on TV graphics
TVAGUESS	Verb to re-plot slice model guess directly on TV graphics
TVAMODEL	Verb to add slice model display directly on TV graphics
TVARESID	Verb to add slice model residuals directly on TV graphics
TVASLICE	Verb to add a slice display on TV graphics from slice file
TVCOMPS	Verb to display slice model components directly on TV graphics
TVGUESS	Verb to display slice model guess directly on TV graphics
TVMODEL	Verb to display slice model directly on TV graphics
TVRESID	Verb to display slice model residuals directly on TV graphics
TVSAD	Finds and fits Gaussians to portions of an image with interaction
TVSET	Verb to set slice Gaussian fitting initial guesses from TV plot
TVSLICE	Verb to display slice file directly on TV
TX2IM	Converts a 1-dimensional text file into cataloged image
UVFIT	Fits source models to uv data.
UVGIT	Fits source models to uv data.
UVMOD	Modify UV database by adding a model incl spectral index
UVSUB	Subtracts/divides a model from/into a uv data base
VHCAL	Do divide by I,Q,U models for linear polarization data
VHCALIB	Do CALIB for linear polarization with I, Q, U models
XG2PL	Plots spectrum of a pixel with XGAUS/AGAUS and ZEMAN/ZAMAN fits
XG2XG	Changes number of components in an XG or ZE table creating new one
XGAUS	Fits 1-dimensional Gaussians to images: restartable
ZAMAN	Fits 1-dimensional Zeeman model to absorption-line data
ZEMAN	Fits 1-dimensional Zeeman model to data

13.20 OBSOLETE

ABACKUP	VMS procedure to back up data on tape
ARESTORE	Restores back up tapes of users data
CLIPM	Renamed CLIP to flag excessive amplitudes
CROSSPOL	Procedure to make complex poln. images and beam.
DDT	previous verion of Y2K to test performance and correctness of code
DDTSIZE	String formerly used by the DDT test
HORUS	makes images from unsorted UV data, applying any calibration
MAPDIF	Was used by the DDT test suite
MX	makes images & deconvolves using UV data directly - replaced
OFFROAM	Procedure to clear the TV from a Roam condition

PHCLN	PHCLN has been removed, use PHAT adverb in APCLN.
POLSN	Make a SN table from cross polarized fringe fit
PSEUDO	Description of POPS pseudoverbs - obsolete list file
SETROAM	Verb used to set roam image mode, then do roam. OBSOLETE
SNCUT	Specifies minimum signal-to-noise ratio
TIMDEST	Verb to destroy all files which are too old
VLBACPOL	Procedure to calibrate cross-polarization delays
XVSS	Information about older Sun OpenWindows-specific TV-Server

13.21 ONED

AGAUS	Fits 1-dimensional Gaussians to absorption-line spectra
FTARS	Fits 1-dimensional polarization spectrum to Q/U text file
IM2TX	Sums an image region into a 1-dimensional text file
PFPL2	Paraform Task to generate a plot file: (slice intensity)
PLCUB	Task to plot intensity vs x panels on grid of y,z pixels
PLROW	Plot intensity of a series of rows with an offset.
RM2PL	Plots spectrum of a pixel with RMFIT fit
RMFIT	Fits 1-dimensional polarization spectrum to Q/U cube
SETSLICE	Set slice endpoints on the TV interactively
SL2PL	Task to convert a Slice File to a Plot File
SLFIT	Task to fit gaussians to slice data.
SLICE	Task to make a slice file from an image
SLPRT	Task to print a Slice File
TK1SET	Verb to reset 1D gaussian fitting initial guess.
TKAMODEL	Verb to add slice model display directly on TEK
TKARESID	Verb to add slice model residuals directly on TEK
TKASLICE	Verb to add a slice display on TEK from slice file
TKGUESS	Verb to display slice model guess directly on TEK
TKMODEL	Verb to display slice model directly on TEK
TKRESID	Verb to display slice model residuals directly on TEK
TKSET	Verb to set 1D gaussian fitting initial guesses.
TKSLICE	Verb to display slice file directly on TEK
TV1SET	Verb to reset 1D gaussian fitting initial guess on TV plot.
TVACOMPS	Verb to add slice model components directly on TV graphics
TVAGUESS	Verb to re-plot slice model guess directly on TV graphics
TVAMODEL	Verb to add slice model display directly on TV graphics
TVARESID	Verb to add slice model residuals directly on TV graphics
TVASLICE	Verb to add a slice display on TV graphics from slice file
TVCOMPS	Verb to display slice model components directly on TV graphics
TVGUESS	Verb to display slice model guess directly on TV graphics
TVMODEL	Verb to display slice model directly on TV graphics
TVRESID	Verb to display slice model residuals directly on TV graphics
TVSET	Verb to set slice Gaussian fitting initial guesses from TV plot
TVSLICE	Verb to display slice file directly on TV
TX2IM	Converts a 1-dimensional text file into cataloged image
XG2PL	Plots spectrum of a pixel with XGAUS/AGAUS and ZEMAN/ZAMAN fits
XG2XG	Changes number of components in an XG or ZE table creating new one
XGAUS	Fits 1-dimensional Gaussians to images: restartable
XPLOT	Plots image rows one at a time on graphics, TV, or plot file
ZAMAN	Fits 1-dimensional Zeeman model to absorption-line data
ZEMAN	Fits 1-dimensional Zeeman model to data

13.22 OOP

BLING	find residual rate and delay on individual baselines
BPEDT	Interactive TV task to edit uv data based on BP tables
BSCOR	Combines two beam-switched images
BSGEO	Beam-switched Az-El image to RA-Dec image translation
BSGRD	Task to image beam-switched single-dish data
CCEDT	Select CC components in BOXes and above mininum flux.
CCSEL	Select signifigant CC components
DSKEW	Geometric interpolation correction for skew
EDITA	Interactive TV task to edit uv data based on TY/SY/SN/CL tables
EDITR	Interactive baseline-oriented visibility editor using the TV
FILIT	Interactive BOXFILE editing with facet images
FINDR	Find normal values for a uv data set
FIXWT	Modify weights to reflect amplitude scatter of data
FLAGR	Edit data based on internal RMS, amplitudes, weights
FLATN	Re-grid multiple fields into one image incl sensitivity
FRCAL	Faraday rotation self calibration task
HLPCLEAN	Cleaning tasks - run-time help
HLPEDIBP	Interactive BP table uv-data editor BPEDT - run-time help
HLPEDICL	Interactive SN/CL table uv-data editor - run-time help
HLPEDIPC	Interactive PC table editor PCEDT - run-time help
HLPEDIPD	Interactive PD table editor PDEDT - run-time help
HLPEDISN	Interactive SN/CL table (not UV) editor - run-time help
HLPEDISS	Interactive SY table (not UV) editor - run-time help
HLPEDISY	Interactive SY table uv-data editor EDITA - run-time help
HLPEDITS	Interactive TY table (not UV) editor - run-time help
HLPEDITY	Interactive TY table uv-data editor EDITA - run-time help
HLPEDIUV	Interactive uv-data editor EDITR - run-time help
HLPFILIT	Interactive Clean box file editing with image display - run-time help
HLPPLAYR	OOP TV class demonstration task - run-time help
HLPSCIMG	Full-featured image plus self-cal loops, editing - run-time help
HLPSCMAP	Imaging plus self-cal and editing SCMAP - run-time help
IM2PARM	Specifes enhancement parameters for OOP-based imaging: 2nd set
IMAGR	Wide-field and/or wide-frequency Cleaning / imaging task.
IMAGRPRM	Specifes enhancement parameters for OOP-based imaging
IMCLP	Clip an image to a specified range.
IMFRING	large image delay fitting with IM2CC and OOFRING
IMSCAL	large image self-cal with IM2CC and OOCAL
MAPBM	Map VLA beam polarization
MBDLY	Fits multiband delays from IF phases, updates SN table
MULIF	Change number of IFs in output
OGEOM	Simple image rotation, scaling, and translation
OHGEO	Geometric interpolation with correction for 3-D effects
OMFIT	Fits sources and, optionally, a self-cal model to uv data
OOCAL	determines antenna complex gain with frequency-dependent models
OOSRT	Sort a UV dataset into a specified order
OOSUB	Subtracts/divides a model from/into a uv data base
PASTE	Pastes a selected subimage of one image into another.
PCEDT	Interactive TV task to edit pulse-cal (PC) tables
PDEDT	Interactive TV task to edit polarization D-term (PD) tables
PLAYR	Task to load one or two images and play with them on the TV
RFI	Look for RFI in uv data
SABOX	create box file from source islands in facet images
SCIMG	Full-featured imaging plus self-calibration loop with editing
SCMAP	Imaging plus self-calibration loop with editing

SDGRD	Task to select and image random-position single-dish data
SNEDT	Interactive SN/CL/TY/SY table editor using the TV
TDEPEND	Time-dependent imaging procedure sequence
TDSCANS	Time-dependent imaging procedure sequence: find intervals
TDSSCAN	Time-dependent imaging procedure sequence: find intervals
TDSTEP3	Time-dependent imaging procedure "step 3"
TDSTEP5	Time-dependent imaging procedure sequence: later steps
UV2MS	Append single-source file to multi-source file.
VLABP	VLA antenna beam polarization correction for snapshot images

13.23 OPTICAL

COSTAR	Verb to plot a symbol at given position on top of a TV image
GSTAR	Task to read a Guide Star (UK) table and create an ST table.
IMFLT	fits and removes a background intensity plane from an image
STFND	Task to find stars in an image and generate an ST table.
STRAN	Task compares ST tables, find image coordinates (e.g. guide star)
TVSTAR	Verb to plot star positions on top of a TV image
XTRAN	Create an image with transformed coordinates

13.24 PARAFORM

CANDY	user-definable (paraform) task to create an AIPS image
DTCHK	Task to check results of a test using simulated data.
FUDGE	modifies UV data with user's algorithm: paraform task
NEWTASK	Information about installing a new task
OFUDG	modifies UV data with user's algorithm no cal adverbs: paraform task
PFPL1	Paraform Task to generate a plot file: (does grey scale)
PFPL2	Paraform Task to generate a plot file: (slice intensity)
PFPL3	Paraform Task to generate a plot file: (does histogram)
TAFFY	User definable task to operate on an image
TBTSK	Paraform OOP task for tables
UVFIL	Create, fill a uv database from user supplied information

13.25 PLOT

AGAUS	Fits 1-dimensional Gaussians to absorption-line spectra
ALIAS	adverb to alias antenna numbers to one another
ALVAR	Plots the Allan Variance statistic of a UV data set
ANBPL	plots and prints uv data converted to antenna based values
ASPMM	Plot scaling parameter - arc seconds per millimeter on plot
AVGCHAN	Controls averaging of spectral channels
AVGIF	Controls averaging of IF channels
BDROP	gives number of pooints dropped at the beginning
BLPLT	Plots BLCAL tables in 2 dimensions as function of baseline and IF
BLSUM	sums images over irregular sub-images, displays spectra
BPEPL	Plots bandpass tables overlaying multiple times like BPEDT
BPERR	Print and plot BPASS closure outputs
BPLOT	Plots bandpass tables in 2 dimensions as function of time
BPPLT	Plots selected contents of bandpass (BP) files versus time
CAPLT	plots closure amplitude and model from CC file
CBPLOT	selects a display of a Clean beam full width at half maximum

CCNTR	Generate a contour plot file from an image with CC or MF plotted
CHEBI	Make text file to plot Chebyshev polynomials
CIRCLEVS	Sets RGBLEVS to fill LEVS with a circular color scheme
CLEV	Contour level multiplier in physical units
CLOSE	plots closure phase or amplitude spectra averaged over closures
CLPLT	plots closure phase and model from CC file
CNTR	generate a contour plot file or TV plot from an image
CODETYPE	specifies the desired operation or data type
CON3COL	Controls use of full 3-color graphics for contouring
CONPL	Plots AIPS gridding convolution functions
COPIES	sets the number of copies to be made
COSTAR	Verb to plot a symbol at given position on top of a TV image
DARKLINE	The level at which vectors are switched from light to dark
DEFCOLOR	Sets adverb PLCOLORS to match s default XAS TV
DELZN	Determines residual atmosphere depth at zenith and clock errors
DFTPL	plots DFT of a UV data set at arbitrary point versus time
DIST	gives a distance
DO3COLOR	Controls whether full 3-color graphics are used in a plot
DOALIGN	specifies how two or more images are aligned in computations
DOBLANK	controls handling of blanking
DOCENTER	selects something related to centering
DOCIRCLE	select a "circular" display (i.e. trace coordinates, ...)
DOCOLOR	specifies whether coloring is done
DOCONT	selects a display of contour lines
DOCRT	selects printer display or CRT display (giving width)
DODARK	specifies whether "dark" vectors are plotted dark or light
DODATE	specifies if the dat and time are to be used in some way
DOEBAR	Controls display of estimates of the uncertainty in the data
DOFRACT	Tells whether to compute a fraction or ratio
DOGREY	selects a display of a grey-scale image
DOHIST	selects a histogram display or controls writing of history file
DOHMS	selects sexagesimal (hours-mins-secs) display format
DOMODEL	selects display of model function
DOPLOT	Controls plotting of something
DOPRINT	selects printer display or CRT display (giving width)
DOPROFIL	selects display of average across rows, columns, planes of an image'
DORESID	selects display of differences between model and data
DOSLICE	selects display of slice data
DOVECT	selects display of polarization vectors
DOWEDGE	selects display of intensity step wedge
EDROP	number of points/iterations to be omitted from end of process
ELFIT	Plots/fits selected contents of SN, TY, SY, PC or CL files
ELINT	Determines and removes gain dependence on elevation
EXTAB	exports AIPS table data as tab-separated text
EXTLIST	lists detailed information about contents of extension files
FACTOR	scales some display or CLEANing process
FGPLT	Plots selected contents of FG table
FLAMLEVS	Sets RGBLEVS to fill LEVS with a red flame color scheme
FRPLT	Task to plot fringe rate spectra
FTARS	Fits 1-dimensional polarization spectrum to Q/U text file
FUNCTYPE	specifies type of intensity transfer function
GAL	Determine parameters from a velocity field
GCPLT	Plots gain curves from text files
GREYS	plots images as contours over multi-level grey
GSTAR	Task to read a Guide Star (UK) table and create an ST table.
HLPUFLAG	Edit uv-data on a grid UFLAG - run-time help
HLPWIPER	edit uv data from UVPLT-like plot WIPER - run-time help
ICUT	specifies a cutoff level in units of the image
IMEAN	displays the mean & extrema and plots histogram of an image
IMRMS	Plot IMEAN rms answers

IMVIM	plots one image's values against another's
IRING	integrates intensity / flux in rings / ellipses
ISPEC	Plots and prints spectrum of region of a cube
KNTR	make a contour/grey plot file from an image w multiple panels
LABEL	selects a type of extra labeling for a plot
LAYER	Task to create an RGB image from multiple images
LEVS	list of multiples of the basic level to be contoured
LPEN	specifies the "pen width" code # =¿ width of plotted lines
LTYPE	specifies the type and degree of axis labels on plots
LWPLA	translates plot file(s) to a PostScript printer or file
MARSP	Check planetary polarization angles, plot histogram of differences
MF2ST	Task to generate an ST ext. file from Model Fit ext. file
NPLOTS	gives number of plots per page or per job
NX	General adverb referring to a number of things in the X direction
NY	General adverb referring to a number of things in the Y direction
OBPLT	Plot columns of an OB table.
OFMFILE	specifies the name of a text file containing OFM values
OKCOLORS	Sets adverb PLCOLORS to use less ink (white background)
PCFIT	Finds delays and phases using a pulse-cal (PC) table
PCHIS	Generates a histogram plot file from text input, e.g. from PCRMS
PCNTR	Generate plot file with contours plus polarization vectors
PCPLT	Plots pulse-cal tables in 2 dimensions as function of time
PCUT	Cutoff in polarized intensity
PDPLT	Plots selected contents of PD tables
PFPL1	Paraform Task to generate a plot file: (does grey scale)
PFPL2	Paraform Task to generate a plot file: (slice intensity)
PFPL3	Paraform Task to generate a plot file: (does histogram)
PLCOLORS	specifies the colors to be used
PLCUB	Task to plot intensity vs x panels on grid of y,z pixels
PLEV	Percentage of peak to use for contour levels
PLGET	gets the adverbs used to make a particular plot file
PLOTC	Plots color schems used by 3-color plot tasks
PLOTR	Basic task to generate a plot file from text input
PLRFI	Plots spectral statistics from output of VBRFI & VLBRF
PLROW	Plot intensity of a series of rows with an offset.
PLVER	specifies the version number of a PL extension file
POL3COL	Controls use of full 3-color graphics for polarization lines
POLPLOT	specifies the desired polarization ratio before plotting.
POSTSCRIP	General comments about AIPS use of PostScript incl macros
PRINTER	Verb to set or show the printer(s) used
PROFL	Generates plot file for a profile display.
PRPLT	Plots profiles of average image values on up to 3 axes.
PRTAB	prints any table-format extension file
PRTAN	prints the contents of the ANtenna extension file
PRTIM	prints image intensities from an MA catalog entry
PVPLOT	selects a display of a polarization vector in a box in a corner
RAINLEVS	Sets RGBLEVS to fill LEVS with a rainbow color scheme
RGBLEVS	colors to be applied to the contour levels
RIRMS	computes rms of real/imag of a selected subset of a uv data set
RM2PL	Plots spectrum of a pixel with RMFIT fit
RMFIT	Fits 1-dimensional polarization spectrum to Q/U cube
RSPEC	Plots and prints spectrum of rms of a cube
SCLIM	operates on an image with a choice of mathematical functions
SL2PL	Task to convert a Slice File to a Plot File
SLPRT	Task to print a Slice File
SNBLP	Plots selected contents of SN or CL files on a baseline basis
SNFIT	Fits parabola to SN amplitudes and plots result
SNIFS	Plots selected contents of SN, TY, SY, PC or CL files by IF
SNPLT	Plots selected contents of SN, SY, TY, PC or CL files versus time
SOUSP	fits source spectral index from SU table or adverbs
SPRMS	Plots spectral statistics of a selected subset of a uv data set

STARS	Task to generate an ST ext. file with star positions
STEPLEVS	Sets RGBLEVS to fill LEVS with a repeated sequence of colors
STFND	Task to find stars in an image and generate an ST table.
STVERS	star display table version number
SYMBOL	General adverb, probably defines a plotting symbol type
SYVSN	Plots selected SY/TY data with SN solution values versus time
TADIF	prints differences between two table-format extension files
TAPLT	Plots data from a Table extension file
TARPL	Plot input and output text files of the TARS task
TEKSERVER	Information about the X-11 Tektronix emulation server
TEKSRV	Information about the X-11 Tektronix emulation server
TEPLT	Plots selected contents of TE (TECOR) parameter tables
TKAMODEL	Verb to add slice model display directly on TEK
TKARESID	Verb to add slice model residuals directly on TEK
TKASLICE	Verb to add a slice display on TEK from slice file
TKBOX	Procedure to set a Clean box with the TK cursor
TKERASE	Erase the graphics screen or window
TKGUESS	Verb to display slice model guess directly on TEK
TKMODEL	Verb to display slice model directly on TEK
TKNBOXS	Procedure to set Clean boxes 1 - n with the TK cursor
TKPL	Task to send a plot file to the TEK
TKPOS	Read a position from the graphics screen or window
TKRESID	Verb to display slice model residuals directly on TEK
TKSLICE	Verb to display slice file directly on TEK
TKWIN	Procedure to set BLC and TRC with Graphics cursor
TVACOMPS	Verb to add slice model components directly on TV graphics
TVAGUESS	Verb to re-plot slice model guess directly on TV graphics
TVAMODEL	Verb to add slice model display directly on TV graphics
TVARESID	Verb to add slice model residuals directly on TV graphics
TVASLICE	Verb to add a slice display on TV graphics from slice file
TVCOLORS	Sets adverb PLCOLORS to match the TV (DOTV=1) usage
TVCOMPS	Verb to display slice model components directly on TV graphics
TVGUESS	Verb to display slice model guess directly on TV graphics
TVMODEL	Verb to display slice model directly on TV graphics
TVPL	Display a plot file on the TV
TVRESID	Verb to display slice model residuals directly on TV graphics
TVSLICE	Verb to display slice file directly on TV
TVSTAR	Verb to plot star positions on top of a TV image
TXPL	Displays a plot (PL) file on a terminal or line printer
UFLAG	Plots and edits data using a uv-plane grid and the TV
UVHGM	Plots statistics of uv data files as histogram.
UVHOL	Prints holography data from a UV data base with calibration
UVPLT	Plots data from a UV data base in multiple ways
UVPRM	measures parameters from a UV data base
UVRMS	computes statistics of a selected subset of a uv data set
VBRFI	Plots spectral statistics of VLBA autocorrelation data for RFI
VLBACRPL	Plots crosscorrelations
VLBASNPL	Plots selected contents of SN or CL files
VLBRF	Plots spectral statistics of VLBA autocorrelation data for RFI
VPLOT	Plots uv data and model from CC file
WETHR	Plots selected contents of WX tables, flags data based on WX
WIPER	plots and edits data from a UV data base using the TV
XAXIS	Which parameter is plotted on the horizontal axis.
XBASL	Fits and subtracts nth-order baselines from cube (x axis)
XG2PL	Plots spectrum of a pixel with XGAUS/AGAUS and ZEMAN/ZAMAN fits
XG2XG	Changes number of components in an XG or ZE table creating new one
XGAUS	Fits 1-dimensional Gaussians to images: restartable
XPLOT	Plots image rows one at a time on graphics, TV, or plot file
ZAMAN	Fits 1-dimensional Zeeman model to absorption-line data
ZEMAN	Fits 1-dimensional Zeeman model to data

13.26 POLARIZA

AFARS	Is used after FARS to determine Position and Value of the maximum
ALBUS	Calculate ionospheric delay and Faraday rotation using ALBUS
AVERSION	Specify a version of the ALBUS container
BANDPOL	specifies polarizations of individual IFs
BDEPO	computes depolarization due to rotation measure gradients
BLPCL	Do divide by I,Q,U models for circular polarization BLCHN
BLPOLCAL	Do BLCHN for I, Q, U, V models
COMB	combines two images by a variety of mathematical methods
CXPOLN	Procedure to make complex polarization images and beam
DFTQU	computes DFT of a UV data set at arbitrary point in Q and U polariz.
DOFARS	Procedure to aid in Faraday rotation synthesis using the FARS task
DOPOL	selects application of any polarization calibration
DT2PD	Write PD table from a PD table text file
FARAD	add ionospheric Faraday rotation to CL table
FARS	Faraday rotation synthesis based on the brightness vs wavelength
FTARS	Fits 1-dimensional polarization spectrum to Q/U text file
HLPRMFIT	Polarization fitting task RMFIT - run-time help
HLPZAMAN	Fits 1-dimensional Zeeman model to absorption data - run-time help
HLPZEMAN	Fits 1-dimensional Zeeman model to data - run-time help
LPCAL	Determines instrumental polarization for UV data
MAPBM	Map VLA beam polarization
MARSP	Check planetary polarization angles, plot histogram of differences
MEDI	combines four images by a variety of mathematical methods
MFIMG	Image the Earth magnetic field model or TEC data used by TECOR
MODAB	Makes simple absorption/emission spectral-line image in I/V
MODIM	adds images of model objects to image cubes in IQU polarization
MODSP	adds images of model objects to image cubes in I/V polarization
MTARS	Compute model inputs to TARS
PCAL	Determines instrumental polarization for UV data
PCNTR	Generate plot file with contours plus polarization vectors
PCUT	Cutoff in polarized intensity
PDEDT	Interactive TV task to edit polarization D-term (PD) tables
PDPLT	Plots selected contents of PD tables
PDVER	specifies the version of the spetral polarization table to use
PMODEL	Polarization model parameters
POLANGLE	Intrinsic polarization angles for up to 30 sources
POLCO	Task to correct polarization maps for Ricean bias
POLPLOT	specifies the desired polarization ratio before plotting.
PPAPP	Applies Right minus Left phase difference to a 2nd data set
PROGRAM	Specify a program to be used
QUFIX	determines Right minus Left phase difference, corrects cal files
QUOUT	writes text file of Q, U versus frequency to be used by RLDIF
QUXTR	extracts text files from Q,U cubes for input to TARS
RFARS	Correct Q/U cubes for Faraday rotation synthesis results
RIFRM	Task to modify a CL and TE table with IFRM values from ALBUS files
RLCAL	Determines instrumental right-left phase versus time (a self-cal)
RLCOR	corrects a data set for R-L phase differences
RLDIF	determines Right minus Left phase difference, corrects cal files
RM	Task to calculate rotation measure and magnetic field
RM2PL	Plots spectrum of a pixel with RMFIT fit
RMFIT	Fits 1-dimensional polarization spectrum to Q/U cube
ROTMEAS	Tells the rotation measure of a source
SWPOL	Swap polarizations in a UV data base
TARPL	Plot input and output text files of the TARS task
TARS	Simulation of Faraday rotation synthesis (mainly task FARS)

TRUEP	determines true antenna polarization from special data sets
VH2RL	Converts from linear to circular polarization and vice versa
VHCAL	Do divide by I,Q,U models for linear polarization data
VHCALIB	Do CALIB for linear polarization with I, Q, U models
VHDIF	find/apply V minus H linear polarization phase difference
VLABP	VLA antenna beam polarization correction for snapshot images
XG2PL	Plots spectrum of a pixel with XGAUS/AGAUS and ZEMAN/ZAMAN fits
XG2XG	Changes number of components in an XG or ZE table creating new one
XYDIF	find/apply X minus Y linear polarization phase difference
ZAMAN	Fits 1-dimensional Zeeman model to absorption-line data
ZEMAN	Fits 1-dimensional Zeeman model to data

13.27 POPS

ABOUT	displays lists and information on tasks, verbs, adverbs
ABS	returns absolute value of argument
ACOS	Returns arc cosine of argument (half-circle)
APROPOS	displays all help 1-line summaries containing specified words
ARRAY	Declares POPS symbol name and dimensions
ARRAY1	General scratch array adverb
ARRAY2	General scratch array adverb
ARRAY3	General scratch array adverb
ASIN	Returns arc sine of argument (half-circle)
ATAN	Returns arc tangent of argument (half-circle)
ATAN2	Returns arc tangent of two arguments (full circle)
BB	spare adverb for use in procedures
BY	gives increment to use in FOR loops in POPS language
CATNO	Specifies AIPS catalog slot number range
CEIL	returns smallest integer greater than or equal the argument
CHAR	converts number to character string
CLRTEMP	clears the temporary literal area during a procedure
COMPRESS	recovers unused POPS address space and new symbols
CORE	displays the used and total space used by parts of POPS table
COS	returns cosine of the argument in degrees
DEBUG	turns on/off the POPS-language's debug messages
DEFAULT	Verb-like sets adverbs for a task or verb to initial values
DELAY	Verb to pause AIPS for DETIME seconds
DELTAX	Increment or size in X direction
DELTAY	Increment or size in Y direction
DENUMB	a scalar decimal number
DOOSRO	calibrating amplitude and phase, and imaging VLA data
DOVLAMP	Produces amp calibration file for phased-VLA VLBI data
DOWNCASE	Convert a string to all lower case letters
DPARM	General numeric array adverb used many places
DUMP	displays portions of the POPS symbol table in all formats
EDIT	enter edit-a-procedure mode in the POPS language
EHEX	converts decimal to extended hex
EHNUMB	an extended hexadecimal "number"
ELSE	starts POPS code done if an IF condition is false (IF-THEN..)
END	marks end of block (FOR, WHILE, IF) of POPS code
ENDEDIT	terminates procedure edit mode of POPS input
ERASE	removes one or more lines from a POPS procedure
ERROR	was there an error
EVLA	puts the list of eVLA antennas in the current file on stack
EXIT	ends an AIPS batch or interactive session
EXP	returns the exponential of the argument
EXPLAIN	displays help + extended information describing a task/symbol
FINISH	terminates the entry and compilation of a procedure

FLOOR	returns largest integer ¡= argument
FOR	starts an iterative sequence of operations in POPS language
GET	restores previously SAVEd full POPS environment
GETDATE	Convert the current date and time to a string
GETPOPSN	Verb to return the pops number on the stack
GG	spare scalar adverb for use in procedures
GRANDOM	Finds a random number with mean 0 and rms 1
HELP	displays information on tasks, verbs, adverbs
HSA	puts the list of HSA antennas in the current file on stack
I	spare scalar adverb for use in procedures
IF	causes conditional execution of a set of POPS statements
INP	displays adverb values for task, verb, or proc - quick form
INPUTS	displays adverb values for task, verb, or proc - to msg file
ISBATCH	declares current AIPS to be, or not to be, batch-like
J	spare scalar adverb for use in procedures
KK	Spare adverb for use in procedures
KLEENEX	ends an AIPS interactive session wiping the slate klean
LENGTH	returns length of string to last non-blank character
LIST	displays the source code text for a POPS procedure
LN	returns the natural logarithm of the argument
LOG	returns the base-10 logarithm of the argument
MAX	returns the maximum of its two arguments
MIN	returns the minimum of its two arguments
MOD	returns remainder after division of 1st argument by 2nd
MODIFY	modifies the text of a line of a procedure and recompiles
MODULUS	returns square root of sum of squares of its two arguments
NOADVERB	Information about the lack of a defined adverb or verb
NUMTELL	selects POPS number of task which is the target of a TELL or ABORT
NX	General adverb referring to a number of things in the X direction
NY	General adverb referring to a number of things in the Y direction
OUTPUTS	displays adverb values returned from task, verb, or proc
PARALLEL	Verb to set or show degree of parallelism
PASSWORD	Verb to change the current password for the login user
PCAT	Verb to list entries in the user's catalog (no log file).
PIPEAIPS	calibrating amplitude and phase, and imaging VLA data
POPSDAT	lists all POPS symbols to create them for MEmory files
POPSYM	Describes the symbols used in POPS
PRINT	Print the value of an expression
PRINTER	Verb to set or show the printer(s) used
PROC	Define a POPS procedure using procedure editor.
PROCEDUR	Define a POPS procedure using procedure editor
PSEUDOVB	Declares a name to be a symbol of type pseudoverb
QINP	displays adverb values for task, verb, or proc - restart form
RANDOM	Compute a random number from 0 to 1
READ	Read a value from the users terminal
REHEX	converts extended hex string to decimal
RENAME	Rename a file (UV or Image)
RESTART	Verb to trim the message log file and restart AIPS
RESTORE	Read POPS memory file from a common area.
RETURN	Exit a procedure allowing a higher level proc to continue.
RUN	Pseudoverb to read an external RUN files into AIPS.
SAVDEST	Verb to destroy all save files of a user.
SAVE	Pseudoverb to save full POPS environment in named file
SCALAR	Declares a variable to be a scalar in a procedure
SCANLENG	specify length of "scan"
SCRATCH	delete a procedure from the symbol table.
SETDEBUG	Verb to set the debug print and execution level

SG2RUN	Verb copies the K area to a text file suitable for RUN
SGDESTR	Verb-like to destroy named POPS environment save file
SGINDEX	Verb lists SAVE areas by name and time of last SAVE.
SGUGET	Gets another user's SAVE/GET file.
SGUINDEX	Verb lists SAVE areas by name and time of last SAVE another user
SIN	Compute the sine of a value
SLOT	Specifies AIPS catalog slot number
SPY	Verb to determine the execution status of all AIPS tasks
SQRT	Square root function
STORE	Store current POPS environment
STQUEUE	Verb to list pending TELL operations
STRING	Declare a symbol to be a string variable in POPS
SUBMIT	Verb which submits a batch work file to the job queue
SUBSTR	Function verb to specify a portion of a STRING variable
T1VERB	Temporary verb for testing (also T2VERB...T9VERB)
TAN	Tangent function
TAPES	Verb to show the TAPES(s) available
TASK	Name of a task
TGET	Verb-like gets adverbs from last GO of a task
TGINDEX	Verb lists those tasks for which TGET will work.
TGUINDEX	Verb lists those tasks for which TUGET will work another user
THEDATE	contains the date and time in a string form
THEN	Specified the action if an IF test is true
TO	Specifies upper limit of a FOR loop
TPUT	Verb-like puts adverbs from a task in file for TGETs
TT	spare scalar adverb for use in procedures
TUGET	Verb-like gets adverbs from last GO of a task from another user #
TYPE	Type the value of an expression
UPCASE	Convert a string to all upper case letters
USAVE	Pseudoverb to save full POPS environment in named file
VALUE	Convert a string to a numeric value
VERB	Declares a name to be a symbol of type verb
VERSION	Specify AIPS version or local task area
VGET	Verb-like gets adverbs from version task parameter save area
VGINDEX	Verb lists those tasks for which VGET will work.
VLA	puts the list of VLA antennas in the current file on stack
VLARUN	Calibrating amplitude and phase, and imaging VLA data
VLBA	puts the list of VLBA antennas in the current file on stack
VLBAMPHC	applies VLBARUN calibration to additional phase stopping centers
VLBAPIPE	Applies amplitude and phase calibration procs to VLBA data
VLBARUN	applies amplitude and phase calibration procs to VLBA data
VNUMBER	Specifies the task parameter (VGET/VPUT) save area
VPUT	Verb-like puts adverbs from a task in files for VGETs
WAITTASK	halt AIPS until specified task is finished
WHILE	Start a conditional statement
X	spare scalar adverb for use in procedures
XHELP	Accesses hypertext help system
Y	spare scalar adverb for use in procedures

13.28 PROCEDUR

ANTNUM	Returns number of a named antenna
BASFIT	fits antenna locations from SN-table data
BLPOLCAL	Do BLCHN for I, Q, U, V models
BOX2CC	Converts CLBOX in pixels to CCBOX in arc seconds
BREAK	procedure to TELL FILLM to break all current uv files, start new
CALMODEL	Applies SPLIT & UVSUB to make a model data set matching a calibrator
CIRCLEVS	Sets RGBLEVS to fill LEVS with a circular color scheme
COTVLOD	Proc to load an image into a TV channel about a coordinate
CRSFRING	Procedure to calibrate cross pol. delay and phase offsets
CXIMAGR	Grid UV data into a complex image, Fourier transform, Clean
CXPOLN	Procedure to make complex polarization images and beam
DEFCOLOR	Sets adverb PLCOLORS to match s default XAS TV
DOFARS	Procedure to aid in Faraday rotation synthesis using the FARS task
FEW	procedure to TELL FILLM to append incoming data to existing uv files
FITDISK	writes images / uv data w extensions to disk in FITS format
FLAMLEVS	Sets RGBLEVS to fill LEVS with a red flame color scheme
FXALIAS	least squares fit aliasing function and remove
FXAVG	Procedure to enable VLBA delay de-correlation corrections
GRANDOM	Finds a random number with mean 0 and rms 1
IMFRING	large image delay fitting with IM2CC and OOFRING
IMSCAL	large image self-cal with IM2CC and OOCAL
LINIMAGE	Build image cube from multi-IF data set
MANY	procedure to TELL FILLM to start new uv files on each scan
MAPPR	Simplified access to IMAGR
MAXTAB	Returns maximum version number of named table
MERGECAL	Procedure to merge calibration records after concatenation
OFFHUINT	Proc which restores TV functions to normal after TVHUEINT
OKCOLORS	Sets adverb PLCOLORS to use less ink (white background)
OOCAL	determines antenna complex gain with frequency-dependent models
OOFRING	Fringe fit data to determine calibration with spectral index options
PEELR	calibrates interfering sources in multi-facet imges
QUIT	procedure to TELL FILLM to stop at the end of the current scan
RAINLEVS	Sets RGBLEVS to fill LEVS with a rainbow color scheme
READISK	Reads images / uv data w extensions in FITS format into AIPS
RUNWAIT	Runs a task and waits for it to finish
SCANTIME	Returns time range for a given scan number
SETXWIN	Procedure to set BLC and TRC with TV cursor
STEPLEVS	Sets RGBLEVS to fill LEVS with a repeated sequence of colors
STOP	procedure to TELL FILLM to break all current uv files and stop
STUFFR	averages together data sets in hour angle
TDEPEND	Time-dependent imaging procedure sequence
TDSCANS	Time-dependent imaging procedure sequence: find intervals
TDSSCAN	Time-dependent imaging procedure sequence: find intervals
TDSTEP3	Time-dependent imaging procedure ”step 3”
TDSTEP5	Time-dependent imaging procedure sequence: later steps
TELFLM	procedure to TELL real-time FILLM a new APARM(1) value
TKBOX	Procedure to set a Clean box with the TK cursor
TKNBOXS	Procedure to set Clean boxes 1 - n with the TK cursor
TKWIN	Procedure to set BLC and TRC with Graphics cursor
TV3LOD	Procedure to load an RGB image into 3 TV channels and display it
TVALL	Procedure loads image to TV, shows labeled wedge, enhances
TVCOLORS	Sets adverb PLCOLORS to match the TV (DOTV=1) usage
TVDIST	determines spherical distance between two pixels on TV screen
TVFLUX	displays coordinates and values selected with the TV cursor

TVMAXFIT	displays fit pixel positions and intensity at maxima on TV
TVRESET	Reset the TV without erasing the image planes
VHCALIB	Do CALIB for linear polarization with I, Q, U models
VLACALIB	Runs CALIB and LISTR for VLA observation
VLACLCAL	Runs CLCAL and prints the results with LISTR
VLALIST	Runs LISTR for VLA observation
VLARESET	Reset calibration tables to a virginal state
VLASUMM	Plots selected contents of SN or CL files
VLATECR	Calculate ionospheric delay and Faraday rotation corrections
VLBAAMP	applies a-priori amplitude corrections to VLBA data
VLBABPSS	computes spectral bandpass correction table
VLBACALA	applies a-priori amplitude corrections to VLBA data
VLBACCOR	applies a-priori amplitude corrections to VLBA data
VLBACRPL	Plots crosscorrelations
VLBAEOPS	Corrects Earth orientation parameters
VLBAFIX	Procedure that fixes VLBA data, if necessary
VLBAFPOL	Checks and corrects polarization labels for VLBA data
VLBAFQS	Copies different FQIDS to separate files
VLBAFRGP	Fringe fit phase referenced data and apply calibration
VLBAFRNG	Fringe fit data and apply calibration
VLBAKRGP	Fringe fit phase referenced data and apply calibration
VLBAKRNG	Fringe fit data and apply calibration
VLBALOAD	Loads VLBA data
VLBAMCAL	Merges redundant calibration data
VLBAMPCL	Calculates and applies manual instrumental phase calibration
VLBAPANG	Corrects for parallactic angle
VLBAPCOR	Calculates and applies instrumental phase calibration
VLBASNPL	Plots selected contents of SN or CL files
VLBASRT	Sorts VLBA data, if necessary
VLBASUBS	looks for subarrays in VLBA data
VLBASUMM	Prints a summary of a VLBI experiment
VLBATECR	Calculate ionospheric delay and Faraday rotation corrections
WRTDISK	writes images / uv data w extensions to disk in FITS format

13.29 PSEUDOVE

```
Type:  General type of POPS symbol
Use:   Pseudo-verbs are magic symbols which cause FORTRAN routines to
       carry out specific actions.  Unlike verbs, pseudo-verbs are
       executed as soon as they are encountered by the compiler even
       in compile mode.  In general, the FORTRAN routines which are
       invoked will parse the remainder of the input line under
       special, non-standard rules.  Any normal code typed on the line
       ahead of the pseudo-verb will not be executed.
Grammar:  See the HELP listings for the specific pseudoverb.
Examples:    HELP  HELP
             ARRAY   JUNK(4, -7 TO 9)
             PROC DUMMY (I,J)
             LIST DUMMY
             DEBUG TRUE
             INPUTS MLOAD
************************************************************
```

ABORTASK	stops a running task
ARRAY	Declares POPS symbol name and dimensions
COMPRESS	recovers unused POPS address space and new symbols
CORE	displays the used and total space used by parts of POPS table
DEBUG	turns on/off the POPS-language's debug messages
EDIT	enter edit-a-procedure mode in the POPS language
ELSE	starts POPS code done if an IF condition is false (IF-THEN..)
ENDBATCH	terminates input to batch work file
ENDEDIT	terminates procedure edit mode of POPS input
ERASE	removes one or more lines from a POPS procedure
FINISH	terminates the entry and compilation of a procedure
GET	restores previously SAVEd full POPS environment
IF	causes conditional execution of a set of POPS statements
ISBATCH	declares current AIPS to be, or not to be, batch-like
LIST	displays the source code text for a POPS procedure
MODIFY	modifies the text of a line of a procedure and recompiles
MSGKILL	turns on/off the recording of messages in the message file
TELL	Send parameters to tasks that know to read them on the fly
WHILE	Start a conditional statement

13.30 RUN

ANCHECK	Checks By sign in Antenna files
CHREFANT	Name of reference antenna
DOOSRO	calibrating amplitude and phase, and imaging VLA data
DOVLAMP	Produces amp calibration file for phased-VLA VLBI data
ELEVLIM	Lower limit on elevation
EOPSFILE	Name of text file containing EOP information
OBITVERS	Python version to use with BDF2AIPS and OBIT imaging verbs
PIPEAIPS	calibrating amplitude and phase, and imaging VLA data
SELFCAL	Requests self-calibration of calibration sources
SMRATES	Requests smoothing and clipping of rates after FRING in pipeline
TECRFILE	Name of file containg TEC data for use in TECOR
TECRTYPE	Type of IONEX file to download in VLA and VLBA TECR procedures
TYSMOLEV	Clip level to use in running TYSMO on a VLB TY table
VLAPROCS	Procedures to simplify the reduction of VLBA data
VLARUN	Calibrating amplitude and phase, and imaging VLA data
VLBAARCH	Procedure to archive VLBA correlator data
VLBAMPHC	applies VLBARUN calibration to additional phase stopping centers
VLBAPIPE	Applies amplitude and phase calibration procs to VLBA data
VLBARFI	Calibrates VLBA autocorrelations, writes out statistics
VLBARUN	Applies amplitude and phase calibration procs to VLBA data
VLBAUTIL	Procedures to simplify the reduction of VLBA data
VLBAWIDE	Calibrates all VLA wide-band autocorrelations with VLBARFI
VLBDDT	Verification tests using simulated data
WRTPROCS	Procedures to simplify the reduction of VLBA data
Y2K	verifies correctness and performance using standard problems
Y2KSAVE	verifies correctness and performance using standard problems

13.31 SINGLEDI

BSAVG	Task to do an FFT-weighted sum of beam-switched images
BSCLN	Hogbom Clean on beam-switched difference image
BSCOR	Combines two beam-switched images
BSFIX	Corrects the ra/dec offsets recorded by the 12m
BSGRD	Task to image beam-switched single-dish data
BSMOD	creates single-dish UV beam-switched data with model sources
BSROT	modifies SD beam-switch continuum data for error in throw
BSTST	Graphical display of solutions to frequency-switched data
CSCOR	applies specified corrections to CS tables
DATA2IN	specifies name of input FITS disk file
DATAIN	specifies name of input FITS disk file
DIFUV	Difference two matching uv data sets, write out or print statistics
INDXR	writes index file describing contents of UV data base
OTFBS	Translates on-the-fly continuum SDD format to AIPS UV file
OTFIN	Lists on-the-fly single-dish SDD format data files
OTFUV	Translates on-the-fly single-dish SDD format to AIPS UV file
PRTSD	prints contents of AIPS single-dish data sets
SDCAL	Task to apply single dish calibration
SDGRD	Task to select and image random-position single-dish data
SDIMG	Task to select and image random-position single-dish data
SDLSF	least squares fit to channels and subtracts from SD uv data
SDMOD	modifies single-dish UV data with model sources
SDTUV	Task to convert SD table files to UV like data.
SDVEL	shifts spectral-line single-dish data to a given velocity
VTEST	Measures velocity discrepancy across fields
WTSUM	Task to do a a sum of images weighted by other images

13.32 SPECTRAL

```
Almost all parts of AIPS are general enough to handle mutiple
dimensions of data including multiple frequency channels in the
uv domain and 3 or more dimensional "cubes" in the image domain.
```

ACFIT	Determine antenna gains from autocorrelations
AGAUS	Fits 1-dimensional Gaussians to absorption-line spectra
ALTDEF	Sets frequency vs velocity relationship into image header
ALTSWTCH	Switches between frequency and velocity in image header
AVSPC	Averages uv-data in the frequency domain
BASRM	Task to remove a spectral baseline from total power spectra
BCHAN	sets the beginning channel number
BDAPL	Applies a BD table to another data set
BLOAT	converts line data to greater number channels
BLPLT	Plots BLCAL tables in 2 dimensions as function of baseline and IF
BLSUM	sums images over irregular sub-images, displays spectra
BPASS	computes spectral bandpass correction table
BPASSPRM	Control adverb array for bandpass calibration
BPEPL	Plots bandpass tables overlaying multiple times like BPEDT
BPERR	Print and plot BPASS closure outputs
BPLOT	Plots bandpass tables in 2 dimensions as function of time
BPSMO	Smooths or interpolates bandpass tables to regular times
BPVER	specifies the version of the bandpass table to be applied
CHANNEL	sets the spectral channel number
CHANSEL	Array of start, stop, increment channel numbers to average

CHINC	the increment between selected channels
CPASS	computes polynomial spectral bandpass correction table
CUBIT	Model a galaxy's density and velocity distribution from full cube
CVEL	shifts spectral-line UV data to a given velocity
DCHANSEL	Array of start, stop, increment channel #S + IF to avoid
DOAPPLY	Flag to indicate whether an operation is applied to the data
ECHAN	define an end for a range of channel numbers
FIXAL	least squares fit aliasing function and remove
FLGIT	flags data based on the rms of the spectrum
FQUBE	collects n-dimensional images into n+1-dimensional FREQID image
FRMAP	Task to build a map using fringe rate spectra
FRPLT	Task to plot fringe rate spectra
FTARS	Fits 1-dimensional polarization spectrum to Q/U text file
FTFLG	interactive flagging of UV data in channel-time using the TV
FXALIAS	least squares fit aliasing function and remove
HLPFTFLG	Interactive time-channel visibility Editor - run-time help
HLPLISPX	Fits spectral index model to data in text file - run-time help
HLPPCFLG	Interactive time-channel PC table Editor PCFLG - run-time help
HLPSPFLG	Interactive time-channel visibility Editor SPFLG - run-time help
HLPTVHUI	Interactive intensity-hue-saturation display - run-time help
HLPTVRGB	Interactive red-green-blue display - run-time help
HLPTVSPX	Fits 1-dimensional spectral index model to data - run-time help
HLPZAMAN	Fits 1-dimensional Zeeman model to absorption data - run-time help
HLPZEMAN	Fits 1-dimensional Zeeman model to data - run-time help
ICHANSEL	Array of start, stop, increment channel #S + IF to average
IMLIN	Fits and removes continuum emission from cube
IRING	integrates intensity / flux in rings / ellipses
ISPEC	Plots and prints spectrum of region of a cube
MCUBE	collects n-dimensional images into n+1-dimensional image
MODAB	Makes simple absorption/emission spectral-line image in I/V
MORIF	Combines IFs or breaks spectral windows into multiple windows (IFs)
NCHAV	Number of channels averaged in an operation
ORDER	Adverb used usually to specify the order of polynomial fit
PCASS	Finds amplitude bandpass shape from pulse-cal table data
PCFIT	Finds delays and phases using a pulse-cal (PC) table
PCPLT	Plots pulse-cal tables in 2 dimensions as function of time
PCVEL	shifts spectral-line UV data to a given velocity: planet version
PDVER	specifies the version of the spetral polarization table to use
PLCUB	Task to plot intensity vs x panels on grid of y,z pixels
PLRFI	Plots spectral statistics from output of VBRFI & VLBRF
POLANGLE	Intrinsic polarization angles for up to 30 sources
POSSM	Task to plot total and cross-power spectra.
PRRFI	Prints excess spectral statistics from output of VBRFI & VLBRF
QUFIX	determines Right minus Left phase difference, corrects cal files
REFREQ	Specifies the frequency to be uased as the reference
REIFS	Breaks spectral windows into multiple spectral windows (IFs)
RM2PL	Plots spectrum of a pixel with RMFIT fit
RMFIT	Fits 1-dimensional polarization spectrum to Q/U cube
RSPEC	Plots and prints spectrum of rms of a cube
SDLSF	least squares fit to channels and subtracts from SD uv data
SDVEL	shifts spectral-line single-dish data to a given velocity
SERCH	Finds line signals in transposed data cube
SMOTH	Task to smooth a subimage from upto a 7-dim. image
SPECFLUX	Specifies the flux at the reference frequency
SPECINDX	Spectral index used to correct calibrations
SPECPARM	Spectral index per polarization per source
SPECR	Spectral regridding task for UV data
SPECTRAL	Flag to indicate whether an operation is spectral or continuum
SPECURVE	Spectral index survature used to correct calibrations

SPFLG	interactive flagging of UV data in channel-TB using the TV
SPMOD	Modify UV database by adding a model with spectral lines
SPRMS	Plots spectral statistics of a selected subset of a uv data set
SQASH	Task to collapse several planes in a cube into one plane or row
SYSVEL	Systemic velocity
UJOIN	modifies UV data converting IFs to spectral channels
UV2TB	Converts UV autocorrelation spectra to tables
UVBAS	averages several channels and subtracts from uv data.
UVDEC	Decrements the number of spectral channels, keeping every nth
UVFRE	Makes one data set have the spectral structure of another
UVGLU	Glues UV data frequency blocks back together
UVLIN	Fits and removes continuum visibility spectrum, also can flag
UVLSD	least squares fit to channels and divides the uv data.
UVLSF	least squares fit to channels and subtracts from uv data.
UVMLN	edits data based on the rms of the spectrum
UVMOD	Modify UV database by adding a model incl spectral index
VBGLU	Glues together data from multiple passes thru the VLBA corr.
VBRFI	Plots spectral statistics of VLBA autocorrelation data for RFI
VLBABPSS	computes spectral bandpass correction table
VLBRF	Plots spectral statistics of VLBA autocorrelation data for RFI
VTEST	Measures velocity discrepancy across fields
WTSUM	Task to do a a sum of images weighted by other images
XBASL	Fits and subtracts nth-order baselines from cube (x axis)
XG2PL	Plots spectrum of a pixel with XGAUS/AGAUS and ZEMAN/ZAMAN fits
XG2XG	Changes number of components in an XG or ZE table creating new one
XGAUS	Fits 1-dimensional Gaussians to images: restartable
XPLOT	Plots image rows one at a time on graphics, TV, or plot file
ZAMAN	Fits 1-dimensional Zeeman model to absorption-line data
ZEMAN	Fits 1-dimensional Zeeman model to data

13.33 TABLE

BSPRT	print BS tables
CLCOP	copy CL/SN file calibration between polarizations or IFs
CLINV	copy CL/SN file inverting the calibration
DOTABLE	selects use of table-format for data
EXTAB	exports AIPS table data as tab-separated text
HF2SV	convert HF tables from FRING/MBDLY to form used by Calc/Solve
HFPRT	write HF tables from CL2HF
HLPEDIBP	Interactive BP table uv-data editor BPEDT - run-time help
HLPEDICL	Interactive SN/CL table uv-data editor - run-time help
HLPEDIPC	Interactive PC table editor PCEDT - run-time help
HLPEDIPD	Interactive PD table editor PDEDT - run-time help
HLPEDISN	Interactive SN/CL table (not UV) editor - run-time help
HLPEDISS	Interactive SY table (not UV) editor - run-time help
HLPEDISY	Interactive SY table uv-data editor EDITA - run-time help
HLPEDITS	Interactive TY table (not UV) editor - run-time help
HLPEDITY	Interactive TY table uv-data editor EDITA - run-time help
MFPRT	prints MF tables in a format needed by modelling software
OBEDT	Task to flag data of orbiting antennas
OBTAB	Recalculate orbit parameters and other spacecraft info
OFLAG	uses on-line flag table information to write a flag table
PRTAB	prints any table-format extension file
PRTOF	prints on-line flag table information

SNCOP	Task to copy SN table averaging some input IFs
TABED	Task to edit tables
TACOP	task to copy tables, other extension files
TADIF	prints differences between two table-format extension files
TAFLG	Flags data in a Table extension file
TAMRG	Task to merge table rows under specified conditions
TAPLT	Plots data from a Table extension file
TAPPE	Task to append 2 tables and merge to output table
TASAV	Task to copy all extension tables to a dummy uv or map file
TASRT	Task to sort extension tables.
TBDIF	Compare entries in two tables
TBIN	Reads a text file AIPS table into AIPS
TBOUT	Writes an AIPS table into a text file for user editting.
TYCOP	copy TY or SY table calibration between IFs
VLBAMCAL	Merges redundant calibration data

13.34 TAPE

ATLOD	Reads ATCA data in RPFITS format into AIPS
AVEOT	Advances tape to end-of-information point
AVFILE	Moves tape forward or back to end-of-file marks
AVMAP	Advance tape by one image (IBM-CV = obsolete tape file)
AVTP	Positions tape to desired file
BAKLD	reads all files of a catalog entry from BAKTP tape
BAKTP	writes all files of a catalog entry to tape in host format
BLOCKING	specifies blocking factor to use on e.g. tape records
DENSITY	gives the desired tape density
DISMOUNT	disables a magnetic tape and dismounts it from the tape drive
DOEOF	selects end-of-file writing or reading until
DOEOT	selects tape positioning before operation: present or EOI
DONEWTAB	do we make new tables, use a new table format, etc.
DOTABLE	selects use of table-format for data
DOTWO	do we make two of something
FILLM	reads VLA on-line/archive format uv data tapes (post Jan 88)
FILLR	reads old VLA on-line-system tapes into AIPS
FIT2A	reads the fits input file and records it to the output ascii file
FITAB	writes images / uv data w extensions to tape in FITS format
FITLD	Reads FITS files to load images or UV (IDI or UVFITS) data to disk
FITTP	writes images / uv data w extensions to tape in FITS format
FORMAT	gives a format code number: e.g. FITS accuracy required
GSCAT	reads Fits Guide star catalog file
IMLOD	Reads tape or disk files to load images to disk
INTAPE	specifies the input tape drive number
MOUNT	makes a tape drive available to user's AIPS and tasks
NFILES	The number of files to skip, usually on a tape.
NPIECE	The number of pieces to make
OUTTAPE	The output tape drive number.
PRTTP	prints contents of tapes and disk files, all supported formats
QUANTIZE	Quantization level to use
REMHOST	gives the name of another computer which will provide service
REMTAPE	gives the number of another computer's tape device
REWIND	Verb to rewind a tape

TAPES	Verb to show the TAPES(s) available
TCOPY	Tape to tape copy with some disk FITS support
TPHEAD	Verb to list image header from FITS or IBM-CV tape
TPMON	Information about the TPMON "Daemon"
UVLOD	Read export or FITS data from a tape or disk
VLAMODE	VLA observing mode
VLAOBS	Observing program or part of observer's name

13.35 TASK

```
TASKS
Type:  General type of POPS symbol (not in symbol table)
Use:   Tasks are separate programs which may be started by
       AIPS and which receive their input parameters from
       AIPS.  In the interactive AIPS, tasks run
       asynchronously from AIPS.  In the batch AIPS,
       the language processor waits for each task to finish
       before starting another one.
Grammar:    TASK = 'name' ;   GO
       will cause the task whose name is assigned to the
       string adverb TASK to be started.  Note: the name
       should have no leading blanks and should be no longer
       than 5 characters.
Alternative grammar:    GO  name ;
       where name is the name of the task to be run.
Related adverbs:
   TASK       Task name
   DOWAIT     On "GO", wait for task completion before returning
                 to AIPS control
   VERSION    Version of task to be executed.
   QCREATE    Create files by fast or more reliable methods
Related verbs:
   GO         Initiate a shed task
   HELP       List information about a task
   INP        List adverb values for a task
   INPUTS     Same as INP but also written to MSG file
   SPY        Inquire which tasks are active
   WAITTASK   Suspend AIPS operation until a specific task
                 is complete
   ABORTASK   Kill a task immediately
   TGET       Get adverb values from last execution of TASK
   TPUT       Save adverb values without execution of TASK
   TGINDEX    List all TGET/SAVE files
************************************************************
```

ACCOR	Corrects cross amplitudes using auto correlation measurements
ACFIT	Determine antenna gains from autocorrelations
ACIMG	Makes image of autocorrelation data showing time vs frequency
ACLIP	edits suto-corr data for amplitudes, phases, and weights out of range
ACSCL	Corrects cross amplitudes using auto correlation measurements
ADDIF	Adds an IF axis to a uv data set
AFARS	Is used after FARS to determine Position and Value of the maximum
AFILE	sorts and edits MkIII correlator A-file.
AGAUS	Fits 1-dimensional Gaussians to absorption-line spectra
AHIST	Task to convert image intensities by adaptive histogram

AIPS	AIPS main program for interactive use
AIPSB	AIPS main program for executing batch jobs
AIPSC	AIPS main program for testing and queuing batch jobs
ALBUS	Calculate ionospheric delay and Faraday rotation using ALBUS
ALVAR	Plots the Allan Variance statistic of a UV data set
ALVPR	prints statistics on the Allan Variance of a UV data set
AMKAT	renumbers MeerKAT antennas so taht antenna number =¿ station ID
ANBPL	plots and prints uv data converted to antenna based values
ANCAL	Places antenna-based Tsys and gain corrections in CL table
ANTAB	Read amplitude calibration information into AIPS
APCAL	Apply TY and GC tables to generate an SN table
APCLN	Deconvolves images with CLEAN algorithm
APGPS	Apply GPS-derived ionospheric corrections
APGS	deconvolves image with Gerchberg-Saxton algorithm
APVC	Deconvolves images with van Cittert algorithm
ATLOD	Reads ATCA data in RPFITS format into AIPS
ATMCA	Determines delay/phase gradient from calibrator observations
AVER	Averages over time UV data sets in 'BT' order
AVGWT	Computes the average visibility weight of the included data
AVSPC	Averages uv-data in the frequency domain
AVTP	Positions tape to desired file
BAKLD	reads all files of a catalog entry from BAKTP tape
BAKTP	writes all files of a catalog entry to tape in host format
BASRM	Task to remove a spectral baseline from total power spectra
BATER	stand-alone program to prepare and submit batch jobs
BDAPL	Applies a BD table to another data set
BDEPO	computes depolarization due to rotation measure gradients
BLANK	blanks out selected, e.g. non-signal, portions of an image
BLAPP	applies baseline-based fringe solutions a la BLAPP
BLAVG	Average cross-polarized UV data over baselines.
BLCAL	Compute closure offset corrections
BLCHN	Compute closure offset corrections on a channel-by-channel basis
BLING	find residual rate and delay on individual baselines
BLOAT	converts line data to greater number channels
BLPCL	Do divide by I,Q,U models for circular polarization BLCHN
BLPLT	Plots BLCAL tables in 2 dimensions as function of baseline and IF
BLSUM	sums images over irregular sub-images, displays spectra
BLWUP	Blow up an image by any positive integer factor.
BOXES	Adds Clean boxes to BOXFILE around sources.
BPASS	computes spectral bandpass correction table
BPCOR	Correct BP table.
BPEDT	Interactive TV task to edit uv data based on BP tables
BPEPL	Plots bandpass tables overlaying multiple times like BPEDT
BPERR	Print and plot BPASS closure outputs
BPFLG	Flgas uv data based on bad BP values
BPLOT	Plots bandpass tables in 2 dimensions as function of time
BPPLT	Plots selected contents of bandpass (BP) files versus time
BPSMO	Smooths or interpolates bandpass tables to regular times
BPWAY	Determines channel-dependent relative weights
BPWGT	Calibrates data and scales weights by bandpass correction
BSAVG	Task to do an FFT-weighted sum of beam-switched images
BSCAN	seeks best scan to use for phase cal, fringe search, ..
BSCLN	Hogbom Clean on beam-switched difference image
BSCOR	Combines two beam-switched images
BSFIX	Corrects the ra/dec offsets recorded by the 12m
BSGEO	Beam-switched Az-El image to RA-Dec image translation
BSGRD	Task to image beam-switched single-dish data
BSMAP	Images weak sources with closure phases
BSMOD	creates single-dish UV beam-switched data with model sources

BSPRT	print BS tables
BSROT	modifies SD beam-switch continuum data for error in throw
BSTST	Graphical display of solutions to frequency-switched data
CALIB	determines antenna calibration: complex gain
CALRD	Reads calibrator source model-image FITS file
CALWR	writes calibrator source model images w CC files to FITS disk files
CANDY	user-definable (paraform) task to create an AIPS image
CAPLT	plots closure amplitude and model from CC file
CC2IM	Make model image from a CC file
CCEDT	Select CC components in BOXes and above mininum flux.
CCFND	prints the contents of a Clean Components extension file.
CCGAU	Converts point CLEAN components to Gaussians
CCMOD	generates clean components to fit specified source model
CCMRG	sums all clean components at the same pixel
CCNTR	Generate a contour plot file from an image with CC or MF plotted
CCRES	Removes or restores a CC file to a map with a gaussian beam.
CCSEL	Select signifigant CC components
CENTR	modifies UV data to center the reference channel
CHEBI	Make text file to plot Chebyshev polynomials
CHKFC	makes images of Clean boxes from Boxfile
CL2HF	Convert CL table to HF table
CLCAL	merges and smooths SN tables, applies them to CL tables
CLCOP	copy CL/SN file calibration between polarizations or IFs
CLCOR	applies user-selected corrections to the calibration CL table
CLINV	copy CL/SN file inverting the calibration
CLIP	edits data based on amplitudes, phases, and weights out of range
CLNUP	removes old MS, RL, SG, TG and ?? files from AIPS disks
CLOSE	plots closure phase or amplitude spectra averaged over closures
CLPLT	plots closure phase and model from CC file
CLSMO	smooths a calibration CL table
CLVLB	Corrects CL table gains for pointing offsets in VLBI data
CMPLX	Combines complex images by a variety of mathematical methods.
CNTR	generate a contour plot file or TV plot from an image
COHER	Baseline Phase coherence measurement
COMB	combines two images by a variety of mathematical methods
CONFI	Optimize array configuration by minimum side lobes
CONPL	Plots AIPS gridding convolution functions
CONVL	convolves an image with a gaussian or another image
CORER	calculates correlator statistics and flags bad ones
CORFQ	corrects uvw for incorrect observing frequency
CPASS	computes polynomial spectral bandpass correction table
CPYRT	replaces history with readme file, inserts copyright
CSCOR	applies specified corrections to CS tables
CUBIT	Model a galaxy's density and velocity distribution from full cube
CVEL	shifts spectral-line UV data to a given velocity
CXCLN	Complex Hogbom CLEAN
DAYFX	Fixes day number problems left by FILLM
DBAPP	appends one or more data sets to the output data set
DBCON	concatenates two UV data sets
DCONV	deconvolves a gaussian from an image
DECOR	Measures the decorrelation between channels and IF of uv data
DEFLG	edits data based on decorrelation over channels and time
DELZN	Determines residual atmosphere depth at zenith and clock errors
DENAN	modifies UV data flagging all NaNs
DESCM	copies a portion of a UV data set
DFCOR	Applies ATMO differential corrections to the calibration CL table
DFQID	modifies UV data changing the indicated FQIDs
DFTIM	Makes image of DFT at arbitrary point showing time vs frequency
DFTPL	plots DFT of a UV data set at arbitrary point versus time
DFTQU	computes DFT of a UV data set at arbitrary point in Q and U polariz.

DGAUS	Correct UV model for convolution by Clean beam
DIFRL	divides the RR data by LL data
DIFUV	Difference two matching uv data sets, write out or print statistics
DISKU	shows disk use by one or all users
DQUAL	Rearranges source list, dropping qualifiers
DRCHK	stand-alone program checks system setup files for consistency
DSKEW	Geometric interpolation correction for skew
DSORC	copies a data set elliminating some source numbers
DSTOK	Drops the cross-hand polarizations
DT2PD	Write PD table from a PD table text file
DTCHK	Task to check results of a test using simulated data.
DTSIM	Generate fake UV data
DTSUM	Task to provide a summary of the contents of a dataset
EDITA	Interactive TV task to edit uv data based on TY/SY/SN/CL tables
EDITR	Interactive baseline-oriented visibility editor using the TV
ELFIT	Plots/fits selected contents of SN, TY, SY, PC or CL files
ELINT	Determines and removes gain dependence on elevation
EVACL	Evaluates statistics in closure phase and amplitude
EVASN	Evaluates statistics in SN/CL tables
EVAUV	Subtracts & divides a model into UV data, does statistics on results
EXTAB	exports AIPS table data as tab-separated text
FACES	makes images of catalog sources for initial calibration
FARAD	add ionospheric Faraday rotation to CL table
FARS	Faraday rotation synthesis based on the brightness vs wavelength
FETCH	Reads an image from an external text file.
FFT	Takes Fourier Transform of an image or images
FGCNT	Counts samples comparing two flag tables or detailed count from ¡= 1
FGDIF	Compares affect of 2 FG tables
FGPLT	Plots selected contents of FG table
FGPRT	Prints counts of selected portions of an FG table.
FGSPW	Flags bad spectral windows
FGTAB	writes frequency-range flags to a text file to be read by UVFLG
FILIT	Interactive BOXFILE editing with facet images
FILLM	reads VLA on-line/archive format uv data tapes (post Jan 88)
FILLR	reads old VLA on-line-system tapes into AIPS
FINDR	Find normal values for a uv data set
FIT2A	reads the fits input file and records it to the output ascii file
FITAB	writes images / uv data w extensions to tape in FITS format
FITLD	Reads FITS files to load images or UV (IDI or UVFITS) data to disk
FITTP	writes images / uv data w extensions to tape in FITS format
FIXAL	least squares fit aliasing function and remove
FIXAN	fixes the contents of the ANtenna extension file
FIXBX	converts a BOXFILE to another for input to IMAGR
FIXFR	Modifies UV data set making all frequencies increase or decrease
FIXRL	correctes right vs left polarizations for a list of antennas
FIXWT	Modify weights to reflect amplitude scatter of data
FLAGR	Edit data based on internal RMS, amplitudes, weights
FLATN	Re-grid multiple fields into one image incl sensitivity
FLGIT	flags data based on the rms of the spectrum
FLOPM	reverses the spectral order of UV data, can fix VLA error
FQUBE	collects n-dimensional images into n+1-dimensional FREQID image
FRCAL	Faraday rotation self calibration task
FRING	fringe fit data to determine antenna calibration, delay, rate
FRMAP	Task to build a map using fringe rate spectra
FRPLT	Task to plot fringe rate spectra
FTARS	Fits 1-dimensional polarization spectrum to Q/U text file
FTFLG	interactive flagging of UV data in channel-time using the TV
FUDGE	modifies UV data with user's algorithm: paraform task
FXPOL	Corrects VLBA polarization assignments

FXTIM	fixes start date so all times are positive
FXVLA	Task to correct VLA data for on-line errors in special cases.
FXVLB	Builds a CQ table to enable VLBA correlator loss corrections
GAL	Determine parameters from a velocity field
GCPLT	Plots gain curves from text files
GETJY	determines calibrator flux densities
GLENS	models galaxy gravitational lens acting on 3 component source
GPSDL	Calculate ionospheric delay and Faraday rotation corrections
GREYS	plots images as contours over multi-level grey
GRIPR	standalone program to enter suggestions/complaints to AIPS
GSCAT	reads Fits Guide star catalog file
GSTAR	Task to read a Guide Star (UK) table and create an ST table.
HA2TI	Converts data processed by TI2HA (STUFFER) back to real times
HAFIX	Recomputes u,v,w when time is hour angle (UVdata is output of TI2HA)
HF2SV	convert HF tables from FRING/MBDLY to form used by Calc/Solve
HFPRT	write HF tables from CL2HF
HGEOM	interpolates image to different gridding and/or geometry
HISEQ	task to translate image by histogram equalization
HLPAGAUS	Interactive Gaussian absorption fitting task AGAUS - run-time help
HLPCLEAN	Cleaning tasks - run-time help
HLPEDIBP	Interactive BP table uv-data editor BPEDT - run-time help
HLPEDICL	Interactive SN/CL table uv-data editor - run-time help
HLPEDIPC	Interactive PC table editor PCEDT - run-time help
HLPEDIPD	Interactive PD table editor PDEDT - run-time help
HLPEDISN	Interactive SN/CL table (not UV) editor - run-time help
HLPEDISS	Interactive SY table (not UV) editor - run-time help
HLPEDISY	Interactive SY table uv-data editor EDITA - run-time help
HLPEDITS	Interactive TY table (not UV) editor - run-time help
HLPEDITY	Interactive TY table uv-data editor EDITA - run-time help
HLPEDIUV	Interactive uv-data editor EDITR - run-time help
HLPFILIT	Interactive Clean box file editing with image display - run-time help
HLPFTFLG	Interactive time-channel visibility Editor - run-time help
HLPIBLED	Interactive Baseline based visibility Editor - run-time help
HLPLISPX	Fits spectral index model to data in text file - run-time help
HLPPCFLG	Interactive time-channel PC table Editor PCFLG - run-time help
HLPPLAYR	OOP TV class demonstration task - run-time help
HLPRMFIT	Polarization fitting task RMFIT - run-time help
HLPSCIMG	Full-featured image plus self-cal loops, editing - run-time help
HLPSCMAP	Imaging plus self-cal and editing SCMAP - run-time help
HLPSPFLG	Interactive time-channel visibility Editor SPFLG - run-time help
HLPTVFLG	Interactive time-baseline visibility Editor TVFLG - run-time help
HLPTVHLD	Interactive image display with histogram equalization - run-time help
HLPTVHUI	Interactive intensity-hue-saturation display - run-time help
HLPTVIEW	Interactive display of cube in 3 transpositions - run-time help
HLPTVRGB	Interactive red-green-blue display - run-time help
HLPTVSAD	Find & fit Gaussians to an image with interaction - run-time help
HLPTVSPC	Interactive display of spectra from a cube - run-time help
HLPTVSPX	Fits 1-dimensional spectral index model to data - run-time help
HLPUFLAG	Edit uv-data on a grid UFLAG - run-time help
HLPWIPER	edit uv data from UVPLT-like plot WIPER - run-time help
HLPXGAUS	Interactive Gaussian fitting task XGAUS - run-time help
HLPZAMAN	Fits 1-dimensional Zeeman model to absorption data - run-time help
HLPZEMAN	Fits 1-dimensional Zeeman model to data - run-time help
HOCLN	Task to do a clean of holography images
HOLGR	Read & process holography visibility data to telescope images
HOLIS	Prints information about holography sampling
HOLOG	Read & process holography visibility data to telescope images
HUINT	make RGB image from images of intensity & hue, like TVHUEINT
IANTB	Writes out GC and TY data in ANTAB format

IBLED Interactive BaseLine based visibility EDitor
IM2CC Task to convert an image to multi-facet Clean Components
IM2TX Sums an image region into a 1-dimensional text file
IM2UV converts an image to a visibility data set
IMAGR Wide-field and/or wide-frequency Cleaning / imaging task.
IMCLP Clip an image to a specified range.
IMEAN displays the mean & extrema and plots histogram of an image
IMERG merges images of different spatial resolutions
IMFIT Fits Gaussians to portions of an image
IMFLT fits and removes a background intensity plane from an image
IMLHS converts images to luminosity/hue TV display
IMLIN Fits and removes continuum emission from cube
IMLOD Reads tape or disk files to load images to disk
IMMOD adds images of model objects to an image
IMRMS Plot IMEAN rms answers
IMTXT Write an image to an external text file.
IMVIM plots one image's values against another's
INDXR writes index file describing contents of UV data base
IRING integrates intensity / flux in rings / ellipses
ISPEC Plots and prints spectrum of region of a cube
JMFIT Fits Gaussians to portions of an image
KNTR make a contour/grey plot file from an image w multiple panels
KRING fringe fit data to determine antenna calibration, delay, rate
LAYER Task to create an RGB image from multiple images
LDGPS load GPS data from an ASCII file
LGEOM regrids images with rotation, shift using interpolation
LISPX Fits spectral indexes to a list of spectra, allows interactive edit
LISTR prints contents of UV data sets and assoc. calibration tables
LISUN prints contents of UV data sets with Sun angles
LOCIT fits antenna locations from SN-table data
LPCAL Determines instrumental polarization for UV data
LTESS makes mosaic images by linear combination
LWPLA translates plot file(s) to a PostScript printer or file
M3TAR translate Haystack MKIII VLBI format "A" TAR's into AIPS
MANDL creates an image of a subset of the Mandlebrot Set
MAPBM Map VLA beam polarization
MARSP Check planetary polarization angles, plot histogram of differences
MASKS Makes mask image of Clean boxes from boxfile
MATCH changes antenna, source, FQ numbers to match a data set
MATHS operates on an image with a choice of mathematical functions
MBDLY Fits multiband delays from IF phases, updates SN table
MCUBE collects n-dimensional images into n+1-dimensional image
MEDI combines four images by a variety of mathematical methods
MF2ST Task to generate an ST ext. file from Model Fit ext. file
MFIMG Image the Earth magnetic field model or TEC data used by TECOR
MFPRT prints MF tables in a format needed by modelling software
MK3IN translate Haystack MKIII VLBI format "A" tapes into AIPS
MK3TX extract text files from a MKIII VLBI archive tape
MOD3D Computes a 3D CC model from a set of facets
MODAB Makes simple absorption/emission spectral-line image in I/V
MODIM adds images of model objects to image cubes in IQU polarization
MODSP adds images of model objects to image cubes in I/V polarization
MODVF task to create a warped velocity field
MOMFT calculates images of moments of a sub-image
MOMNT calculates images of moments along x-axis (vel, freq, ch)
MORIF Combines IFs or breaks spectral windows into multiple windows (IFs)
MOVE Task to copy or move data from one user or disk to another
MSORT Sort a UV dataset into a specified order
MTARS Compute model inputs to TARS

MULIF	Change number of IFs in output
MULTI	Task to convert single-source to multi-source UV data
MWFLT	applies linear & non-linear filters to images
NANS	reads an image or a UV data set and looks for NaNs
NINER	Applies various 3x3 area operaters to an image.
NNLSQ	Non-Negative-Least-Squares decomposition of spectrum
NOBAT	Task to lock lower priority users out of the AP
NOIFS	makes all IFs into single spectrum
OBEDT	Task to flag data of orbiting antennas
OBPLT	Plot columns of an OB table.
OBTAB	Recalculate orbit parameters and other spacecraft info
OFLAG	uses on-line flag table information to write a flag table
OFMPL	Task to display and contrast system and user OFM color tables.
OFUDG	modifies UV data with user's algorithm no cal adverbs: paraform task
OGEOM	Simple image rotation, scaling, and translation
OHGEO	Geometric interpolation with correction for 3-D effects
OMFIT	Fits sources and, optionally, a self-cal model to uv data
OOSRT	Sort a UV dataset into a specified order
OOSUB	Subtracts/divides a model from/into a uv data base
OTFBS	Translates on-the-fly continuum SDD format to AIPS UV file
OTFIN	Lists on-the-fly single-dish SDD format data files
OTFUV	Translates on-the-fly single-dish SDD format to AIPS UV file
PADIM	Task to increase image size by padding with some value
PANEL	Convert HOLOG output to panel adjustment table
PASTE	Pastes a selected subimage of one image into another.
PATGN	Task to create a user specified test or primary-beam pattern
PBCOR	Task to apply or correct an image for a primary beam
PBEAM	Fits the analytic function to the measured values of the beam
PCABL	Copies the Cable cal values from PC table 1 to PC table n
PCAL	Determines instrumental polarization for UV data
PCASS	Finds amplitude bandpass shape from pulse-cal table data
PCAVG	Averages pulse-cal (PC) tables over time
PCCOR	Corrects phases using PCAL tones data from PC table
PCDIV	Scales pulse-cal amplitudes by Tsys or Psum
PCEDT	Interactive TV task to edit pulse-cal (PC) tables
PCFIT	Finds delays and phases using a pulse-cal (PC) table
PCFLG	interactive flagging of Pulse-cal data in channel-TB using the TV
PCHIS	Generates a histogram plot file from text input, e.g. from PCRMS
PCLOD	Reads ascii file containing pulse-cal info to PC table.
PCNTR	Generate plot file with contours plus polarization vectors
PCPLT	Plots pulse-cal tables in 2 dimensions as function of time
PCRMS	Finds statistics of a pulse-cal table; flags bad times and channels
PCVEL	shifts spectral-line UV data to a given velocity: planet version
PDEDT	Interactive TV task to edit polarization D-term (PD) tables
PDPLT	Plots selected contents of PD tables
PEEK	fits pointing model function to output from the VLA
PFPL1	Paraform Task to generate a plot file: (does grey scale)
PFPL2	Paraform Task to generate a plot file: (slice intensity)
PFPL3	Paraform Task to generate a plot file: (does histogram)
PGEOM	Task to transform an image into polar coordinates.
PHASE	Baseline Phase coherence measurement
PHNEG	Negates a UV datafile's visibility phase.
PHSRF	Perform phase-referencing within a spectral line database.
PLAYR	Task to load one or two images and play with them on the TV
PLCUB	Task to plot intensity vs x panels on grid of y,z pixels
PLOTC	Plots color schems used by 3-color plot tasks
PLOTR	Basic task to generate a plot file from text input
PLRFI	Plots spectral statistics from output of VBRFI & VLBRF
PLROW	Plot intensity of a series of rows with an offset.
POLCO	Task to correct polarization maps for Ricean bias

POSSM	Task to plot total and cross-power spectra.
PPAPP	Applies Right minus Left phase difference to a 2nd data set
PROFL	Generates plot file for a profile display.
PRPLT	Plots profiles of average image values on up to 3 axes.
PRRFI	Prints excess spectral statistics from output of VBRFI & VLBRF
PRTAB	prints any table-format extension file
PRTAC	prints contents and summaries of the accounting file
PRTAN	prints the contents of the ANtenna extension file
PRTCC	prints the contents of a Clean Components extension file.
PRTIM	prints image intensities from an MA catalog entry
PRTOF	prints on-line flag table information
PRTSD	prints contents of AIPS single-dish data sets
PRTSY	Task to print statistics from the SY table
PRTTP	prints contents of tapes and disk files, all supported formats
PRTUV	prints contents of a visibility (UV) data set
QBEAM	Plot holography data as contour image and outputs interpolated image
QUACK	Flags beginning or end portions of UV-data scans
QUFIX	determines Right minus Left phase difference, corrects cal files
QUOUT	writes text file of Q, U versus frequency to be used by RLDIF
QUXTR	extracts text files from Q,U cubes for input to TARS
REAMP	Scale visibility amplitudes by IF- or subarray-dependent factor
REFLG	Attempts to compress a flag table
REGRD	Regrids an image from one co-ordinate frame to another
REIFS	Breaks spectral windows into multiple spectral windows (IFs)
REMAG	Task to replace magic blanks with a user specified value
RESEQ	Renumber antennas
RESOU	Changes (renumbers) a list of source numbers in UV data and tables
REWAY	computes weights based in rms in spectra
REWGT	modifies UV data re-scaling the weights only
RFARS	Correct Q/U cubes for Faraday rotation synthesis results
RFI	Look for RFI in uv data
RFLAG	Flags data set based on time and freq rms in fringe visibilities
RGBMP	Task to create an RGB image from the 3rd dim of an image
RIFRM	Task to modify a CL and TE table with IFRM values from ALBUS files
RIRMS	computes rms of real/imag of a selected subset of a uv data set
RLCAL	Determines instrumental right-left phase versus time (a self-cal)
RLCOR	corrects a data set for R-L phase differences
RLDIF	determines Right minus Left phase difference, corrects cal files
RLDLY	fringe fit data to determine antenna R-L delay difference
RM	Task to calculate rotation measure and magnetic field
RM2PL	Plots spectrum of a pixel with RMFIT fit
RMFIT	Fits 1-dimensional polarization spectrum to Q/U cube
RMSD	Calculate rms for each pixel using data in a box around the pixel
RMTFC	Convolves rotation measure transfer function with slab models.
RSPEC	Plots and prints spectrum of rms of a cube
RSTOR	Restores a Clean component file to a map with a Gaussian beam.
RTIME	Task to test compute times
SABOX	create box file from source islands in facet images
SAD	Finds and fits Gaussians to portions of an image
SBCOR	Task to correct VLBA data for phase shift between USB & LSB
SCIMG	Full-featured imaging plus self-calibration loop with editing
SCLIM	operates on an image with a choice of mathematical functions
SCMAP	Imaging plus self-calibration loop with editing
SDCAL	Task to apply single dish calibration
SDCLN	deconvolves image by Clark and then "SDI" cleaning methods
SDGRD	Task to select and image random-position single-dish data
SDIMG	Task to select and image random-position single-dish data
SDLSF	least squares fit to channels and subtracts from SD uv data
SDMOD	modifies single-dish UV data with model sources
SDTUV	Task to convert SD table files to UV like data.
SDVEL	shifts spectral-line single-dish data to a given velocity

SERCH Finds line signals in transposed data cube
SETAN Reads an ANtenna file info from a text file
SETFC Makes a BOXFILE for input to IMAGR, recommends cell and image sizes.
SETJY Task to enter source info into source (SU) table.
SHADO Calculate the shadowing of antennas at the array
SHADW Generates the "shadowed" representation of an image
SHAHO Flags Holography data for shadowing
SHOUV displays uv data in various ways.
SKYVE Regrids a DSS image from one co-ordinate frame to another
SL2PL Task to convert a Slice File to a Plot File
SLCOL Task to collate slice data and models.
SLFIT Task to fit gaussians to slice data.
SLICE Task to make a slice file from an image
SLPRT Task to print a Slice File
SMOTH Task to smooth a subimage from upto a 7-dim. image
SNBLP Plots selected contents of SN or CL files on a baseline basis
SNCOP Task to copy SN table averaging some input IFs
SNCOR applies user-selected corrections to the calibration SN table
SNDUP copies and duplicates SN table from single pol file to dual pol
SNEDT Interactive SN/CL/TY/SY table editor using the TV
SNFIT Fits parabola to SN amplitudes and plots result
SNFLG Writes flagging info based on the contents of SN files
SNIFS Plots selected contents of SN, TY, SY, PC or CL files by IF
SNP2D Task to convert SN table single-channel phase to delay
SNPLT Plots selected contents of SN, SY, TY, PC or CL files versus time
SNREF Chooses best reference antenna to minimize R-L differences
SNRMS Print statistics for selected contents of SN, SY, TY, PC or CL files
SNSMO smooths and filters a calibration SN table
SOLCL adjust gains for solar data according to nominal sensitivity
SOUSP fits source spectral index from SU table or adverbs
SPCAL Determines instrumental polzn. for spec. line UV data
SPCOR Task to correct an image for a primary beam and spectral index
SPECR Spectral regridding task for UV data
SPFIX Makes cube from input to and output from SPIXR spectral index
SPFLG interactive flagging of UV data in channel-TB using the TV
SPIXR Fits spectral indexes to each row of an image incl curvature
SPLAT Applies calibration and splits or assemble selected sources.
SPLIT converts multi-source to single-source UV files w calibration
SPMOD Modify UV database by adding a model with spectral lines
SPRMS Plots spectral statistics of a selected subset of a uv data set
SPXMD adds images of model objects to image cubes in spectral index
SQASH Task to collapse several planes in a cube into one plane or row
STACK Task to co-add a set of 2-dimensional images with weighting
STARS Task to generate an ST ext. file with star positions
STEER Task which deconvolves the David Steer way.
STESS Task which finds sensitivity in mosaicing
STFND Task to find stars in an image and generate an ST table.
STFUN Task to calculate a structure function image
STRAN Task compares ST tables, find image coordinates (e.g. guide star)
SUBIM Task to select a subimage from up to a 7-dim. image
SUFIX modifies source numbers on uv data
SUMIM Task to sum overlapping, sequentially-numbered images
SUMSQ Task to sum the squared pixel values of overlapping images
SWAPR modifies UV data by swapping real and imaginary parts
SWPOL Swap polarizations in a UV data base
SY2TY Task to generate a TY extension file from an EVLA SY table
SYFIX modifies OTF SY DATA to match the source numbers in the data set
SYHIS Task to clip, filter, smooth SY tables and plot histograms
SYPRT Task to print values from the SY table

SYSOL	undoes and re-does nominal sensitivity application for Solar data
SYVSN	Plots selected SY/TY data with SN solution values versus time
TABED	Task to edit tables
TACOP	task to copy tables, other extension files
TADIF	prints differences between two table-format extension files
TAFFY	User definable task to operate on an image
TAFLG	Flags data in a Table extension file
TAMRG	Task to merge table rows under specified conditions
TAPLT	Plots data from a Table extension file
TAPPE	Task to append 2 tables and merge to output table
TARPL	Plot input and output text files of the TARS task
TARS	Simulation of Faraday rotation synthesis (mainly task FARS)
TASAV	Task to copy all extension tables to a dummy uv or map file
TASRT	Task to sort extension tables.
TBAVG	Time averages data combining all baselines.
TBDIF	Compare entries in two tables
TBIN	Reads a text file AIPS table into AIPS
TBOUT	Writes an AIPS table into a text file for user editting.
TBSUB	Make a new table from a subset of an old table
TBTSK	Paraform OOP task for tables
TCOPY	Tape to tape copy with some disk FITS support
TECOR	Calculate ionospheric delay and Faraday rotation corrections
TEPLT	Plots selected contents of TE (TECOR) parameter tables
TFILE	sorts and edits MkIII correlator UNIX-based A-file.
TGREP	finds all user numbers which have used specified task
TI2HA	modifies times in UV data to hour angles
TIORD	checks data for time baseline ordering, displays failures
TKPL	Task to send a plot file to the TEK
TLCAL	Converts JVLA telcal files to an SN file
TRANS	Task to transpose a subimage of an up to 7-dim. image
TRUEP	determines true antenna polarization from special data sets
TVCPS	Task to copy a TV screen-image to a PostScript file.
TVDIC	Task to copy a TV screen-image to a Dicomed film recorder.
TVFLG	interactive flagging of UV data using the TV
TVHLD	Interactive task to load an image to the TV with histogram equalization
TVHUI	make TV image from images of intensity, hue, saturation
TVIEW	Display image cube in all 3 transpositions with rotation & reference
TVPL	Display a plot file on the TV
TVRGB	make TV image from images of true color (RGB) images
TVSAD	Finds and fits Gaussians to portions of an image with interaction
TVSPC	Display images and spectra from a cube including fitting of spectra
TVSPX	Fits spectral indexes to each image row, allows interactive edit
TXPL	Displays a plot (PL) file on a terminal or line printer
TX2IM	Converts a 1-dimensional text file into cataloged image
TYAPL	undoes and re-does nominal sensitivity application
TYCOP	copy TY or SY table calibration between IFs
TYSMO	smooths and filters a calibration TY or SY table
UBAVG	Baseline dependent time averaging of uv data
UFLAG	Plots and edits data using a uv-plane grid and the TV
UJOIN	modifies UV data converting IFs to spectral channels
UNCAL	sets up tables for uncalibrating Australia Telescope data
UNSPX	Removes spectral index model from a data cube
USUBA	Assign subarrays within a uv-data file
UTESS	deconvolves images by maximizing emptiness
UV2MS	Append single-source file to multi-source file.
UV2TB	Converts UV autocorrelation spectra to tables
UVADC	Fourier transforms and corrects a model and adds to uv data.
UVADD	Combines two matching uv data sets
UVAVG	Average or merge a sorted (BT, TB) uv database
UVBAS	averages several channels and subtracts from uv data.

UVCMP	Convert a UV database to or from compressed format
UVCON	Generate sample UV coverage given a user defined array layout
UVCOP	Task to copy a subset of a UV data file
UVCRS	Finds the crossing points of UV-ellipses.
UVDEC	Decrements the number of spectral channels, keeping every nth
UVDGP	Copy a UV data file, deleting a portion of it
UVDI1	Subtract UV data(averaged up to one time) from the other UV data
UVDIF	prints differences between two UV data sets
UVFIL	Create, fill a uv database from user supplied information
UVFIT	Fits source models to uv data.
UVFIX	Recomputes u,v,w for a uv database
UVFLG	Flags UV-data
UVFND	prints selected data from UV data set to search for problems
UVFRE	Makes one data set have the spectral structure of another
UVGIT	Fits source models to uv data.
UVGLU	Glues UV data frequency blocks back together
UVHGM	Plots statistics of uv data files as histogram.
UVHIM	Makes image of the histogram on two user-chosen axes
UVHOL	Prints holography data from a UV data base with calibration
UVIMG	Grid UV data into an "image"
UVLIN	Fits and removes continuum visibility spectrum, also can flag
UVLOD	Read export or FITS data from a tape or disk
UVLSD	least squares fit to channels and divides the uv data.
UVLSF	least squares fit to channels and subtracts from uv data.
UVMAP	makes images from calibrated UV data.
UVMLN	edits data based on the rms of the spectrum
UVMOD	Modify UV database by adding a model incl spectral index
UVMTH	Averages one data set and applied it to another.
UVNOU	flags uv samples near the U,V axes to reduce interference
UVPLT	Plots data from a UV data base in multiple ways
UVPOL	modifies UV data to make complex image and beam
UVPRM	measures parameters from a UV data base
UVPRT	prints data from a UV data base with calibration
UVRFI	Mitigate RFI by Fourier transform or fitting the circle
UVRMS	computes statistics of a selected subset of a uv data set
UVSEN	Determine RMS sidelobe level and brightness sensitivity
UVSIM	Generate sample UV coverage given a user defined array layout
UVSRT	Sort a UV dataset into a specified order
UVSUB	Subtracts/divides a model from/into a uv data base
UVWAX	flags uv samples near the U,V axes to reduce interference
VBCAL	Scale visibility amplitudes by antenna based constants
VBGLU	Glues together data from multiple passes thru the VLBA corr.
VBMRG	Merge VLBI data, eliminate duplicate correlations
VBRFI	Plots spectral statistics of VLBA autocorrelation data for RFI
VH2RL	Converts from linear to circular polarization and vice versa
VHCAL	Do divide by I,Q,U models for linear polarization data
VHDIF	find/apply V minus H linear polarization phase difference
VHMOD	Compute linear polarization point source model, applies to data
VLABP	VLA antenna beam polarization correction for snapshot images
VLAMP	Makes ANTAB file for phased VLA used in VLBI observations
VLANT	applies VLA/EVLA antenna position corrections from OPs files
VLBCD	Reads ascii file containing VLBA calibration info into a CD table
VLBIN	Task to read VLBI data from an NRAO/MPI MkII correlator
VLBRF	Plots spectral statistics of VLBA autocorrelation data for RFI
VLBSY	Reads ascii file containing VLBA total power info into an SY table
VLOG	Pre-process external VLBA calibration files
VPFLG	Resets flagging to all or all corss-hand whenever some are flagged
VPLOT	Plots uv data and model from CC file
VTESS	Deconvolves sets of images by the Maximum Entropy Method

VTEST	Measures velocity discrepancy across fields
WARP	Model warps in Galaxies
WETHR	Plots selected contents of WX tables, flags data based on WX
WIPER	plots and edits data from a UV data base using the TV
WTMOD	modifies weights in a UV data set
WTSUM	Task to do a a sum of images weighted by other images
WXLOD	Reads KEYIN-format weather information making WX table
XBASL	Fits and subtracts nth-order baselines from cube (x axis)
XG2PL	Plots spectrum of a pixel with XGAUS/AGAUS and ZEMAN/ZAMAN fits
XG2XG	Changes number of components in an XG or ZE table creating new one
XGAUS	Fits 1-dimensional Gaussians to images: restartable
XMOM	Fits one-dimensional moments to each row of an image
XPLOT	Plots image rows one at a time on graphics, TV, or plot file
XSMTH	Smooth data along the x axis
XSUM	Sum or average images on the x axis
XTRAN	Create an image with transformed coordinates
XYDIF	find/apply X minus Y linear polarization phase difference
XYMOD	Compute linear polarization point source model, applies to data
ZAMAN	Fits 1-dimensional Zeeman model to absorption-line data
ZEMAN	Fits 1-dimensional Zeeman model to data

13.36 TV

BLANK	blanks out selected, e.g. non-signal, portions of an image
CHARMULT	Changes the multiplication factor for TV characters
CNTR	generate a contour plot file or TV plot from an image
COLORS	specifies the desired TV colors
COSTAR	Verb to plot a symbol at given position on top of a TV image
COTVLOD	Proc to load an image into a TV channel about a coordinate
CURBLINK	switch TV cursor between steady and blinking displays
CURVALUE	displays image intensities selected via the TV cursor
DEFCOLOR	Sets adverb PLCOLORS to match s default XAS TV
DELBOX	Verb to delete boxes with TV cursor & graphics display.
DELTAX	Increment or size in X direction
DELTAY	Increment or size in Y direction
DFILEBOX	Verb to delete Clean boxes with TV cursor & write to file
DONEWTAB	do we make new tables, use a new table format, etc.
DOTV	selects use of TV display option in operation
DRAWBOX	Verb to draw Clean boxes on the display
FACTOR	scales some display or CLEANing process
FILEBOX	Verb to reset Clean boxes with TV cursor & write to file
FILIT	Interactive BOXFILE editing with facet images
GR2CHAN	specifies a second TV graphics channel to be used
GRBLINK	Verb which blinks 2 TV graphics planes
GRCHAN	specifies the TV graphics channel to be used
GRCLEAR	clears the contents of the specified TV graphics channels
GREAD	reads the colors of the specified TV graphics channel
GROFF	turns off specified TV graphics channelS
GRON	turns on specified TV graphics channelS
GWRITE	Sets the colors of the specified TV graphics channel
HLPTVHLD	Interactive image display with histogram equalization - run-time help
HLPTVIEW	Interactive display of cube in 3 transpositions - run-time help
HLPTVSAD	Find & fit Gaussians to an image with interaction - run-time help
HLPTVSPC	Interactive display of spectra from a cube - run-time help
HUEWEDGE	Show a wedge on the TV suitable for TVHUEINT displays
HUINT	make RGB image from images of intensity & hue, like TVHUEINT

IM2TV	Verb to convert pixel coordinates to TV pixels
IMERASE	replaces an image portion of the TV screen with zeros
IMLHS	converts images to luminosity/hue TV display
IMPOS	displays celestial coordinates selected by the TV cursor
IMWEDGE	load step wedge of full range of image values to TV
IMXY	returns pixel coordinates selected by the TV cursor
MFITSET	gets adverbs for running IMFIT and JMFIT
NBOXES	Number of boxes
NCCBOX	Number of clean component boxes
OFFHUINT	Proc which restores TV functions to normal after TVHUEINT
OFFPSEUD	Verb which deactivates all pseudo-color displays
OFFSCROL	Verb which deactivates scroll of an image
OFFTRAN	Verb which restores transfer function to normal
OFFZOOM	Verb which returns the hardware IIS zoom to normal
OFMCOLOR	Verb to access many OFMs to enhance TV pseudo-color displays
OFMFILE	specifies the name of a text file containing OFM values
OKCOLORS	Sets adverb PLCOLORS to use less ink (white background)
PCNTR	Generate plot file with contours plus polarization vectors
PIX2XY	Specifies a pixel in an image
PIXAVG	Average image value
PIXRANGE	Range of pixel values to display
PIXSTD	RMS pixel deviation
PIXVAL	Value of a pixel
PROFL	Generates plot file for a profile display.
REBOX	Verb to reset boxes with TV cursor & graphics display.
REMOVIE	Verb to rerun a previously loaded (TVMOVIE) movie
REROAM	Verb to use previous roam image mode, then does roam
RGBCOLOR	specifies the desired TV graphics color
RGBGAMMA	specifies the desired color gamma corrections
RGBMP	Task to create an RGB image from the 3rd dim of an image
ROAM	Roam around an image too large for the display.
ROAMOFF	Verb to recover image from roam display in simple display mode
ROMODE	Specified roam mode
SETMAXAP	Examines/alters system parameter limiting dynamic pseudo-AP
SETSLICE	Set slice endpoints on the TV interactively
SETXWIN	Procedure to set BLC and TRC with TV cursor
STARTTV	Restarts XAS (TV), MS, TK servers that have disappered
TBLC	Gives the bottom left corner of an image to be displayed
TTRC	Specifies the top right corner of a subimage to be displayed
TV1SET	Verb to reset 1D gaussian fitting initial guess on TV plot.
TV2CHAN	Specified a second TV channel (plane) used in some operations
TV2COLOR	Verb to activate a 2-color pseudo-color display on the TV
TV3COLOR	Verb to initiate 3-color display using 3 TV channels
TVACOMPS	Verb to add slice model components directly on TV graphics
TVAGUESS	Verb to re-plot slice model guess directly on TV graphics
TV3LOD	Procedure to load an RGB image into 3 TV channels and display it
TVALL	Procedure loads image to TV, shows labeled wedge, enhances
TVAMODEL	Verb to add slice model display directly on TV graphics
TVANOT	Verb to load anotation to the TV image or graphics
TVARESID	Verb to add slice model residuals directly on TV graphics
TVASLICE	Verb to add a slice display on TV graphics from slice file
TVBANDW	Verb to activate three types of black and white TV enhancements
TVBLINK	Verb which blinks 2 TV planes, can do enhancement also
TVBOX	Verb to set boxes with TV cursor & graphics display.
TVBUT	Tells which AIPS TV button was pushed
TVCHAN	Specified a TV channel (plane)
TVCLEAR	Verb to clear image from TV channel(s)
TVCOLORS	Sets adverb PLCOLORS to match the TV (DOTV=1) usage
TVCOMPS	Verb to display slice model components directly on TV graphics
TVCORN	Specified the TV pixel for the bottom left corner of an image
TVCPS	Task to copy a TV screen-image to a PostScript file.

TVCUBE	Verb to load a cube into tv channel(s) & run a movie
TVDATA	displays information about the current TV display
TVDIC	Task to copy a TV screen-image to a Dicomed film recorder.
TVDIST	determines spherical distance between two pixels on TV screen
TVFACTS	displays information about the current TV display
TVFIDDLE	Verb enhances B/W or color TV image with zooms
TVFLUX	displays coordinates and values selected with the TV cursor
TVGUESS	Verb to display slice model guess directly on TV graphics
TVHELIX	Verb to activate a helical hue-intensity TV pseudo-coloring
TVHLD	Interactive task to load an image to the TV with histogram equalization
TVHUEINT	Verb to make hue/intensity display from 2 TV channels
TVHUI	make TV image from images of intensity, hue, saturation
TVIEW	Display image cube in all 3 transpositions with rotation & reference
TVILINE	Verb to draw a straight line on an image on the TV
TVINIT	Verb to return TV display to a virgin state
TVLABEL	Verb to label the (map) image on the TV
TVLAYOUT	Verb to label the holography image on the TV with panel layout
TVLEVS	Gives the peak intensity to be displayed in levels
TVLINE	Verb to load a straight line to the TV image or graphics
TVLOD	Verb to load an image into a TV channel
TVLUT	Verb which modifies the transfer function of the image
TVMAXFIT	displays fit pixel positions and intensity at maxima on TV
TVMBLINK	Verb which blinks 2 TV planes either auto or manually
TVMLUT	Verb which modifies the transfer function of the image
TVMODEL	Verb to display slice model directly on TV graphics
TVMOVIE	Verb to load a cube into tv channel(s) & run a movie
TVNAME	Verb to fill image name of that under cursor
TVOFF	Verb which turns off TV channel(s).
TVON	Turns on one or all TV image planes
TVPHLAME	Verb to activate "flame-like" pseudo-color displays
TVPL	Display a plot file on the TV
TVPOS	Read a TV screen position using cursor
TVPSEUDO	Verb to activate three types of pseudo-color displays
TVRESET	Reset the TV without erasing the image planes
TVRESID	Verb to display slice model residuals directly on TV graphics
TVRESTOR	Restarts XAS (TV), MS, TK servers that have disappered
TVRGB	make TV image from images of true color (RGB) images
TVROAM	Load up to 16 TV image planes and roam a subset thereof
TVSAD	Finds and fits Gaussians to portions of an image with interaction
TVSCROL	Shift position of image on the TV screen
TVSET	Verb to set slice Gaussian fitting initial guesses from TV plot
TVSLICE	Verb to display slice file directly on TV
TVSPC	Display images and spectra from a cube including fitting of spectra
TVSPLIT	Compare two TV image planes, showing halves
TVSTAR	Verb to plot star positions on top of a TV image
TVSTAT	Find the mean and RMS in a blotch region on the TV
TVTRANSF	Interactively alters the TV image plane transfer function
TVWEDGE	Show a linear wedge on the TV
TVWINDOW	Set a window on the TV with the cursor
TVWLABEL	Put a label on the wedge that you just put on the TV
TVXY	Pixel position on the TV screen
TVZOOM	Activate the TV zoom
TXINC	TV X coordinate increment
TYINC	TV Y coordinate increment
TZINC	TV Z coordinate increment
WEDERASE	Load a wedge portion of the TV with zeros
XAS	Information about TV-Servers

13.37 TV-APPL

AGAUS	Fits 1-dimensional Gaussians to absorption-line spectra
BLSUM	sums images over irregular sub-images, displays spectra
BPEDT	Interactive TV task to edit uv data based on BP tables
EDITA	Interactive TV task to edit uv data based on TY/SY/SN/CL tables
EDITR	Interactive baseline-oriented visibility editor using the TV
FTARS	Fits 1-dimensional polarization spectrum to Q/U text file
FTFLG	interactive flagging of UV data in channel-time using the TV
GAMMASET	changes the gamma-correction exponent used in the TV OFM
HLPAGAUS	Interactive Gaussian absorption fitting task AGAUS - run-time help
HLPCLEAN	Cleaning tasks - run-time help
HLPEDIBP	Interactive BP table uv-data editor BPEDT - run-time help
HLPEDICL	Interactive SN/CL table uv-data editor - run-time help
HLPEDIPC	Interactive PC table editor PCEDT - run-time help
HLPEDIPD	Interactive PD table editor PDEDT - run-time help
HLPEDISN	Interactive SN/CL table (not UV) editor - run-time help
HLPEDISS	Interactive SY table (not UV) editor - run-time help
HLPEDISY	Interactive SY table uv-data editor EDITA - run-time help
HLPEDITS	Interactive TY table (not UV) editor - run-time help
HLPEDITY	Interactive TY table uv-data editor EDITA - run-time help
HLPEDIUV	Interactive uv-data editor EDITR - run-time help
HLPFILIT	Interactive Clean box file editing with image display - run-time help
HLPFTFLG	Interactive time-channel visibility Editor - run-time help
HLPIBLED	Interactive Baseline based visibility Editor - run-time help
HLPLISPX	Fits spectral index model to data in text file - run-time help
HLPPCFLG	Interactive time-channel PC table Editor PCFLG - run-time help
HLPPLAYR	OOP TV class demonstration task - run-time help
HLPRMFIT	Polarization fitting task RMFIT - run-time help
HLPSCIMG	Full-featured image plus self-cal loops, editing - run-time help
HLPSCMAP	Imaging plus self-cal and editing SCMAP - run-time help
HLPSPFLG	Interactive time-channel visibility Editor SPFLG - run-time help
HLPTVFLG	Interactive time-baseline visibility Editor TVFLG - run-time help
HLPTVHUI	Interactive intensity-hue-saturation display - run-time help
HLPTVRGB	Interactive red-green-blue display - run-time help
HLPTVSPX	Fits 1-dimensional spectral index model to data - run-time help
HLPUFLAG	Edit uv-data on a grid UFLAG - run-time help
HLPWIPER	edit uv data from UVPLT-like plot WIPER - run-time help
HLPXGAUS	Interactive Gaussian fitting task XGAUS - run-time help
HLPZAMAN	Fits 1-dimensional Zeeman model to absorption data - run-time help
HLPZEMAN	Fits 1-dimensional Zeeman model to data - run-time help
IBLED	Interactive BaseLine based visibility EDitor
IMAGR	Wide-field and/or wide-frequency Cleaning / imaging task.
LISPX	Fits spectral indexes to a list of spectra, allows interactive edit
OFMADJUS	interactive linear adjustment of current TV OFM lookup tables
OFMCONT	creates/modifies TV color OFMs with level or wedged contours
OFMDIR	lists names of the user's and system's OFM files from OFMFIL
OFMGET	loads TV OFMS from a system or user OFM save file
OFMLIST	lists the current TV OFM table(s) on the terminal or printer
OFMPL	Task to display and contrast system and user OFM color tables.
OFMSAVE	saves the TV's current OFM lookup table in a text file
OFMSTRCH	interactive modification of current TV OFM lookup tables
OFMTWEAK	interactive modification of current TV OFM lookup tables
OFMZAP	deletes an OFM lookup table save file
PCEDT	Interactive TV task to edit pulse-cal (PC) tables
PCFLG	interactive flagging of Pulse-cal data in channel-TB using the TV
PDEDT	Interactive TV task to edit polarization D-term (PD) tables

PLAYR	Task to load one or two images and play with them on the TV
RM2PL	Plots spectrum of a pixel with RMFIT fit
RMFIT	Fits 1-dimensional polarization spectrum to Q/U cube
SNEDT	Interactive SN/CL/TY/SY table editor using the TV
SPFLG	interactive flagging of UV data in channel-TB using the TV
TDEPEND	Time-dependent imaging procedure sequence
TVFLG	interactive flagging of UV data using the TV
TVSPX	Fits spectral indexes to each image row, allows interactive edit
UFLAG	Plots and edits data using a uv-plane grid and the TV
WIPER	plots and edits data from a UV data base using the TV
XBASL	Fits and subtracts nth-order baselines from cube (x axis)
XG2PL	Plots spectrum of a pixel with XGAUS/AGAUS and ZEMAN/ZAMAN fits
XG2XG	Changes number of components in an XG or ZE table creating new one
XGAUS	Fits 1-dimensional Gaussians to images: restartable
XPLOT	Plots image rows one at a time on graphics, TV, or plot file
ZAMAN	Fits 1-dimensional Zeeman model to absorption-line data
ZEMAN	Fits 1-dimensional Zeeman model to data

13.38 UTILITY

ANTNUM	Returns number of a named antenna
CCEDT	Select CC components in BOXes and above mininum flux.
CCSEL	Select signifigant CC components
CL2HF	Convert CL table to HF table
DOOSRO	calibrating amplitude and phase, and imaging VLA data
DOVLAMP	Produces amp calibration file for phased-VLA VLBI data
EPOCONV	Convert between J2000 and B1950 coordinates
MAXTAB	Returns maximum version number of named table
MBDLY	Fits multiband delays from IF phases, updates SN table
MK3TX	extract text files from a MKIII VLBI archive tape
MOVE	Task to copy or move data from one user or disk to another
NANS	reads an image or a UV data set and looks for NaNs
OPCODE	General adverb, defines an operation
OPTELL	The operation to be passed to a task by TELL
PIPEAIPS	calibrating amplitude and phase, and imaging VLA data
PRNUMBER	POPS number of messages
PRTIME	Time limit
RUNWAIT	Runs a task and waits for it to finish
SCANTIME	Returns time range for a given scan number
SHOW	Verblike to display the TELL adverbs of a task.
SORT	Specified desired sort order
SQASH	Task to collapse several planes in a cube into one plane or row
STRAN	Task compares ST tables, find image coordinates (e.g. guide star)
SWAPR	modifies UV data by swapping real and imaginary parts
TBDIF	Compare entries in two tables
TBIN	Reads a text file AIPS table into AIPS
TBOUT	Writes an AIPS table into a text file for user editting.
TBSUB	Make a new table from a subset of an old table
TBTSK	Paraform OOP task for tables
TCOPY	Tape to tape copy with some disk FITS support
UVAVG	Average or merge a sorted (BT, TB) uv database
UVCMP	Convert a UV database to or from compressed format
UVDI1	Subtract UV data(averaged up to one time) from the other UV data
UVNOU	flags uv samples near the U,V axes to reduce interference
UVRFI	Mitigate RFI by Fourier transform or fitting the circle
UVWAX	flags uv samples near the U,V axes to reduce interference

VLAPROCS	Procedures to simplify the reduction of VLBA data
VLARUN	Calibrating amplitude and phase, and imaging VLA data
VLASUMM	Plots selected contents of SN or CL files
VLBAAMP	applies a-priori amplitude corrections to VLBA data
VLBAARCH	Procedure to archive VLBA correlator data
VLBACALA	applies a-priori amplitude corrections to VLBA data
VLBACCOR	applies a-priori amplitude corrections to VLBA data
VLBACRPL	Plots crosscorrelations
VLBAEOPS	Corrects Earth orientation parameters
VLBAFIX	Procedure that fixes VLBA data, if necessary
VLBAFPOL	Checks and corrects polarization labels for VLBA data
VLBAFQS	Copies different FQIDS to separate files
VLBAFRGP	Fringe fit phase referenced data and apply calibration
VLBAFRNG	Fringe fit data and apply calibration
VLBAKRGP	Fringe fit phase referenced data and apply calibration
VLBAKRNG	Fringe fit data and apply calibration
VLBALOAD	Loads VLBA data
VLBAMCAL	Merges redundant calibration data
VLBAMPCL	Calculates and applies manual instrumental phase calibration
VLBAMPHC	applies VLBARUN calibration to additional phase stopping centers
VLBAPANG	Corrects for parallactic angle
VLBAPCOR	Calculates and applies instrumental phase calibration
VLBAPIPE	Applies amplitude and phase calibration procs to VLBA data
VLBARUN	Applies amplitude and phase calibration procs to VLBA data
VLBASNPL	Plots selected contents of SN or CL files
VLBASRT	Sorts VLBA data, if necessary
VLBASUBS	looks for subarrays in VLBA data
VLBASUMM	Prints a summary of a VLBI experiment
VLBATECR	Calculate ionospheric delay and Faraday rotation corrections
VLBAUTIL	Procedures to simplify the reduction of VLBA data

13.39 UV

ACCOR	Corrects cross amplitudes using auto correlation measurements
ACIMG	Makes image of autocorrelation data showing time vs frequency
ACLIP	edits suto-corr data for amplitudes, phases, and weights out of range
ACSCL	Corrects cross amplitudes using auto correlation measurements
ADDIF	Adds an IF axis to a uv data set
AFILE	sorts and edits MkIII correlator A-file.
ALIAS	adverb to alias antenna numbers to one another
ALVAR	Plots the Allan Variance statistic of a UV data set
ALVPR	prints statistics on the Allan Variance of a UV data set
AMKAT	renumbers MeerKAT antennas so taht antenna number =¿ station ID
ANBPL	plots and prints uv data converted to antenna based values
ASDMFILE	Full path to EVLA ASDM/BDF directory
ATLOD	Reads ATCA data in RPFITS format into AIPS
AVER	Averages over time UV data sets in 'BT' order
AVGWT	Computes the average visibility weight of the included data
AVOPTION	Controls type or range of averaging done by a task
AVSPC	Averages uv-data in the frequency domain
BAND	specifies the approximate frequency of UV data to be selected
BASFIT	fits antenna locations from SN-table data
BASRM	Task to remove a spectral baseline from total power spectra
BDF2AIPS	Read EVLA ASDM/BDF data into AIPS
BDFLIST	List contents of EVLA ASD data file

BIF	gives first IF (spectral window) to be included
BLAVG	Average cross-polarized UV data over baselines.
BLOAT	converts line data to greater number channels
BLPLT	Plots BLCAL tables in 2 dimensions as function of baseline and IF
BPASSPRM	Control adverb array for bandpass calibration
BPEDT	Interactive TV task to edit uv data based on BP tables
BPEPL	Plots bandpass tables overlaying multiple times like BPEDT
BPFLG	Flgas uv data based on bad BP values
BPLOT	Plots bandpass tables in 2 dimensions as function of time
BPPLT	Plots selected contents of bandpass (BP) files versus time
BPSMO	Smooths or interpolates bandpass tables to regular times
BPWAY	Determines channel-dependent relative weights
BPWGT	Calibrates data and scales weights by bandpass correction
BREAK	procedure to TELL FILLM to break all current uv files, start new
BSCAN	seeks best scan to use for phase cal, fringe search, ..
BSMOD	creates single-dish UV beam-switched data with model sources
BSROT	modifies SD beam-switch continuum data for error in throw
CALIB	determines antenna calibration: complex gain
CALMODEL	Applies SPLIT & UVSUB to make a model data set matching a calibrator
CAPLT	plots closure amplitude and model from CC file
CENTR	modifies UV data to center the reference channel
CLIP	edits data based on amplitudes, phases, and weights out of range
CLOSE	plots closure phase or amplitude spectra averaged over closures
CLPLT	plots closure phase and model from CC file
CMETHOD	specifies the method by which the uv model is computed
CMODEL	specifies the method by which the uv model is computed
COHER	Baseline Phase coherence measurement
CONFIG	Configuration ID number within an EVLA ASDM/BDF data set
CORER	calculates correlator statistics and flags bad ones
CORFQ	corrects uvw for incorrect observing frequency
CVEL	shifts spectral-line UV data to a given velocity
CXIMAGR	Grid UV data into a complex image, Fourier transform, Clean
DAYFX	Fixes day number problems left by FILLM
DBAPP	appends one or more data sets to the output data set
DBCON	concatenates two UV data sets
DECOR	Measures the decorrelation between channels and IF of uv data
DEFER	Controls when file creation takes place
DEFLG	edits data based on decorrelation over channels and time
DENAN	modifies UV data flagging all NaNs
DESCM	copies a portion of a UV data set
DFQID	modifies UV data changing the indicated FQIDs
DFTIM	Makes image of DFT at arbitrary point showing time vs frequency
DFTPL	plots DFT of a UV data set at arbitrary point versus time
DFTQU	computes DFT of a UV data set at arbitrary point in Q and U polariz.
DGAUS	Correct UV model for convolution by Clean beam
DIFRL	divides the RR data by LL data
DIFUV	Difference two matching uv data sets, write out or print statistics
DOACOR	specifies whether autocorrelation data are included
DOARRAY	spcifies if subarrays are ignored or the information used
DOBTWEEN	Controls smoothing between sources in calibration tables
DOCONCAT	selects concatenated or indivudual output files
DOEBAR	Controls display of estimates of the uncertainty in the data
DOIFS	controls functions done across IFs
DOREAL	Say to do operation with real/imaginary rather than amplitude/phase
DOROBUST	Controls method of averaging - simple mean/rms or robust
DOSTOKES	selects options related to polarizations
DOUVCOMP	selects use of compression in writing UV data to disk
DQUAL	Rearranges source list, dropping qualifiers
DSORC	copies a data set elliminating some source numbers
DSTOK	Drops the cross-hand polarizations

DT2PD	Write PD table from a PD table text file
DTCHK	Task to check results of a test using simulated data.
DTSUM	Task to provide a summary of the contents of a dataset
EDITA	Interactive TV task to edit uv data based on TY/SY/SN/CL tables
EDITR	Interactive baseline-oriented visibility editor using the TV
EIF	last IF (spectral window) number to be included in operation
ELFIT	Plots/fits selected contents of SN, TY, SY, PC or CL files
EVACL	Evaluates statistics in closure phase and amplitude
EVASN	Evaluates statistics in SN/CL tables
EVAUV	Subtracts & divides a model into UV data, does statistics on results
EVLA	puts the list of eVLA antennas in the current file on stack
FEW	procedure to TELL FILLM to append incoming data to existing uv files
FGCNT	Counts samples comparing two flag tables or detailed count from ¡= 1
FGDIF	Compares affect of 2 FG tables
FGPLT	Plots selected contents of FG table
FGPRT	Prints counts of selected portions of an FG table.
FGSPW	Flags bad spectral windows
FGTAB	writes frequency-range flags to a text file to be read by UVFLG
FILLM	reads VLA on-line/archive format uv data tapes (post Jan 88)
FILLR	reads old VLA on-line-system tapes into AIPS
FINDR	Find normal values for a uv data set
FITAB	writes images / uv data w extensions to tape in FITS format
FITDISK	writes images / uv data w extensions to disk in FITS format
FITTP	writes images / uv data w extensions to tape in FITS format
FIXAL	least squares fit aliasing function and remove
FIXAN	fixes the contents of the ANtenna extension file
FIXFR	Modifies UV data set making all frequencies increase or decrease
FIXRL	correctes right vs left polarizations for a list of antennas
FIXWT	Modify weights to reflect amplitude scatter of data
FLAGR	Edit data based on internal RMS, amplitudes, weights
FLGIT	flags data based on the rms of the spectrum
FLOPM	reverses the spectral order of UV data, can fix VLA error
FOV	Specifies the field of view
FQCENTER	specifies that the frequency axis should be centered
FREQSEL	List of frequencies desired
FRMAP	Task to build a map using fringe rate spectra
FRPLT	Task to plot fringe rate spectra
FTFLG	interactive flagging of UV data in channel-time using the TV
FUDGE	modifies UV data with user's algorithm: paraform task
FXALIAS	least squares fit aliasing function and remove
FXPOL	Corrects VLBA polarization assignments
FXTIM	fixes start date so all times are positive
FXVLA	Task to correct VLA data for on-line errors in special cases.
FXVLB	Builds a CQ table to enable VLBA correlator loss corrections
GCPLT	Plots gain curves from text files
GETITIME	Verb to return the UV data integration time.
HA2TI	Converts data processed by TI2HA (STUFFER) back to real times
HAFIX	Recomputes u,v,w when time is hour angle (UVdata is output of TI2HA)
HLPEDIBP	Interactive BP table uv-data editor BPEDT - run-time help
HLPEDICL	Interactive SN/CL table uv-data editor - run-time help
HLPEDIPC	Interactive PC table editor PCEDT - run-time help
HLPEDIPD	Interactive PD table editor PDEDT - run-time help
HLPEDISN	Interactive SN/CL table (not UV) editor - run-time help
HLPEDISS	Interactive SY table (not UV) editor - run-time help
HLPEDISY	Interactive SY table uv-data editor EDITA - run-time help
HLPEDITS	Interactive TY table (not UV) editor - run-time help
HLPEDITY	Interactive TY table uv-data editor EDITA - run-time help
HLPEDIUV	Interactive uv-data editor EDITR - run-time help
HLPFTFLG	Interactive time-channel visibility Editor - run-time help

HLPIBLED	Interactive Baseline based visibility Editor - run-time help
HLPPCFLG	Interactive time-channel PC table Editor PCFLG - run-time help
HLPSPFLG	Interactive time-channel visibility Editor SPFLG - run-time help
HLPTVFLG	Interactive time-baseline visibility Editor TVFLG - run-time help
HLPUFLAG	Edit uv-data on a grid UFLAG - run-time help
HLPWIPER	edit uv data from UVPLT-like plot WIPER - run-time help
HOLGR	Read & process holography visibility data to telescope images
HOLIS	Prints information about holography sampling
HOLOG	Read & process holography visibility data to telescope images
HSA	puts the list of HSA antennas in the current file on stack
IBLED	Interactive BaseLine based visibility EDitor
IM2UV	converts an image to a visibility data set
IMFRING	large image delay fitting with IM2CC and OOFRING
IMSCAL	large image self-cal with IM2CC and OOCAL
INTTIME	adds user-generated lines to the history extension file
LISTR	prints contents of UV data sets and assoc. calibration tables
LISUN	prints contents of UV data sets with Sun angles
LOCIT	fits antenna locations from SN-table data
LPCAL	Determines instrumental polarization for UV data
M3TAR	translate Haystack MKIII VLBI format "A" TAR's into AIPS
MANY	procedure to TELL FILLM to start new uv files on each scan
MAPBM	Map VLA beam polarization
MATCH	changes antenna, source, FQ numbers to match a data set
MK3IN	translate Haystack MKIII VLBI format "A" tapes into AIPS
MOD3D	Computes a 3D CC model from a set of facets
MORIF	Combines IFs or breaks spectral windows into multiple windows (IFs)
MSORT	Sort a UV dataset into a specified order
MULIF	Change number of IFs in output
MULTI	Task to convert single-source to multi-source UV data
NANS	reads an image or a UV data set and looks for NaNs
NCHAN	Number of spectral channels in each spectral window
NIF	Number of IFs (spectral windows) in a data set
NOIFS	makes all IFs into single spectrum
NPIECE	The number of pieces to make
NSKIP	Number of objects to skip
OBJECT	The name of an object
OBPLT	Plot columns of an OB table.
OFUDG	modifies UV data with user's algorithm no cal adverbs: paraform task
OMFIT	Fits sources and, optionally, a self-cal model to uv data
OOCAL	determines antenna complex gain with frequency-dependent models
OOSRT	Sort a UV dataset into a specified order
OOSUB	Subtracts/divides a model from/into a uv data base
OTFMODE	Specifies whether the data are in on-the-fly mode or not
PCAL	Determines instrumental polarization for UV data
PCASS	Finds amplitude bandpass shape from pulse-cal table data
PCCOR	Corrects phases using PCAL tones data from PC table
PCEDT	Interactive TV task to edit pulse-cal (PC) tables
PCFIT	Finds delays and phases using a pulse-cal (PC) table
PCFLG	interactive flagging of Pulse-cal data in channel-TB using the TV
PCPLT	Plots pulse-cal tables in 2 dimensions as function of time
PCVEL	shifts spectral-line UV data to a given velocity: planet version
PDEDT	Interactive TV task to edit polarization D-term (PD) tables
PDPLT	Plots selected contents of PD tables
PEELR	calibrates interfering sources in multi-facet imges
PHASE	Baseline Phase coherence measurement
PHNEG	Negates a UV datafile's visibility phase.
PHSLIMIT	gives a phase value in degrees
PHSRF	Perform phase-referencing within a spectral line database.
PLOTC	Plots color schems used by 3-color plot tasks

PLRFI	Plots spectral statistics from output of VBRFI & VLBRF
POSSM	Task to plot total and cross-power spectra.
PPAPP	Applies Right minus Left phase difference to a 2nd data set
PRRFI	Prints excess spectral statistics from output of VBRFI & VLBRF
PRTAN	prints the contents of the ANtenna extension file
PRTUV	prints contents of a visibility (UV) data set
QCREATE	adverb controlling the way large files are created
QUACK	Flags beginning or end portions of UV-data scans
QUAL	Source qualifier
QUFIX	determines Right minus Left phase difference, corrects cal files
QUIT	procedure to TELL FILLM to stop at the end of the current scan
QUOUT	writes text file of Q, U versus frequency to be used by RLDIF
READISK	Reads images / uv data w extensions in FITS format into AIPS
REAMP	Scale visibility amplitudes by IF- or subarray-dependent factor
REFDATE	To specify the initial or reference date of a data set
REFLG	Attempts to compress a flag table
REIFS	Breaks spectral windows into multiple spectral windows (IFs)
RESEQ	Renumber antennas
RESOU	Changes (renumbers) a list of source numbers in UV data and tables
REWAY	computes weights based in rms in spectra
REWEIGHT	Reweighting factors for UV data weights.
REWGT	modifies UV data re-scaling the weights only
RFLAG	Flags data set based on time and freq rms in fringe visibilities
RIRMS	computes rms of real/imag of a selected subset of a uv data set
RLCAL	Determines instrumental right-left phase versus time (a self-cal)
RLCOR	corrects a data set for R-L phase differences
RLDIF	determines Right minus Left phase difference, corrects cal files
ROBUST	Uniform weighting "robustness" parameter
SBCOR	Task to correct VLBA data for phase shift between USB & LSB
SDLSF	least squares fit to channels and subtracts from SD uv data
SDMOD	modifies single-dish UV data with model sources
SDVEL	shifts spectral-line single-dish data to a given velocity
SETAN	Reads an ANtenna file info from a text file
SHADO	Calculate the shadowing of antennas at the array
SHAHO	Flags Holography data for shadowing
SHOUV	displays uv data in various ways.
SMOOTH	Specifies spectral smoothing
SNBLP	Plots selected contents of SN or CL files on a baseline basis
SNEDT	Interactive SN/CL/TY/SY table editor using the TV
SNFIT	Fits parabola to SN amplitudes and plots result
SNFLG	Writes flagging info based on the contents of SN files
SNIFS	Plots selected contents of SN, TY, SY, PC or CL files by IF
SNPLT	Plots selected contents of SN, SY, TY, PC or CL files versus time
SNREF	Chooses best reference antenna to minimize R-L differences
SNRMS	Print statistics for selected contents of SN, SY, TY, PC or CL files
SORT	Specified desired sort order
SOUSP	fits source spectral index from SU table or adverbs
SPCAL	Determines instrumental polzn. for spec. line UV data
SPECR	Spectral regridding task for UV data
SPFLG	interactive flagging of UV data in channel-TB using the TV
SPLAT	Applies calibration and splits or assemble selected sources.
SPLIT	converts multi-source to single-source UV files w calibration
SPMOD	Modify UV database by adding a model with spectral lines
SPRMS	Plots spectral statistics of a selected subset of a uv data set
STOKES	Stokes parameter
STOP	procedure to TELL FILLM to break all current uv files and stop
STUFFR	averages together data sets in hour angle
SUFIX	modifies source numbers on uv data
SWAPR	modifies UV data by swapping real and imaginary parts
SWPOL	Swap polarizations in a UV data base

SYFIX	modifies OTF SY DATA to match the source numbers in the data set
SYVSN	Plots selected SY/TY data with SN solution values versus time
TBAVG	Time averages data combining all baselines.
TELFLM	procedure to TELL real-time FILLM a new APARM(1) value
TEPLT	Plots selected contents of TE (TECOR) parameter tables
TFILE	sorts and edits MkIII correlator UNIX-based A-file.
TI2HA	modifies times in UV data to hour angles
TIORD	checks data for time baseline ordering, displays failures
TLCAL	Converts JVLA telcal files to an SN file
TRUEP	determines true antenna polarization from special data sets
TVFLG	interactive flagging of UV data using the TV
U2CAT	list a user's UV and scratch files on disk IN2DISK
U3CAT	list a user's UV and scratch files on disk IN3DISK
U4CAT	list a user's UV and scratch files on disk IN4DISK
U5CAT	list a user's UV and scratch files on disk IN5DISK
UBAVG	Baseline dependent time averaging of uv data
UCAT	list a user's UV and scratch files on disk INDISK
UFLAG	Plots and edits data using a uv-plane grid and the TV
UJOIN	modifies UV data converting IFs to spectral channels
UOCAT	list a user's UV and scratch files on disk OUTDISK
USUBA	Assign subarrays within a uv-data file
UV1TYPE	Convolving function type 1, pillbox or square wave
UV2MS	Append single-source file to multi-source file.
UV2TB	Converts UV autocorrelation spectra to tables
UV2TYPE	Convolving function type 2, exponential function
UV3TYPE	Convolving function type 3, sinc function
UV4TYPE	Convolving function type 4, exponent times sinc function
UV5TYPE	Convolving function type 5, spheroidal function
UV6TYPE	Convolving function type 6, exponent times BessJ1(x) / x
UVADC	Fourier transforms and corrects a model and adds to uv data.
UVADD	Combines two matching uv data sets
UVAVG	Average or merge a sorted (BT, TB) uv database
UVBAS	averages several channels and subtracts from uv data.
UVBOX	radius of the smoothing box used for uniform weighting
UVBXFN	type of function used when counting for uniform weighting
UVCMP	Convert a UV database to or from compressed format
UVCON	Generate sample UV coverage given a user defined array layout
UVCOP	Task to copy a subset of a UV data file
UVCOPPRM	Parameter adverb array for task UVCOP
UVCRS	Finds the crossing points of UV-ellipses.
UVDEC	Decrements the number of spectral channels, keeping every nth
UVDGP	Copy a UV data file, deleting a portion of it
UVDI1	Subtract UV data(averaged up to one time) from the other UV data
UVDIF	prints differences between two UV data sets
UVFIL	Create, fill a uv database from user supplied information
UVFIT	Fits source models to uv data.
UVFIX	Recomputes u,v,w for a uv database
UVFIXPRM	Parameter adverb array for task UVFIX
UVFLG	Flags UV-data
UVFND	prints selected data from UV data set to search for problems
UVFRE	Makes one data set have the spectral structure of another
UVGIT	Fits source models to uv data.
UVGLU	Glues UV data frequency blocks back together
UVHGM	Plots statistics of uv data files as histogram.
UVHIM	Makes image of the histogram on two user-chosen axes
UVHOL	Prints holography data from a UV data base with calibration
UVIMG	Grid UV data into an "image"
UVLIN	Fits and removes continuum visibility spectrum, also can flag
UVLOD	Read export or FITS data from a tape or disk

UVLSD	least squares fit to channels and divides the uv data.
UVLSF	least squares fit to channels and subtracts from uv data.
UVMLN	edits data based on the rms of the spectrum
UVMOD	Modify UV database by adding a model incl spectral index
UVMTH	Averages one data set and applied it to another.
UVNOU	flags uv samples near the U,V axes to reduce interference
UVPLT	Plots data from a UV data base in multiple ways
UVPOL	modifies UV data to make complex image and beam
UVPRM	measures parameters from a UV data base
UVPRT	prints data from a UV data base with calibration
UVRANGE	Specify range of projected baselines
UVRFI	Mitigate RFI by Fourier transform or fitting the circle
UVRMS	computes statistics of a selected subset of a uv data set
UVSEN	Determine RMS sidelobe level and brightness sensitivity
UVSIM	Generate sample UV coverage given a user defined array layout
UVSIZE	specifies number of pixels on X and Y axes of a UV image
UVSRT	Sort a UV dataset into a specified order
UVSUB	Subtracts/divides a model from/into a uv data base
UVTAPER	Widths in U and V of gaussian weighting taper function
UVWAX	flags uv samples near the U,V axes to reduce interference
UVWTFN	Specify weighting function, Uniform or Natural
VBGLU	Glues together data from multiple passes thru the VLBA corr.
VBRFI	Plots spectral statistics of VLBA autocorrelation data for RFI
VECTOR	selects method of averaging UV data
VH2RL	Converts from linear to circular polarization and vice versa
VHDIF	find/apply V minus H linear polarization phase difference
VHMOD	Compute linear polarization point source model, applies to data
VLA	puts the list of VLA antennas in the current file on stack
VLBA	puts the list of VLBA antennas in the current file on stack
VLBIN	Task to read VLBI data from an NRAO/MPI MkII correlator
VLBRF	Plots spectral statistics of VLBA autocorrelation data for RFI
VPFLG	Resets flagging to all or all corss-hand whenever some are flagged
VPLOT	Plots uv data and model from CC file
VTEST	Measures velocity discrepancy across fields
WEIGHTIT	Controls modification of weights before gain/fringe solutions
WETHR	Plots selected contents of WX tables, flags data based on WX
WIPER	plots and edits data from a UV data base using the TV
WRTDISK	writes images / uv data w extensions to disk in FITS format
WRTPROCS	Procedures to simplify the reduction of VLBA data
WTMOD	modifies weights in a UV data set
WTUV	Specifies the weight to use for UV data outside UVRANGE
XYDIF	find/apply X minus Y linear polarization phase difference
XYMOD	Compute linear polarization point source model, applies to data
XYPHASE	X-Y phase differences
ZEROSP	Specify how to include zero spacing fluxes in FT of UV data

13.40 VERB

```
VERB
Type:  General type of POPS symbol
Use:   Verbs are the magic words which cause FORTRAN code to
       execute some function.  They are compiled into AIPS
       by the programmers and their meaning remains fixed
       at least until the programmers change their minds.
Grammar:   Verbs may be given either in compile mode or in
       regular execute mode.  In the former, their pointers
       are stored with the procedure and they are executed
       when the procedure is invoked.  In the latter, they
       are compiled with the other statements and parameters
       on the input line and then executed before a new
       input line is read.
Execution:  Verbs are executed when the line in which they
       appear is executed and are simply referenced by their
       name.  The syntax "GO verb_name" is converted by AIPS to
       "TPUT verb_name ; verb_name" which saves the adverbs of
       "verb_name" for a later TGET and then executes
       "verb_name".  The syntax "TASK = 'verb_name' ; GO" will
       not work.
************************************************************
```

ABOUT	displays lists and information on tasks, verbs, adverbs
ABS	returns absolute value of argument
ACOS	Returns arc cosine of argument (half-circle)
ACTNOISE	puts estimate of actual image uncertainty and zero in header
ADDBEAM	Inserts clean beam parameters in image header
ADDDISK	makes a computer's disks available to the current AIPS session
ALLDEST	Delete a group or all of a users data files
ALTDEF	Sets frequency vs velocity relationship into image header
ALTSWTCH	Switches between frequency and velocity in image header
APROPOS	displays all help 1-line summaries containing specified words
ASIN	Returns arc sine of argument (half-circle)
ATAN	Returns arc tangent of argument (half-circle)
ATAN2	Returns arc tangent of two arguments (full circle)
AVEOT	Advances tape to end-of-information point
AVFILE	Moves tape forward or back to end-of-file marks
AVMAP	Advance tape by one image (IBM-CV = obsolete tape file)
AXDEFINE	Define or modify an image axis description
BAMODIFY	edits characters in a line of a batch work file
BATCH	starts entry of commands into batch-job work file
BATCLEAR	removes all text from a batch work file
BATEDIT	starts an edit (replace, insert) session on a batch work file
BATLIST	lists the contents of a batch work file
BDF2AIPS	Read EVLA ASDM/BDF data into AIPS
BDFLIST	List contents of EVLA ASD data file
BY	gives increment to use in FOR loops in POPS language
CALDIR	lists calibrator source models available as AIPS FITS files
CAT2LOG	list one or more entries in the user's data directory: #2 file name
CAT3LOG	list one or more entries in the user's data directory: #3 file name
CAT4LOG	list one or more entries in the user's data directory: #4 file name
CAT5LOG	list one or more entries in the user's data directory: #5 file name
CATALOG	list one or more entries in the user's data directory
CATOLOG	list one or more entries in the user's data directory: out file name
CEIL	returns smallest integer greater than or equal the argument
CELGAL	switches header between celestial and galactic coordinates
CHAR	converts number to character string

CHARMULT	Changes the multiplication factor for TV characters
CHKNAME	Checks for existence of the specified image name
CLR2NAME	clears adverbs specifying the second input image
CLR3NAME	clears adverbs specifying the third input image
CLR4NAME	clears adverbs specifying the fourth input image
CLR5NAME	clears adverbs specifying the fifth input image
CLRMSG	deletes messages from the user's message file
CLRNAME	clears adverbs specifying the first input image
CLRONAME	clears adverbs specifying the first output image
CLRSTAT	remove any read or write status flags on a directory entry
CLRTEMP	clears the temporary literal area during a procedure
CODECIML	Convert between decimal and sexagesimal coordinate values
COODEFIN	Define or modify an image axis coordinate description
COPIXEL	Convert between physical and pixel coordinate values
COS	returns cosine of the argument in degrees
COSTAR	Verb to plot a symbol at given position on top of a TV image
COWINDOW	Set a window based on coordinates
CPUTIME	displays curren tcpu and real time usage of the AIPS task
CURBLINK	switch TV cursor between steady and blinking displays
CURVALUE	displays image intensities selected via the TV cursor
DAYNUMBR	finds day nuumber of an image or uv data set
DEFAULT	Verb-like sets adverbs for a task or verb to initial values
DELAY	Verb to pause AIPS for DETIME seconds
DELBOX	Verb to delete boxes with TV cursor & graphics display.
DFILEBOX	Verb to delete Clean boxes with TV cursor & write to file
DISMOUNT	disables a magnetic tape and dismounts it from the tape drive
DOCC	Adverb to control application to Clean component files
DOWNCASE	Convert a string to all lower case letters
DRAWBOX	Verb to draw Clean boxes on the display
DUMP	displays portions of the POPS symbol table in all formats
EGETHEAD	returns parameter value from image header and error code
EGETNAME	fills in input name adverbs by catalog slot number, w error
EHEX	converts decimal to extended hex
END	marks end of block (FOR, WHILE, IF) of POPS code
EPOCONV	Convert between J2000 and B1950 coordinates
EPOSWTCH	Switches between B1950 and J2000 coordinates in header
EVLA	puts the list of eVLA antennas in the current file on stack
EXIT	ends an AIPS batch or interactive session
EXP	returns the exponential of the argument
EXPLAIN	displays help + extended information describing a task/symbol
EXTDEST	deletes one or more extension files
EXTLIST	lists detailed information about contents of extension files
FILEBOX	Verb to reset Clean boxes with TV cursor & write to file
FILEZAP	Delete an external file
FLOOR	returns largest integer ¡= argument
FOR	starts an iterative sequence of operations in POPS language
FREESPAC	displays available disk space for AIPS in local system
GAMMASET	changes the gamma-correction exponent used in the TV OFM
GET2NAME	fills 2nd input image name parameters by catalog slot number
GET3NAME	fills 3rd input image name parameters by catalog slot number
GET4NAME	fills 4th input image name parameters by catalog slot number
GET5NAME	fills 5th input image name parameters by catalog slot number
GETDATE	Convert the current date and time to a string
GETHEAD	returns parameter value from image header
GETITIME	Verb to return the UV data integration time.
GETNAME	fills 1st input image name parameters by catalog slot number
GETONAME	fills 1st output image name parameters by catalog slot number
GETPOPSN	Verb to return the pops number on the stack
GETTHEAD	Returns keyword values from a table header, also column #, # rows

GETVERS	finds maximum version number of an extension file
GO	starts a task, detaching it from AIPS or AIPSB
GRBLINK	Verb which blinks 2 TV graphics planes
GRCLEAR	clears the contents of the specified TV graphics channels
GRDROP	deletes the specified gripe entry
GREAD	reads the colors of the specified TV graphics channel
GRINDEX	lists users and time of all gripe entries
GRIPE	enter a suggestion or bug report for the AIPS programmers
GRLIST	lists contents of specified gripe entry
GROFF	turns off specified TV graphics channelS
GRON	turns on specified TV graphics channelS
GWRITE	Sets the colors of the specified TV graphics channel
HELP	displays information on tasks, verbs, adverbs
HINOTE	adds user-generated lines to the history extension file
HITEXT	writes lines from history extension file to text file
HSA	puts the list of HSA antennas in the current file on stack
HUEWEDGE	Show a wedge on the TV suitable for TVHUEINT displays
IM2HEAD	displays the image 2 header contents to terminal, message file
IM2TV	Verb to convert pixel coordinates to TV pixels
IM3HEAD	displays the image 3 header contents to terminal, message file
IM4HEAD	displays the image 4 header contents to terminal, message file
IM5HEAD	displays the image 5 header contents to terminal, message file
IMCENTER	returns pixel position of sub-image centroid
IMDIST	determines spherical distance between two pixels
IMERASE	replaces an image portion of the TV screen with zeros
IMHEADER	displays the image header contents to terminal, message file
IMOHEAD	displays the output image header contents
IMPOS	displays celestial coordinates selected by the TV cursor
IMSTAT	Returns statistics of a sub-image
IMVAL	returns image intensity and coordinate at specified pixel
IMWEDGE	load step wedge of full range of image values to TV
IMXY	returns pixel coordinates selected by the TV cursor
INP	displays adverb values for task, verb, or proc - quick form
INPUTS	displays adverb values for task, verb, or proc - to msg file
JOBLIST	lists contents of a submitted and pending batch job
KLEENEX	ends an AIPS interactive session wiping the slate klean
LENGTH	returns length of string to last non-blank character
LN	returns the natural logarithm of the argument
LOG	returns the base-10 logarithm of the argument
M2CAT	displays images in the user's catalog directory for IN2DISK
M3CAT	displays images in the user's catalog directory for IN3DISK
M4CAT	displays images in the user's catalog directory for IN4DISK
M5CAT	displays images in the user's catalog directory for IN5DISK
MAX	returns the maximum of its two arguments
MAXFIT	returns pixel position and image intensity at a maximum
MCAT	lists images in the user's catalog directory on disk INDISK
MFITSET	gets adverbs for running IMFIT and JMFIT
MIN	returns the minimum of its two arguments
MOCAT	displays images in the user's catalog directory for OUTDISK
MOD	returns remainder after division of 1st argument by 2nd
MODULUS	returns square root of sum of squares of its two arguments
MOUNT	makes a tape drive available to user's AIPS and tasks
NAMEGET	fills 1st input image name parameters by default matching
OBITIMAG	Access to OBIT task Imager without self-cal or peeling
OBITMAP	Simplified access to OBIT task Imager
OBITPEEL	Access to OBIT task Imager with self-cal and peeling
OBITSCAL	Access to OBIT task Imager with self-cal, NOT peeling
OFFPSEUD	Verb which deactivates all pseudo-color displays
OFFSCROL	Verb which deactivates scroll of an image
OFFTRAN	Verb which restores transfer function to normal

OFFZOOM	Verb which returns the hardware IIS zoom to normal
OFMADJUS	interactive linear adjustment of current TV OFM lookup tables
OFMCOLOR	Verb to access many OFMs to enhance TV pseudo-color displays
OFMCONT	creates/modifies TV color OFMs with level or wedged contours
OFMDIR	lists names of the user's and system's OFM files from OFMFIL
OFMGET	loads TV OFMS from a system or user OFM save file
OFMLIST	lists the current TV OFM table(s) on the terminal or printer
OFMSAVE	saves the TV's current OFM lookup table in a text file
OFMSTRCH	interactive modification of current TV OFM lookup tables
OFMTWEAK	interactive modification of current TV OFM lookup tables
OFMZAP	deletes an OFM lookup table save file
OUTPUTS	displays adverb values returned from task, verb, or proc
PARALLEL	Verb to set or show degree of parallelism
PASSWORD	Verb to change the current password for the login user
PCAT	Verb to list entries in the user's catalog (no log file).
PLGET	gets the adverbs used to make a particular plot file
PRINT	Print the value of an expression
PRINTER	Verb to set or show the printer(s) used
PROC	Define a POPS procedure using procedure editor.
PROCEDUR	Define a POPS procedure using procedure editor
PRTHI	prints selected contents of the history extension file
PRTMSG	prints selected contents of the user's message file
PSEUDOVB	Declares a name to be a symbol of type pseudoverb
PUTHEAD	Verb to modify image header parameters.
PUTTHEAD	inserts a given value into a table keyword/value pair
PUTVALUE	Verb to store a pixel value at specified position
Q2HEADER	Verb to summarize the image 2 header: positions at center
Q3HEADER	Verb to summarize the image 3 header: positions at center
Q4HEADER	Verb to summarize the image 4 header: positions at center
Q5HEADER	Verb to summarize the image 5 header: positions at center
QGETVERS	finds maximum version number of an extension file quietly
QHEADER	Verb to summarize the image header: positions at center
QIMVAL	returns image intensity and coordinate at specified pixel
QINP	displays adverb values for task, verb, or proc - restart form
QOHEADER	Verb to summarize the output image header: center positions
QUEUES	Verb to list all submitted jobs in the job queue
RANDOM	Compute a random number from 0 to 1
READ	Read a value from the users terminal
REBOX	Verb to reset boxes with TV cursor & graphics display.
RECAT	Verb to compress the entries in a catalog file
REHEX	converts extended hex string to decimal
REMDISK	removes a computer's disks from the current AIPS session
REMOVIE	Verb to rerun a previously loaded (TVMOVIE) movie
RENAME	Rename a file (UV or Image)
RENUMBER	Verb to change the catalog number of an image.
REROAM	Verb to use previous roam image mode, then does roam
RESCALE	Verb to modify image scale factor and offset
RESTART	Verb to trim the message log file and restart AIPS
RESTORE	Read POPS memory file from a common area.
RETURN	Exit a procedure allowing a higher level proc to continue.
REVERSN	checks disk for presence of extension files
REWIND	Verb to rewind a tape
ROAM	Roam around an image too large for the display.
ROAMOFF	Verb to recover image from roam display in simple display mode
RUN	Pseudoverb to read an external RUN files into AIPS.
SAVDEST	Verb to destroy all save files of a user.
SAVE	Pseudoverb to save full POPS environment in named file
SCALAR	Declares a variable to be a scalar in a procedure
SCAT	lists scratch files in the user's catalog directory on all disks
SCRATCH	delete a procedure from the symbol table.

SCRDEST	Verb to destroy scratch files left by bombed tasks.
SETDEBUG	Verb to set the debug print and execution level
SETMAXAP	Examines/alters system parameter limiting dynamic pseudo-AP
SETSLICE	Set slice endpoints on the TV interactively
SG2RUN	Verb copies the K area to a text file suitable for RUN
SGDESTR	Verb-like to destroy named POPS environment save file
SGINDEX	Verb lists SAVE areas by name and time of last SAVE.
SGUGET	Gets another user's SAVE/GET file.
SGUINDEX	Verb lists SAVE areas by name and time of last SAVE another user
SHOW	Verblike to display the TELL adverbs of a task.
SIN	Compute the sine of a value
SIZEFILE	return file size plus estimate of IMAGR work file size
SPY	Verb to determine the execution status of all AIPS tasks
SQRT	Square root function
STALIN	revises history by deleting lines from history extension file
STARTTV	Restarts XAS (TV), MS, TK servers that have disappered
STQUEUE	Verb to list pending TELL operations
STRING	Declare a symbol to be a string variable in POPS
SUBMIT	Verb which submits a batch work file to the job queue
SUBSTR	Function verb to specify a portion of a STRING variable
SYSTEM	Verb to send a command to the operating system
T1VERB	Temporary verb for testing (also T2VERB...T9VERB)
TABGET	returns table entry for specified row, column and subscript.
TABPUT	replaces table entry for specified row, column and subscript.
TAN	Tangent function
TAPES	Verb to show the TAPES(s) available
TGET	Verb-like gets adverbs from last GO of a task
TGINDEX	Verb lists those tasks for which TGET will work.
TGUINDEX	Verb lists those tasks for which TUGET will work another user
THEN	Specified the action if an IF test is true
TK1SET	Verb to reset 1D gaussian fitting initial guess.
TKAGUESS	Verb to add a plot of the slice model guess directly on TEK
TKAMODEL	Verb to add slice model display directly on TEK
TKARESID	Verb to add slice model residuals directly on TEK
TKASLICE	Verb to add a slice display on TEK from slice file
TKERASE	Erase the graphics screen or window
TKGUESS	Verb to display slice model guess directly on TEK
TKMODEL	Verb to display slice model directly on TEK
TKPOS	Read a position from the graphics screen or window
TKRESID	Verb to display slice model residuals directly on TEK
TKSET	Verb to set 1D gaussian fitting initial guesses.
TKSLICE	Verb to display slice file directly on TEK
TKVAL	Verb to obtain value under cursor from a slice
TKXY	Verb to obtain pixel value under cursor
TO	Specifies upper limit of a FOR loop
TPHEAD	Verb to list image header from FITS or IBM-CV tape
TPUT	Verb-like puts adverbs from a task in file for TGETs
TUGET	Verb-like gets adverbs from last GO of a task from another user #
TV1SET	Verb to reset 1D gaussian fitting initial guess on TV plot.
TV2COLOR	Verb to activate a 2-color pseudo-color display on the TV
TV3COLOR	Verb to initiate 3-color display using 3 TV channels
TVACOMPS	Verb to add slice model components directly on TV graphics
TVAGUESS	Verb to re-plot slice model guess directly on TV graphics
TVAMODEL	Verb to add slice model display directly on TV graphics
TVANOT	Verb to load anotation to the TV image or graphics
TVARESID	Verb to add slice model residuals directly on TV graphics
TVASLICE	Verb to add a slice display on TV graphics from slice file
TVBANDW	Verb to activate three types of black and white TV enhancements
TVBLINK	Verb which blinks 2 TV planes, can do enhancement also

TVBOX	Verb to set boxes with TV cursor & graphics display.
TVCLEAR	Verb to clear image from TV channel(s)
TVCOMPS	Verb to display slice model components directly on TV graphics
TVCUBE	Verb to load a cube into tv channel(s) & run a movie
TVDATA	displays information about the current TV display
TVFACTS	displays information about the current TV display
TVFIDDLE	Verb enhances B/W or color TV image with zooms
TVGUESS	Verb to display slice model guess directly on TV graphics
TVHELIX	Verb to activate a helical hue-intensity TV pseudo-coloring
TVHUEINT	Verb to make hue/intensity display from 2 TV channels
TVILINE	Verb to draw a straight line on an image on the TV
TVINIT	Verb to return TV display to a virgin state
TVLABEL	Verb to label the (map) image on the TV
TVLAYOUT	Verb to label the holography image on the TV with panel layout
TVLINE	Verb to load a straight line to the TV image or graphics
TVLOD	Verb to load an image into a TV channel
TVLUT	Verb which modifies the transfer function of the image
TVMBLINK	Verb which blinks 2 TV planes either auto or manually
TVMLUT	Verb which modifies the transfer function of the image
TVMODEL	Verb to display slice model directly on TV graphics
TVMOVIE	Verb to load a cube into tv channel(s) & run a movie
TVNAME	Verb to fill image name of that under cursor
TVOFF	Verb which turns off TV channel(s).
TVON	Turns on one or all TV image planes
TVPHLAME	Verb to activate "flame-like" pseudo-color displays
TVPOS	Read a TV screen position using cursor
TVPSEUDO	Verb to activate three types of pseudo-color displays
TVRESID	Verb to display slice model residuals directly on TV graphics
TVRESTOR	Restarts XAS (TV), MS, TK servers that have disappered
TVROAM	Load up to 16 TV image planes and roam a subset thereof
TVSCROL	Shift position of image on the TV screen
TVSET	Verb to set slice Gaussian fitting initial guesses from TV plot
TVSLICE	Verb to display slice file directly on TV
TVSPLIT	Compare two TV image planes, showing halves
TVSTAR	Verb to plot star positions on top of a TV image
TVSTAT	Find the mean and RMS in a blotch region on the TV
TVTRANSF	Interactively alters the TV image plane transfer function
TVWEDGE	Show a linear wedge on the TV
TVWINDOW	Set a window on the TV with the cursor
TVWLABEL	Put a label on the wedge that you just put on the TV
TVZOOM	Activate the TV zoom
TYPE	Type the value of an expression
U2CAT	list a user's UV and scratch files on disk IN2DISK
U3CAT	list a user's UV and scratch files on disk IN3DISK
U4CAT	list a user's UV and scratch files on disk IN4DISK
U5CAT	list a user's UV and scratch files on disk IN5DISK
UCAT	list a user's UV and scratch files on disk INDISK
UNQUE	remove a given job from the job queue
UOCAT	list a user's UV and scratch files on disk OUTDISK
UPCASE	Convert a string to all upper case letters
USAVE	Pseudoverb to save full POPS environment in named file
VALUE	Convert a string to a numeric value
VERB	Declares a name to be a symbol of type verb
VGET	Verb-like gets adverbs from version task parameter save area
VGINDEX	Verb lists those tasks for which VGET will work.
VLA	puts the list of VLA antennas in the current file on stack
VLBA	puts the list of VLBA antennas in the current file on stack
VPUT	Verb-like puts adverbs from a task in files for VGETs
WAITTASK	halt AIPS until specified task is finished
WEDERASE	Load a wedge portion of the TV with zeros
XHELP	Accesses hypertext help system
ZAP	Delete a catalog entry and its extension files

13.41 VLA

ALBUS	Calculate ionospheric delay and Faraday rotation using ALBUS
APGPS	Apply GPS-derived ionospheric corrections
ASDMFILE	Full path to EVLA ASDM/BDF directory
BDF2AIPS	Read EVLA ASDM/BDF data into AIPS
BDFLIST	List contents of EVLA ASD data file
BPWAY	Determines channel-dependent relative weights
BREAK	procedure to TELL FILLM to break all current uv files, start new
CALDIR	lists calibrator source models available as AIPS FITS files
CALIN	specifies name of input disk file usually with calibration data
CLCOR	applies user-selected corrections to the calibration CL table
CONFIG	Configuration ID number within an EVLA ASDM/BDF data set
CVEL	shifts spectral-line UV data to a given velocity
DAYFX	Fixes day number problems left by FILLM
DFCOR	Applies ATMO differential corrections to the calibration CL table
DOOSRO	calibrating amplitude and phase, and imaging VLA data
DOVLAMP	Produces amp calibration file for phased-VLA VLBI data
FARAD	add ionospheric Faraday rotation to CL table
FEW	procedure to TELL FILLM to append incoming data to existing uv files
FILLM	reads VLA on-line/archive format uv data tapes (post Jan 88)
FIXAL	least squares fit aliasing function and remove
FLOPM	reverses the spectral order of UV data, can fix VLA error
FRMAP	Task to build a map using fringe rate spectra
FXALIAS	least squares fit aliasing function and remove
GCPLT	Plots gain curves from text files
GPSDL	Calculate ionospheric delay and Faraday rotation corrections
LDGPS	load GPS data from an ASCII file
MANY	procedure to TELL FILLM to start new uv files on each scan
MAPBM	Map VLA beam polarization
MORIF	Combines IFs or breaks spectral windows into multiple windows (IFs)
NCHAN	Number of spectral channels in each spectral window
NIF	Number of IFs (spectral windows) in a data set
PANEL	Convert HOLOG output to panel adjustment table
PCVEL	shifts spectral-line UV data to a given velocity: planet version
PEEK	fits pointing model function to output from the VLA
PIPEAIPS	calibrating amplitude and phase, and imaging VLA data
PRTSY	Task to print statistics from the SY table
QUIT	procedure to TELL FILLM to stop at the end of the current scan
REIFS	Breaks spectral windows into multiple spectral windows (IFs)
REWAY	computes weights based in rms in spectra
RFLAG	Flags data set based on time and freq rms in fringe visibilities
STOP	procedure to TELL FILLM to break all current uv files and stop
SYHIS	Task to clip, filter, smooth SY tables and plot histograms
SYPRT	Task to print values from the SY table
SYSOL	undoes and re-does nominal sensitivity application for Solar data
TECOR	Calculate ionospheric delay and Faraday rotation corrections
TELFLM	procedure to TELL real-time FILLM a new APARM(1) value
TLCAL	Converts JVLA telcal files to an SN file
TYAPL	undoes and re-does nominal sensitivity application
TYSMO	smooths and filters a calibration TY or SY table
USERLIST	Alphabetic and numeric list of VLA users, points to real list
VLABP	VLA antenna beam polarization correction for snapshot images
VLAMP	Makes ANTAB file for phased VLA used in VLBI observations
VLANT	applies VLA/EVLA antenna position corrections from OPs files

VLAPROCS	Procedures to simplify the reduction of VLBA data
VLARUN	Calibrating amplitude and phase, and imaging VLA data
VLATECR	Calculate ionospheric delay and Faraday rotation corrections

13.42 VLBI

ACCOR	Corrects cross amplitudes using auto correlation measurements
ACFIT	Determine antenna gains from autocorrelations
ACSCL	Corrects cross amplitudes using auto correlation measurements
AFILE	sorts and edits MkIII correlator A-file.
ALBUS	Calculate ionospheric delay and Faraday rotation using ALBUS
ALIAS	adverb to alias antenna numbers to one another
ANBPL	plots and prints uv data converted to antenna based values
ANCAL	Places antenna-based Tsys and gain corrections in CL table
ANTAB	Read amplitude calibration information into AIPS
ANTNAME	A list of antenna (station) names
APCAL	Apply TY and GC tables to generate an SN table
ASTROMET	Describes the process of astrometric/geodetic reduction in AIPS
BANDPOL	specifies polarizations of individual IFs
BLING	find residual rate and delay on individual baselines
BPASS5	Normalization control in BPASS for VLBARUN
BPSOLINT	Sets SOLINT in BPASS for VLBARUN
BSCAN	seeks best scan to use for phase cal, fringe search, ..
BSPRT	print BS tables
CALIN	specifies name of input disk file usually with calibration data
CALTASK	Controls which calibration task is used when by VLBARUN
CAPLT	plots closure amplitude and model from CC file
CHREFANT	Name of reference antenna
CL2HF	Convert CL table to HF table
CLCOR	applies user-selected corrections to the calibration CL table
CLOSE	plots closure phase or amplitude spectra averaged over closures
CLPLT	plots closure phase and model from CC file
CLVLB	Corrects CL table gains for pointing offsets in VLBI data
CRSFRING	Procedure to calibrate cross pol. delay and phase offsets
CVEL	shifts spectral-line UV data to a given velocity
CXPOLN	Procedure to make complex polarization images and beam
DFCOR	Applies ATMO differential corrections to the calibration CL table
DOCLIP	Controls whether and how a CLIP operation is performed
DOVLAMP	Produces amp calibration file for phased-VLA VLBI data
DTSIM	Generate fake UV data
EDITR	Interactive baseline-oriented visibility editor using the TV
ELEVLIM	Lower limit on elevation
EOPSFILE	Name of text file containing EOP information
FRING	fringe fit data to determine antenna calibration, delay, rate
FRMAP	Task to build a map using fringe rate spectra
FRPLT	Task to plot fringe rate spectra
FXAVG	Procedure to enable VLBA delay de-correlation corrections
FXPOL	Corrects VLBA polarization assignments
HF2SV	convert HF tables from FRING/MBDLY to form used by Calc/Solve
HFPRT	write HF tables from CL2HF
HLPEDIPC	Interactive PC table editor PCEDT - run-time help
HLPEDIPD	Interactive PD table editor PDEDT - run-time help
HLPIBLED	Interactive Baseline based visibility Editor - run-time help
HLPSCIMG	Full-featured image plus self-cal loops, editing - run-time help
HLPSCMAP	Imaging plus self-cal and editing SCMAP - run-time help
IANTB	Writes out GC and TY data in ANTAB format
IBLED	Interactive BaseLine based visibility EDitor
KRING	fringe fit data to determine antenna calibration, delay, rate

M3TAR	translate Haystack MKIII VLBI format "A" TAR's into AIPS
MATCH	changes antenna, source, FQ numbers to match a data set
MBDLY	Fits multiband delays from IF phases, updates SN table
MERGECAL	Procedure to merge calibration records after concatenation
MK3IN	translate Haystack MKIII VLBI format "A" tapes into AIPS
MK3TX	extract text files from a MKIII VLBI archive tape
OBPLT	Plot columns of an OB table.
OOFRING	Fringe fit data to determine calibration with spectral index options
PCABL	Copies the Cable cal values from PC table 1 to PC table n
PCASS	Finds amplitude bandpass shape from pulse-cal table data
PCAVG	Averages pulse-cal (PC) tables over time
PCCOR	Corrects phases using PCAL tones data from PC table
PCDIV	Scales pulse-cal amplitudes by Tsys or Psum
PCEDT	Interactive TV task to edit pulse-cal (PC) tables
PCFIT	Finds delays and phases using a pulse-cal (PC) table
PCFLG	interactive flagging of Pulse-cal data in channel-TB using the TV
PCHIS	Generates a histogram plot file from text input, e.g. from PCRMS
PCLOD	Reads ascii file containing pulse-cal info to PC table.
PCPLT	Plots pulse-cal tables in 2 dimensions as function of time
PCRMS	Finds statistics of a pulse-cal table; flags bad times and channels
PCVEL	shifts spectral-line UV data to a given velocity: planet version
PHSLIMIT	gives a phase value in degrees
PLRFI	Plots spectral statistics from output of VBRFI & VLBRF
PRRFI	Prints excess spectral statistics from output of VBRFI & VLBRF
RESEQ	Renumber antennas
RLDLY	fringe fit data to determine antenna R-L delay difference
SEARCH	Ordered list of antennas for fring searches
SELFCAL	Requests self-calibration of calibration sources
SMRATES	Requests smoothing and clipping of rates after FRING in pipeline
SNEDT	Interactive SN/CL/TY/SY table editor using the TV
TAUO	Opacities by antenna number
TECOR	Calculate ionospheric delay and Faraday rotation corrections
TFILE	sorts and edits MkIII correlator UNIX-based A-file.
TIME2	used by VLBAMPCL as the time range for a second calibrator scan
TIMETOL	used by MERGECAL to control accuracy when comparing times
TRECVR	Receiver temperatures by polarization and antenna
TYSMOLEV	Clip level to use in running TYSMO on a VLB TY table
UVPOL	modifies UV data to make complex image and beam
VBCAL	Scale visibility amplitudes by antenna based constants
VBGLU	Glues together data from multiple passes thru the VLBA corr.
VBMRG	Merge VLBI data, eliminate duplicate correlations
VBRFI	Plots spectral statistics of VLBA autocorrelation data for RFI
VLAMP	Makes ANTAB file for phased VLA used in VLBI observations
VLASUMM	Plots selected contents of SN or CL files
VLBAAMP	applies a-priori amplitude corrections to VLBA data
VLBAARCH	Procedure to archive VLBA correlator data
VLBACALA	applies a-priori amplitude corrections to VLBA data
VLBACCOR	applies a-priori amplitude corrections to VLBA data
VLBACRPL	Plots crosscorrelations
VLBAEOPS	Corrects Earth orientation parameters
VLBAFIX	Procedure that fixes VLBA data, if necessary
VLBAFPOL	Checks and corrects polarization labels for VLBA data
VLBAFQS	Copies different FQIDS to separate files
VLBAFRGP	Fringe fit phase referenced data and apply calibration
VLBAFRNG	Fringe fit data and apply calibration
VLBAKRGP	Fringe fit phase referenced data and apply calibration
VLBAKRNG	Fringe fit data and apply calibration
VLBALOAD	Loads VLBA data
VLBAMCAL	Merges redundant calibration data

VLBAMPCL	Calculates and applies manual instrumental phase calibration
VLBAMPHC	applies VLBARUN calibration to additional phase stopping centers
VLBAPANG	Corrects for parallactic angle
VLBAPCOR	Calculates and applies instrumental phase calibration
VLBARFI	Calibrates VLBA autocorrelations, writes out statistics
VLBAPIPE	Applies amplitude and phase calibration procs to VLBA data
VLBARUN	Applies amplitude and phase calibration procs to VLBA data
VLBASNPL	Plots selected contents of SN or CL files
VLBASRT	Sorts VLBA data, if necessary
VLBASUBS	looks for subarrays in VLBA data
VLBASUMM	Prints a summary of a VLBI experiment
VLBATECR	Calculate ionospheric delay and Faraday rotation corrections
VLBAUTIL	Procedures to simplify the reduction of VLBA data
VLBAWIDE	Calibrates all VLA wide-band autocorrelations with VLBARFI
VLBCD	Reads ascii file containing VLBA calibration info into a CD table
VLBIN	Task to read VLBI data from an NRAO/MPI MkII correlator
VLBINPRM	Control parameters to read data from NRAO/MPI MkII correlators
VLBRF	Plots spectral statistics of VLBA autocorrelation data for RFI
VLBSY	Reads ascii file containing VLBA total power info into an SY table
VLOG	Pre-process external VLBA calibration files
VPLOT	Plots uv data and model from CC file
WEIGHTIT	Controls modification of weights before gain/fringe solutions
WXLOD	Reads KEYIN-format weather information making WX table

13.43 Additional recipes

13.43.1 Sautéd sole tobago with bananas, pecans and lime

1. Preheat 1/2 cup **vegetable oil** in a heavy sauce pan over medium-high heat.
2. Dredge 8 filets of **sole** or **flounder** lightly in **flour**.
3. Sautée until golden brown, about 3 minutes each side. Remove to warm platter.
4. Pour off excess oil and wipe down sauce pan. Place pan back on stove over high head; add 1/4 cup **butter**.
5. When foamy and just starting to brown, add 2 cups diagonally sliced **bananas** (1/2" slices) and 1 cup **pecan** halves. Toss and cook for 1 minute.
6. Add 1/2 cup fresh **lime juice** and 1 cup dry white **wine** (or light stock) . Cook for another 2 minutes.
7. Add 1/4 cup **fresh herbs** (mint, parsley, coriander, basil or tarragon).
8. Pour sauce and bananas over fish. Garnish with additional banana slices and lime wedges.

Thanks to Turbana Corporation (`www.turbana.com`).

13.43.2 Banana Bombay salad

1. Puree 3 **bananas**.
2. Whisk with 1/4 cup **lemon juice**, 1/4 cup **mayonnaise**, 1/4 cup **plain yogurt**, and 1/8 – 1/4 ounce **taragon**. Refrigerate at least 2 hours.
3. Cut 2 pounds cooked **turkey** or **chicken breast** into bitesize pieces.
4. Add 1/2 cup **raisins**, 3 **green apples** cut into pieces, and 1/2 cup chopped **walnuts**. Mix.
5. Add banana puree and mix. Cut 2 **bananas** into thick chunks and add. Serve chilled.

Thanks to Chiquita Bananas.

13.43.3 Banana mandarin cheese pie

1. In large mixer bowl, beat 8 ounces softened **cream cheese** until fluffy.
2. Gradually beat in 8 ounces **sweetened condensed milk** until smooth.
3. Stir in 1 teaspoon **lemon juice** and 1 teaspoon **vanilla extract**.
4. Slice 2 medium **bananas**, dip in lemon juice, and drain.
5. Line 8(?)-inch **graham cracker pie crust** with bananas and about 2/3 of an 11-ounce can (drained) **mandarin oranges**.
6. Pour filling over fruit and chill for 3 hours or until set.
7. Garnish top with remaining orange segments and 1 medium **banana** sliced and dipped in lemon juice.

13.43.4 Cream of banana soup

1. Cook 1 quart green **banana** pulp, 1 1/2 quarts **chicken stock**, 1 small **celery stalk**, 1/2 **onion**, 1 **carrot**, 1 small **bay leaf**, 5 **peppercorns**, and **salt** to taste together for about 30 minutes until the mixture thickens.
2. Strain over 1/4 cup **flour** and 1/4 cup **butter** which have been combined as for a white sauce. Cook until thickened.
3. Just before serving, add 2 cups **cream** or **milk** and heat.
4. Serve with a slice of lemon on each plate as a garnish.

13.43.5 Banana relish

1. Cut 12 **bananas**, 1 pound **dates**, and 2 pounds **Bermuda onions** into small pieces.
2. Add 2/3 cup **molasses**, 1/2 teaspoon ground **ginger**, 1 teaspoon **salt**, 1 teaspoon **allspice**, 1 cup **water**, and 2 cups **vinegar**; mix well.
3. Turn into a large stone jar or crock, bake in a slow oven till rich brown, seal in jars while hot.

13.43.6 Banana coffeelate

1. Peel and mash 2 ripe **bananas**.
2. Blend in 1/2 teaspoon **vanilla extract**, a few grains **salt**, 1/4 cup **chocolate syrup**, 2 teaspoons **sugar**, and 2 teaspoons instant powdered **coffee**.
3. Add $1\frac{1}{2}$ cups **milk**.
4. Beat with rotary beater or electric mixer until smooth and creamy. Chill.

13.43.7 Banana pick-me-up

1. Slice ripe, peeled **bananas** into 3 cm chunks.
2. Wrap each chunk in strip blanched **bacon**.
3. Prepare mixture of **brown sugar** and **cinnamon** to taste.
4. Sprinkle mixture over banana chunks.
5. Bake at 350° F until the bacon is crisp and the sugar slightly caramelized.

A Easy Continuum UV-Data Calibration and Imaging

This appendix contains a step-by-step guide to calibrating simple continuum data from interferometers using $\mathcal{AIPS}$. This older guide is now preceded with information about a new "pipeline" of $\mathcal{AIPS}$ procedures which can *automagically* calibrate, image, and even self-calibrate many multi-source continuum (and line) data sets. Although this pipeline is unlikely to produce immediately publishable results in most cases, it does produce a data set with preliminary calibration and editing which can then be edited further "by hand." Following the additional editing, the pipeline may be re-run to produce improved images. When good single-source data sets are produced, self-calibration can also be performed using the methods described in § 5.4.

A.1 VLARUN

The user should read $\mathcal{AIPS}$ Memo 112, "Capabilities of the VLA pipeline in AIPS," by Lorant O. Sjouwerman dated March 19, 2007. This memo goes into details and advice that are beyond the scope of this appendix. The memo may be obtained from the $\mathcal{AIPS}$ web set `https://www.aips.nrao.edu/`.

To use the pipeline, you must first load the VLA Archive or other multi-source visibility data into $\mathcal{AIPS}$; see § O.1.1. To acquire the pipeline, including all procedures and special adverbs, enter

> RUN VLARUN ↵ — to define the procedures and adverbs

Do this only once. The procedures will be remembered in your `LASTEXIT` `SAVE`/`GET` file.

The pipeline has three stages: (1) calibration and editing, (2) basic imaging, and (3) self-calibration. Study the recommendations and other advice in `EXPLAIN VLARUN` ↵. Note that `VLARUN` can also handle spectral-line data. Then, the inputs for the calibration stage include

> TASK 'VLARUN' ; INP ↵ — to review the inputs needed.

> WORKDISK n ↵ — to specify the disk containing the uv data.

> CATNUM $catn$ ↵ — to specify the catalog number of the uv data; `INNAME` et al. may also be used.

> FASTSW 1 ↵ — to correct source name peculiarities induced by fast-switching observations.

> AUTOFLAG 2 ↵ — to use `FLAGR` on all data and `QUACK` on all but high frequency data sets

> PHAINT 1 ; AMPINT 5 ↵ — to set the phase calibration interval to 1 minute and the amplitude calibration interval to 5 minutes.

> DOMODEL 1 ↵ — to use standard flux calibrator models where available.

> NOPAUSE -1 ↵ — to have the pipeline pause after `GETJY` to allow you to evaluate whether it is okay to proceed.

> AUTOPLOT -1 ↵ — to make no diagnostic plot files.

> DOIMAGES -1 ↵ — to do calibration and editing only.

> INP ; TPUT ↵ — to double check the inputs and save them.

> VLARUN ↵ — to calibrate and edit your data.

The pipeline can do imaging and even self-calibration, although the latter is not for the faint at heart. To

include this in your pipeline run, enter

> DOIMAGES 1 ↵	to request imaging.
> ARRYSIZE 0 ↵	to let the procedure find the array dimensions.
> IMSIZE -1 ↵	to image the target source over the full primary beam, with smaller areas for calibrators.
> NITER *nit* ; CUTOFF *f* ↵	to set the number of iterations *nit* and Clean flux limit *f*. NITER must be > 0 for the image to be Cleaned. Use conservative values to begin with and higher limits in subsequent runs.
> ALLIMG 1 ↵	to make images of calibrators as well as target sources.
> SLFCAL 0 ↵	to avoid self-calibration initially.
> INP ; TPUT ↵	to double check the inputs and save them.
> VLARUN ↵	to calibrate, edit, and image your data.

In the first run of VLARUN, it is recommended to avoid self-calibration although the making of images will help you evaluate the quality of the auto-editing. If your data are not "perfect" (*i.e.*, good enough), then consider one or more of the interactive editing tasks such as TVFLG (§ O.1.6), SPFLG (§ 10.2.2), EDITA (§ 4.3.11), and EDITR (§ 5.5.2) along with the advice given on editing in § 4.3.5, § 4.3.13, § O.1.5, and § 5.5. Then

> TGET VLARUN ↵	to restore the inputs.
> AUTOFLAG -1 ↵	to do no more automatic flagging.

Consider raising NITER and/or lowering CUTOFF in this second pass, plus

> SLFCAL *snit* ↵	to also do self-calibration with interactivity on the TV ($snit > 0$) or in a batch-like process ($snit < 0$) for $\|snit\|$ self-cal iterations.
> INP ; TPUT ↵	to double check the inputs and save them.
> VLARUN ↵	to calibrate, edit, and image your data.

Repeat as needed. Note that VLARUN has been used on many of the data sets in the VLA archive, with the results available via `https://www.vla.nrao.edu/astro/nvas/`. These results include both images and calibrated *uv* data sets.

A.2 Basic calibration

From VLA Archive Tape to a UV FITS Tape
AIPS Memo No. 76 Updated
Glen Langston

The Gentle User enters the Computer room with a VLA archive tape containing a scientific breakthrough. The user's sources are named SOURCE1 and SOURCE2. The interferometer phase is calibrated by observations of CAL1 and CAL2. The flux density scale is calibrated by observing 3C48 (=0137+331) and polarization is calibrated with observations of 3C286 and/or 3C138. Mount the tape on drive number *n*, log in and start *AIPS*. Example input: AIPS NEW. Mount the tape: INTAPE=*n*; DENS=6250; MOUNT.

PRTTP Find out what is on the tape, get project number and bands. TASK='PRTTP'; PRTLEV=-2; NFILES=0; INP; GO; WAIT; REWIND.

FILLM Load your data from tape. Select only one band at a time to process. TASK='FILLM'; VLAOBS='?'; BAND=' '; NFILES=?; DOWEIGHT 1; INP; GO (Replace all ?'s with appropriate values.) FILLM will load your visibilities (*uv*-data) with weights suitable for calibration into a large file for each band. It also creates 6 *AIPS* tables each; these tables have two letter names which are:

HI Human readable history of things done to your data. Use PRTHI to read it.

AN Antenna location and polarization tables. Antenna polarization calibration is placed here.

NX Index into visibility file based source name and observation time. Not modified by calibration.

SU Source table contains the list of sources observed and indexes into the frequency table. The flux densities of the calibration sources are entered into this table.

FQ Frequencies of observation and bandwidth with index into visibility data. Not modified.

CL Calibration table describing the antenna based gains. Version 1 should never be modified. The CL table contains entries at regular time intervals (*i.e.*, 2 minutes) for each antenna. **The ultimate goal of calibration is to create a good** CL **version 2.** Use PRTAB to read tables.

PRTAN Print out the antenna locations. TASK='PRTAN'; PRTLEV=0; INP; GO. Choose a good *Reference* antenna (called *R*) near the center of the array (REFANT=R). Check the VLA operator log to make sure the antenna was OK during the entire observation.

QUACK Flag the bad points at the beginning of each scan, even the ones with good amplitudes could have bad phases. Creates a Flag Table (FG). You want to use FG table version 1 for all tasks. TASK='QUACK'; FLAGV=1; OPCOD=' '; APARM=0; SOUR=' '; INP; GO deletes the first six seconds of each scan, which may not be enough.

FG A flag table marks bad data. FG tables contain an index into the UV data based on time range, antenna number, frequency and IF number.

LISTR Lists your UV data in a variety of ways. Make a list of your observations. TASK='LISTR'; OPTYP='SCAN'; DOCRT=-1; SOUR=' '; CALC='*'; TIMER=0; INP; GO. NOTE: IF you have observed in a way as to create more than one FREQID, you must run through the entire calibration once for EACH FREQID. For new users, it is better to use UVCOP to copy each FREQID into separate files and calibrate each file separately. This is required if you are doing polarization calibration.

UVCOP Skip this step if your data consists of only one FREQID. Copy different FREQIDs into separate files. TASK='UVCOP'; FREQID=?; CLRON; OUTDI=INDI; INP; GO. The result will be a ??.UVCOP file.

SETJY Sets the flux of your flux calibration source in the SU table. TASK='SETJY'; SOUR='3C48',' '; OPTYP='CALC'; FREQID=1; INP; GO. Adjust flux density for partial resolution following the rules in the VLA Calibration Source Manual or the *AIPS CookBook*.

TASAV As insurance, make a copy of all your tables. TASK='TASAV'; CLRON; OUTDI=INDI; INP; GO.

CALIB CALIB is the heart of the *AIPS* calibration package. RUN VLAPROCS, an *AIPS runfile*, to create procedures VLACALIB, VLACLCAL and VLARESET. The procedure VLACALIB runs CALIB. Set UVRANGE and ANTENNA to zero to allow use of models for 3C48. For L, C and X band 5% and 5 degree errors are OK; for other bands the limits are higher. CALIB places antenna amplitude and phase corrections into an SN table for the time of observation of phase calibration sources.

SN Solution table contains antenna based amplitude and phase corrections for the time of observations of the calibration sources. These SN table results are latter interpolated for all times of observation and placed in a CL table. Only the CL table corrections will be applied to the program sources.

TASK='VLACAL'; CALS='3C48',' '; CALCODE='*'; REFANT=*R*; UVRA=0; SNVER=1; DOCALIB=-1; DOPRINT=1; MINAMP=10; MINPH=10; INP; VLACAL. VLACALIB will load and use a source model for 3C48, making UVRANGE unnecessary. The task CALIB lists antenna pairs which deviate significantly from the solution. If you have lots of errors, then carefully examine your data using TVFLG or LISTR. (See *AIPS CookBook* for a lengthy discussion on flagging.) If one antenna is bad over a limited time range, use UVFLG to flag that antenna for the time from just after the previous good CAL observation to before the next good CAL observation.

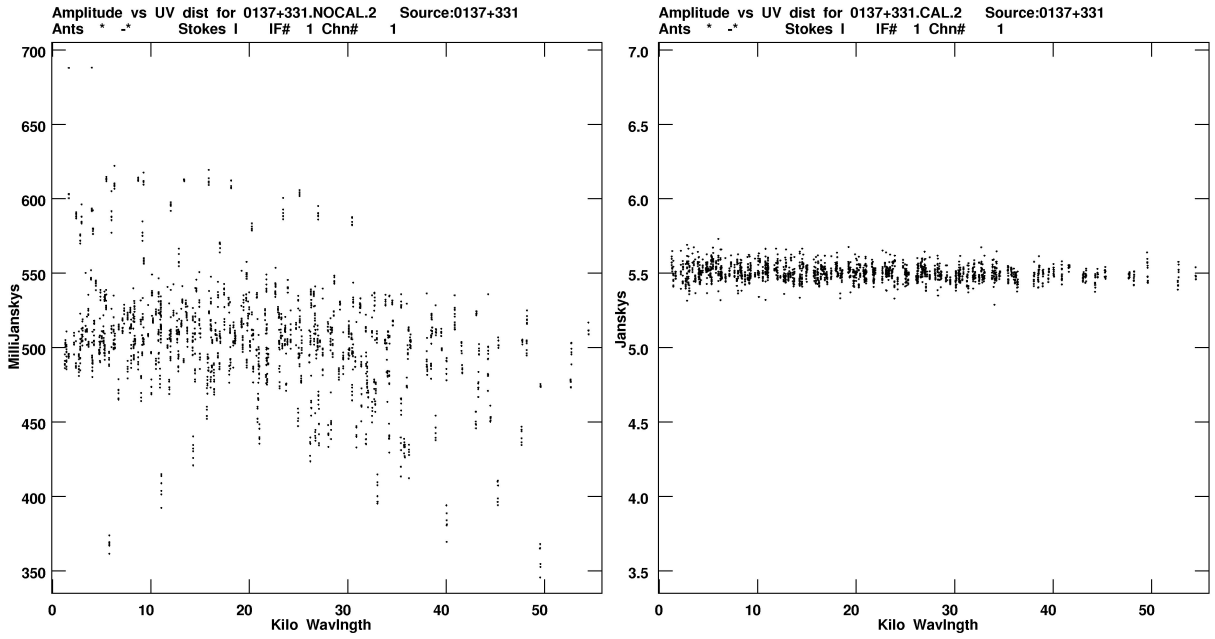


Figure A.1: *(left)* Un-calibrated *uv*-data and *(right)* calibrated *uv*-data from a C-band snapshot of 3C48. Default VLA gains are a tenth of the actual gains and can show significant scatter. Only wild *uv* points ∼50% greater than the average can be detected before calibration.

UVFLG Flag bad UV-data. TASK='UVFLG'; ANTEN=?,0; BASELI=?,0; TIMER=?; OUTFGVER=1; SOUR=' '; OPCOD=' '; INP; GO. If in doubt about any data, FLAG THEM! If you have flagged the primary calibrator, return to CALIB above and try again.

CALIB Now calibrate the antenna gain based on the rest of the cal sources. Look in the Calibrator manual for UV limits; if there are limits, VLACAL must be run separately for these sources. TGET VLACAL; CALS='CAL1','CAL2',' '; ANTEN=0; BASELI=0; UVRANGE=?,?; INP; VLACAL. Flag bad antennas listed. Each execution of CALIB replaces previous corrections in the SN table or appends new corrections. If unsatisfied with a VLACAL execution, all effects of it are removed by running VLACAL again for the same sources (but different ADVERBS or after flagging bad data).

GETJY Sets the flux of phase calibration sources in the SU table. TASK 'GETJY'; SOUR='CAL1,'CAL2',' '; CALS='3C48',' '; BIF=0; EIF=0; INP; GO. GETJY over-writes existing SU table entries, and is not affected by previous executions.

TASAV Good time to save your tables. TGET TASAV; INP; GO.

CLCAL Read the antenna amplitude and phase corrections from the SN table and interpolate the corrections into a new CL table. CLCAL applies calibration source corrections to the program sources. Each execution of CLCAL adds to output CL table version 2. CLCAL is run using the procedure VLACLCAL. TASK='VLACLC'; SOUR='SOURCE1','CAL1',' '; CALS='CAL1',' '; OPCODE='CALI'; TIMER=0; INTERP='2PT'; INP; VLACLC. Run CLCAL for the second source using the second calibrator. TGET VLACLC; SOUR='SOURCE2','CAL2',' '; CALS='CAL2',' '; INP; VLACLC. Move the SN table corrections for 3C48 into the CL table. TGET VLACLC; SOUR='3C48',' ';CALS='3C48',' '; INP; VLACLC. (3C48 could also be calibrated with CAL1 or CAL2.)

LISTR Make a matrix listing of the Amplitude and RMS of calibration sources with calibration

applied. Look for wild points. TASK='LISTR'; OPTYP='MATX'; SOUR='CAL1','CAL2',' '; DOCAL=2; DOCRT=-1; DPARM=3,1,0; UVRA=0; ANTEN=0; BASELI=0; BIF=1; INP; GO. If only a few points are bad, flag them and continue. If too many are bad, delete CL table 2 and the SN tables using VLARESET. Then return to the first CALIB step. If the data look good, run LISTR again for IF two. TGET LISTR; BIF=2; INP; GO

UVPLT Plot the *uv*-data in a variety of ways. Make a Flux versus Time plot first. Choose XINC so the plot will have no more than 1000 points. TASK='UVPLT'; SOUR='SOURCE1',' '; XINC=10; BPARM(1)=11; DOCAL=2; BIF=1; INP; GO. Look at the plot with LWPLA, TKPL, TVPL or TXPL. Plot other IF . Flag wild points. Plot Flux versus baseline. TGET UVPLT; BPARM=0; INP; GO.

Calibration is now complete for continuum, un-polarized observations. Write the calibrated data to tape with FITTP if you don't want to calibrate the polarization. To create images from the *uv*-data use SPLIT to calibrate the multi-source data and create a single source *uv*-data set. (FITTP and SPLIT are described at the end of the polarization calibration process.)

A.3 Polarization calibration

For polarization observations, the following steps are required. For 21cm or longer wavelength observations, ionospheric Faraday rotation corrections may be needed. See FARAD in the *AIPS CookBook*, but don't expect much help anymore.

TASAV As added insurance, save your tables again. TGET TASAV; INP; GO.

LISTR Print the parallactic angles of the calibration sources. TGET LISTR; SOUR=' '; CALC='*'; OPTYP='GAIN'; DPARM=9,0; INP; GO

PCAL Intrinsic antenna polarization calculation. PCAL will be successful only if cal. sources are observed at several parallactic angles. PCAL will modify the AN and SU tables. TASK='PCAL'; CALS='CAL1','CAL2',' '; BIF=1; EIF=2; DOCAL=2; REFANT=*R*; INP; GO

RLDIF Now determine the absolute linear polarization angle. Make a matrix listing of the angle of 3C286. TASK 'RLDIF'; SOUR='3C286',' '; DOCAL=2; BIF=1; EIF=0; DOPOL=1; GAINUSE=2; DOCRT=-1; INP; GO. The observed angles are different for each frequency and IF . This task returns the average angles for all IFs in CLCORPRM.

CLCOR Now apply the angle corrections to CL table 2. The relative phase of Left and Right circular polarization produces the linear polarization angle and the phase correction is applied to L. The phase difference (twice the angle of linear polarization) for 3C286 is 66^o and for 3C138, $\phi = -18^o$ at L band, perhaps -24^o at higher frequencies. Change RLDIF's results to this form with FOR I = 1:20; CLCORP(I)=66-CLCORP(I); END. Then TASK='CLCOR'; STOKES='L'; SOUR=' '; OPCOD='POLR'; BIF=1; EIF=2; GAINVER=2; GAINUSE=0; INP; GO. Run RLDIF again to check the phases. TGET RLDIF; INP; GO. Note that CLCOR copies the CL table version 2 to 3 while applying the phase correction. If the phases are wrong, delete version 3, return to PCAL and RLDIF and then do another CLCOR.

A.4 Backup and imaging

FITTP Writes the output *uv*-data to tape. DISMOUNT your archive; MOUNT your output tape. TASK='FITTP'; DOEOT=1; OUTTAP=INTAP; INP; GO. Use DOEOT=-1 when at the beginning of a new tape.

SPLIT The *AIPS* calibration process only modifies the tables associated with the multi-source *uv*-data set. SPLIT selects individual sources, reads the CL table and multiplies the visibilities by the corrections to produce a calibrated single-source *uv*-data set. TASK='SPLIT'; SOUR=' '; CALC=' '; UVRA=0; TIMER=0; DOCAL=2; FLAGVER=1; GAINUSE=3; DOPOL=1; DOBAN=-1; BIF=0; EIF=0; STOKES=' '; BLVER=-1; APARM=0; DOUVCOM=1; ICHANSEL=0; INP; GO

Mapping Use your favorite Fourier Transform task (*e.g.*, IMAGR) to produce images from the calibrated data. Procedure MAPPR provides a simplified interface to IMAGR.

A.5 Additional recipes

A.5.1 Banana colada

1. Peel and slice 1 ripe **banana**.
2. Place sliced banana in blender along with 6 ounces **pineapple juice** (or crushed tinned pineapple in its own juice) and 1 ounce **rum** plus 1 ounce **coconut rum** *or* 2 ounce **rum** plus 1 teaspoon **Coco Lopez**.
3. Optionally add 1 ounce **banana liqueur**.
4. Blend until smooth.
5. Add crushed ice, if so desired.
6. If the mixture is too thick, add more juice (or more rum if you prefer!); if too thin, add more banana. This is a really easy recipe to adjust to one's taste.

A.5.2 Breaded chicken and bananas

1. In food processor, blend 1 can **condensed milk**, 1/3 cup **milk**, 1/2 cup flaked **coconut**, and 1/4 cup **lemon juice** until smooth. Pour into a bowl.
2. Prepare 3 cups **corn flake crumbs** in another bowl or plate.
3. Cut 6 very firm **bananas** lengthwise, dip in milk mixture, roll in corn flakes, and set aside.
4. Cut 2 **chickens** into pieces, dip in milk mixture, roll in corn flakes, and place in greased baking pans (2 13x9 pans may be required).
5. Sprinkle chicken with 1/2 cup melted **butter** and bake as 350° F for one hour.
6. Arrange bananas over the chicken. Sprinkle with 1/4 cup melted **butter**. Bake 15 minutes longer or until chicken juices run clear.
7. Garnish with sliced star and/or kiwi fruits if desired.

Thanks to Turbana Corporation (`www.turbana.com`).

A.5.3 Banana-pineapple bread

1. Mix together 1 cup chopped **nuts**, 2-1/2 cups **sugar**, 5 cups **flour**, 1 teaspoon **salt**, 1 teaspoon **baking powder**, and 1 teaspoon **cinnamon**.
2. Mix together 1-1/2 cups **vegetable oil**, 3 **eggs**, 3 mashed **bananas**, 1 teaspoon **lemon juice**, and 1 can **crushed pineapple** (drained).
3. Combine. Bake at 350° F for one hour.

Thanks to Tim D. Culey, Baton Rouge, La. (`tsculey@bigfoot.com`).

B A Step-by-Step Guide to Spectral-Line Data Analysis in $\mathcal{AIPS}$

Initially contributed by Andrea L. Cox and Daniel Puche

In this guide, we assume that the reader is familiar with the basic tools of $\mathcal{AIPS}$; *e.g.*, MCAT (§ 3.3), GETN (§ 3.3.1), IMHEAD (§ 3.3.4) and other $\mathcal{AIPS}$ tools involving the manipulation of the data catalog are not mentioned. This guide contains three main sections covering editing and calibration of spectral-line data, making and Cleaning of map cubes, and moment analysis and rotation curves of galaxies. *It is assumed through these sections that all sources in the data set were observed at the same frequency; the final section of this guide describes what you should do before beginning data reduction if this is not the case.*

This is an outline of a typical reduction procedure for spectral-line data from the VLA; different users may use slightly different approaches. This guide is a supplement to the $\mathcal{AIPS}$ EXPLAIN and ABOUT verbs (§ 3.8) and the $\mathcal{AIPS}$ *CookBook*. Some of the less obvious or more important parameters for each task will be mentioned, but the user should *always* check to ensure that the rest of the parameters are specified correctly. When in doubt, the defaults are usually fairly safe. Words in boldface or typewriter fonts represent $\mathcal{AIPS}$ tasks and their inputs. When you see a phrase enclosed in brackets, replace the phrase and the brackets with the correct input. For example, to specify the source 0134+329

SOURCE '< *source_name* >' would be typed as

SOURCE '0134+329' The text below is in a three-column format, showing a step number on the left, a descriptive paragraph in the center, the name of the $\mathcal{AIPS}$ task or verb on the right.

B.1 Editing and calibrating spectral-line data

B.1.1 Loading the data

(1) Go to the VLA archive web page at

`https://data.nrao.edu/`

log in wih your `my.nrao.edu` account and locate the desired data set using the basic retrieval tool. Use the observing date, program code, or observer's name to locate the data. If the data are within the proprietary period (usually 6 months), you will be allowed access only if your login is recognized as an owner. Select the data file(s) desired and ask for them with $\mathcal{AIPS}$-friendly names. Wait for the e-mail notification that the files are ready and then follow the supplied advice to copy the data files to your computer.

(2) Before starting AIPS, create an environment variable pointing at the disk directory containing the data files. Use `cd` < *data_area* > followed by `setenv MYDATA` ‘`pwd`‘ if you use C-shell or its variants or `export MYDATA=`‘`pwd`‘ if you use the bash shell. (Note the back tick marks surrounding the `pwd`.) Then start AIPS.

(3) Load the data from the archive file(s) to *AIPS* data areas. Use the defaults to load the data without further averaging. The m disk files will be named $< program_name > _i$ for $i = n+1, \ldots, n+m$. **FILLM**

```
VLAOBS '< program_name >' ; DOWEIGHT 1
NFILES n ; NCOUNT m
DATAIN 'MYDATA:< program_name > _'
```

The data loaded to disk will normally be in two parts: One will have the class "CH0" and the other will have the class "LINE." The names of the files are thus

```
DATE.CH0
DATE.LINE
```

where "DATE" is the date the observations began. Each file is a multi-source file, containing observations for all your sources: flux calibrators, phase calibrators, and target sources. Your spectral-line data are contained in the LINE file, while the CH0 file is a "pseudo-continuum" file; it is the average of the inner 75% of the bandpass and will be used for gain and phase calibration.

(4) List the "scan summaries" from the CH0 data. *Keep the output for future reference.* Note that the frequency in the header of a multi-source file is always the sky frequency in the center of the band of the first scan of the observation (see § B.4). **LISTR**

```
OPTYPE 'SCAN' ; DOCRT -1 ; OUTPRINT ' '
```

to print on the line printer. To make a text file instead

```
OUTPRINT 'MYDATA:< filename_in_all_caps >'
```

(5) Print the antenna configuration file. *Keep the output for future reference.* **PRTAN**

B.1.2 Inspecting and editing the data

There are a number of different ways to isolate and edit bad *uv* points from your data set. The method described below is typical. Other tasks of interest can be found by typing ABOUT EDITING and ABOUT UV and by consulting § 4.3.5, § O.1.6, § 4.3.11, § 5.5.2, § 10.2.2, and § 4.3.13 in the *AIPS CookBook*.

(6) Plot amplitude versus baseline length for your flux and phase calibrators. Inspect each source, Stokes, and IF separately. Set XINC so there are only a few thousand visibilities on the output plot (the total number of visibilities is listed on the scan summary sheets from Step 4). Use the TV to save trees. **UVPLT**

```
BPARM 0 ; DOTV 1
```

If there are anomalous amplitude points, continue to the next step. If your data points have a small scatter, you may not need to edit and can skip to calibration (Step 10).

(7) Determine if the anomalous data points are from a particular baseline, antenna, Stokes, or IF, inspecting each Stokes and IF separately. The output of this task will be all points that have anomalous amplitudes, based on your selection criteria. **UVFND**

```
OPCODE 'CLIP' ; APARM(1) < max_flux >
APARM(3) < min_flux >
```

(8) Once you have determined which data points to flag with UVFND, flag them with UVFLG. You can flag by time-range, baseline, or antenna and you can flag any or all of the Stokes parameters or IFs. To assist in undoing flags, set a REASON. **UVFLG**

```
OPCODE 'FLAG' ; OUTFGVER 1
```

(9) Examine the *uv* data for your calibrators on the TV to check for any obvious problems which you might have missed; see § O.1.6. Check each IF and each Stokes separately and edit the data more carefully, if necessary, before continuing. **TVFLG**

```
DOCAL -1 ; CALCODE '*' ; FLAGVER 1
```

B.1.3 Calibrating the data

Steps 10–15 should be applied to the CH0 data alone, not to the LINE data. To ensure that all inputs are set to their defaults before continuing, type

```
RESTORE 0
```

Then, when you are satisfied with your editing, type

```
RUN VLAPROCS
```

to set up VLA-specific parameters and procedures for calibration. You may turn all adverbs for a specific $< task >$ to their default values with

```
DEFAULT < task >
```

If you have multi-frequency data, each frequency must be calibrated separately; this can be done by specifying the FREQID *parameter in each task (see § B.4).* More information on calibrating your data can be found by typing ABOUT CALIBRAT and HELP CALIBRAT and consulting Chapter 4.

(10) Calculate the flux of the primary flux calibrator for the channel zero (CH0) data. **SETJY**

```
SOURCE '< flux_calibrator >' , ' ' ; OPTYPE 'CALC
```

(11) Calculate gain and phase solutions for *all* of the calibrators. In this case, you must run this procedure once for each source. Check for the presence of an appropriate model with CALDIR. For those sources having a model **VLACALIB**

```
DOCALIB 1 ; UVRANGE 0 ; ANTENNAS 0
CALSOUR '< flux_calibrator_1 >' , ' '
```

For sources without a model, choose a UVRANGE according to the tables in the *VLA Calibration Manual*. You may do more than one source at a time, if they have the same UVRANGE. The output of this procedure is a solution (SN) extension table, which is printed automatically. Select a reference antenna (REFANT) which did not have any problems during the observing run and which is located near the center of the array.

```
DOCALIB 1 ; UVRANGE < uv_min > , < uv_max >
CALSOUR '< calibrator_1 >' , ... , '< calibrator_n >'
DOPRINT 1; OUTPRINT '< filename_in_all_caps >'
```

The output from VLACALIB will include a list of closure errors. If there are too many large errors, edit your data carefully using UVFND, TVFLG, or LISTR as described above. *Destroy old SN tables with* EXTDEST and then re-run VLACALIB until the solutions are satisfactory. The output will include amplitudes and phases for each baseline; for each calibrator, the amplitudes should be approximately constant and the phases should vary smoothly over time.

(12) Calculate the flux densities of the secondary (phase) calibrators from the primary (flux) calibrator, based upon the flux densities in the source (SU) table and the antenna gain solutions in the solution (SN) table. *Destroy bad or redundant versions of the* SN *tables before using this task.* Compare the computed fluxes with those listed in the *VLA Calibration Manual*. **GETJY**

```
SOURCES '< phase_cal_1 >' , ... , '< phase_cal_n >'
CALSOUR '< flux_cal >' , ' '
```

(13) This procedure interpolates the solutions derived from the calibrators into the calibration (CL) table for *all* sources. Run this procedure once for each phase calibrator (which may be used to calibrate multiple sources). **VLACLCAL**

```
SOURCES '< phase_cal >' ,'< source_1 >' , ... , '< source_n >'
CALSOUR '< phase_cal >' , ' '
OPCODE 'CALI' ; INTERPOL '2PT'
OUTPRINT '< filename_in_all_caps >'
```

Note: if you are observing at low frequencies or there are gaps in your observations of phase calibrators, you may want to use SAMPTYPE 'BOX' and review the other inputs carefully. They have changed.

(14) Apply the calibration to the phase calibrators and examine the amplitudes, which should be nearly constant, and the phases, which should be nearly zero. **LISTR**

```
SOURCES ' ' ; CALCODE '*'          ⇐ print results for all calibrators
OPTYPE 'MATX' ; DOCALIB 2 ; GAINUSE 2; DPARM 5, 1, 0
UVRANGE 0 ; DOCRT -1 ; OUTPRINT '< filename_in_all_caps >'
```

(15) Examine the *uv* data for your *sources* on the TV to check for any obvious problems which you may have missed. Re-edit the data (Steps 6–9) if necessary. **TVFLG**

```
DOCAL 2 ; GAINUSE 2 ; CALCOD '-CAL'
```

If you have too many visibilities to fit on the TV screen, you may want to set TIMERANG, SOURCES or DPARM(6) (the input averaging time) to limit the amount of data displayed. There are also interactive options to set the on-screen averaging time and the time range currently displayed.

(16) To calibrate the spectral-line data, simply copy the calibration (CL) table from the CH0 to the LINE data. **TACOP**

```
INEXT 'CL' ; NCOUNT 1 ; INVERS 2
```

Also be sure to copy the flagging (FG) table

```
INEXT 'FG' ; INVERS 1
```

Steps 17 and 18 should be applied to the LINE data alone, not to the CH0 data

(17) Calibrate the bandpass for the LINE data using the primary (flux) calibrator. The output from this task is a table (BP) of the bandpass spectrum. **BPASS**

```
GET3NAME < CH0data >
CALSOUR '< flux_cal >' , ' '; CALCODE '*'
DOCALIB -1 ; FLAGVER 1
```

(18) Examine the bandpass for each of the antennas on the TV. **POSSM**

```
APARM(8) 2 ; DOTV 1 ; STOKES 'RR'
ANTENNAS 0 ; NCOUNT 4              ⇐plot 4 antennas at a time
```

Then do the LL Stokes. After this, generate a plot (PL) file of the total bandpass for each Stokes.

```
NCOUNT 0 ; DOTV -1
```

Plot the bandpass on the laser printer; specify PLVER for each Stokes. **LWPLA**

(19) Now that the calibration is completed, write the calibrated CH0 and LINE data to tape. **FITTP**

Print the contents of the tape(s) for your data. **PRTTP**

(20) Apply the calibration and editing tables, writing single-source *uv* files for imaging. **SPLIT**

```
CALCODE '-CAL'                    ⇐write uv for all non-calibrator sources
DOCALIB 2; GAINVER 2 ; DOBAND 1; BPVER 1
```

B.2 Making and Cleaning image cubes

(21) Determine the imaging parameters needed for a full-field image of the continuum. Set the BPARMs to cover the full single-dish beam area (or more at 21-cm wavelength). **SETFC**

```
IMSIZE 0 ; CELLSIZE 0
BOXFILE 'MYDATA:< source_name >.BOX'
```

Delete the output BOXFILE and re-run the task with your chosen (more congenial or exact) CELLSIZE.

(22) Make a CH0 image with a large field (set by IMSIZE and CELLSIZE returned by SETFC) to look for strong continuum sources. Use uniform weighting (UVWTFN ' ') with ROBUST of 0 or -1 if resolution is more important than detecting significantly extended sources; set ROBUST to 2 or more if the converse is true. **IMAGR**

```
STOKES 'I' ; NITER 0
```

Then make a dirty image cube of your LINE data in the center field only (unless more are needed for the line source).

```
NFIELD 1 ; BCHAN 1 ; ECHAN 0
```

(23) Examine the cube on the TV to determine which channels are free of line emission. **TVMOVIE**

(24) Calculate the noise in a few of the line-free channels.

Display a channel on the TV **TVALL**

```
TBLC = 0 , 0 , < channel_number > ; TTRC 0
```

Select a large window that contains no continuum sources. **TVWIN**

Calculate the RMS noise inside the window. The verb TVSTAT is helpful if there are no large rectangular windows free of continuum emission. **IMSTAT**

Having an initial guess of the signal-free RMS, you may attempt to refine that estimate (see message generated) and also produce a histogram plot. **IMEAN**

```
BLC 0 ; TRC 0; DOTV 1; DOHIST 1; NBOXES 256
```

(25) Remove the continuum emission by fitting a baseline in the *uv* plane to the line-free channels; see EXPLAIN UVLSF and § 8.3. If you don't want this task for flagging, but only for continuum subtraction, the important parameters are **UVLSF**

```
ORDER 1 ; DOOUT 1 ; FLUX 0 ; CUTOFF 0
```

Choose your line-free channels, normally avoiding the channels at the edges of your bandpass since they are usually quite noisy.

```
ICHANSEL < begin_1 >,< end_1 >,, 0, 0, < begin_2 >,< end_2 >
```

This task will also write a new continuum data set which is a better estimate of the continuum than the CH0 file which contains contributions from the line signals. If there is a bright continuum source far away from the phase center, you *will* want to use SHIFT to center it while running UVLSF. Multiple interfering sources might even require UVSUB with a good continuum model rather than UVLSF.

(26) Make a dirty cube containing only the line emission. Use the same parameters as in Step 22, including usually only one field — but consider multi-scale Clean. **IMAGR**

(27) Calculate the noise in the line-free channels as in Step 24.

(28) Make a contour plot of the beam.
Display a channel on the TV. **TVALL**

`TBLC = 0, 0,` $< channel_number >$ `; TTRC 0`

Select a window containing the source (or beam) you want to plot. **TVWIN**
Display the contour plot on the TV. For a beam plot, good parameters are **KNTR**

`PLEV 10 ; LEVS -3,-1,1,3,5,7,9 DOTV 1`

When you are happy with the plot, generate a plot (PL) extension file.

`DOTV -1`

Send the plot file to the default printer. **LWPLA**
Measure the beam diameter at half-power (FWHP) to get beam parameters (BMAJ, BMIN, BPA) for the Cleaning process. If you do not specify these, IMAGR will choose them automatically by fitting a Gaussian beam with an elliptical cross-section. This is usually fine for uniformly-weighted images (ROBUST around 0), but may not be desirable for naturally-weighted images for which the beam is often rather non-Gaussian.

(29) Select boxes containing all of the line emission in the cube. Use as many boxes as necessary (up to 50). The better that you constrain the locations of real emission, the faster the Cleaning process will go. Note that IMAGR allows you to set the boxes interactively when you run with DOTV 1, which is important when Cleaning reveals emission initially lost in the sidelobes of the stronger objects. **TVBOX**

(30) Make a Clean cube of the line emission. Start by Cleaning just one or two channels (set with BCHAN and ECHAN) to ensure that your inputs are set correctly. If you don't want to waste time, it is a good idea to Clean only those channels with emission that is ≥ 4 times the RMS noise (Step 20). Use SUBIM and MCUBE to construct a full cube of the images later. Use the same parameters as in previous runs of IMAGR, but specify the beam parameters (BMAJ, BMIN, BPA) and control the Cleaning depth with FLUX and NITER. **IMAGR**

(31) Examine the Cleaned cube and do progressively deeper cleaning with IMAGR until you are satisfied with the result. The DOTV 1 option in IMAGR will help you reach this state more quickly. You can examine the Clean components with PRTCC. You should Clean until the total Cleaned flux converges. **TVMOVIE**

(32) Correct for attenuation away from the center of the primary beam. Use your Cleaned cube as the input, the default parameters should be adequate, and the output is a corrected cube. **PBCOR**

(33) Back up the Cleaned cubes and the cubes after PBCOR to tape. **FITTP**
Print the contents of the tapes for your records. **PRTTP**

B.3 Moment analysis and rotation curve of galaxies

After correcting for primary beam attenuation with PBCOR, the noise in the images will depend upon position. Because of this, you should use the uncorrected line cube for moment analysis.

(34) The frequency axis can be labeled in either frequency or velocity units. Make sure that the desired units are chosen; use IMHEAD to check. Velocity is recommended for moment analysis. **ALTSW**

(35) Transpose the axes to VEL-RA-DEC (or FREQ-RA-DEC) order. **TRANS**

```
TRANSCOD '312'
```

(36) Generate images of the total emission (MOM0), the velocity field (MOM1), and the line width (MOM2) using the transposed cube as input. Be sure to exclude the end channels as they generally are very noisy. Try various values for the flux cutoff FLUX and the width of the smoothing functions (set by CELLSIZE) until you are satisfied with the results. **MOMNT**

(37) Correct for the attenuation away from the center of the primary beam in the MOM0 image, as in Step 28. If you computed the moment images with a velocity axis, use ALTSW to change to a frequency axis before running PBCOR. **PBCOR**

(38) Make contour plots of the MOM0 and MOM1 images; see Step 24. Note that KNTR can superpose contour and grey-scale plots as in Figure 8.4. **KNTR**

(39) The task GAL allows you to generate a "tilted-ring" model rotation curve of two types from a galaxy's velocity field or to fit single-parameter rotation curves to annuli of a specified width. The EXPLAIN file for this task describes all of the parameters in detail and contains general advice on how to obtain an optimum fit. **GAL**

The task CUBIT, written by Judith Irwin, allows you to fit a rotating galaxy model to the full Cleaned image cube. The EXPLAIN file for this task describes all of the parameters in detail and contains general advice on how to obtain an optimum fit. **CUBIT**

B.4 Multi-frequency observations

B.4.1 General frequency information

For any *uv* data file, the frequency listed in the header information is the sky frequency of the center of the band (LINE or CH0) during the first scan of the observation. This is true regardless of whether you observed at a single frequency or multiple frequencies. After you SPLIT a multi-source file into single-source files, the frequency in the header refers to the sky frequency at the center of the band during the first scan *on that source*.

Corrections for the Doppler shift due to the rotation of the Earth can be taken into account within *AIPS*, if the data were observed at fixed frequency. Task CVEL may be used, but it requires that the spectra be well sampled in frequency.

B.4.2 Multi-frequency *uv* files

A simple rule of thumb: If you want to calibrate sources together, load the data with the same value of FREQID in FILLM. If you want to calibrate sources separately, give them different FREQIDs. For multi-frequency files, you should be sure to assign a different qualifier (QUAL) to each observing frequency (or velocity) with the OBSERVE program *before taking the observations*. There are essentially two types of multi-frequency observations:

1. Standard multi-frequency observations in which you want to do the entire calibration process separately for each frequency. When reading in such data with FILLM, set CPARM(7) = 0. This sets a different value of FREQID to data that differ by more than the maximum Doppler shift in a source in a day. During calibration, you can control which data you process by choosing the appropriate values for FREQID and QUAL. After calibration of each source/frequency, you may destroy the SN table to avoid using it to calibrate sources with different FREQIDs. For each source/frequency, you should create a new version of the calibration (CL) and bandpass (BP) tables (*e.g.*, for the second source/frequency, you will create version 3 of the CL table and version 2 of the BP table). (It principle, FREQIDs may co-exist in single tables without interference, but if they are in carefully specified separate tables they cannot interfere with each other.)
2. Observations of, or affected by, Galactic emission or absorption, in which you want to combine data at different frequencies to do the calibration. Normally, these are observations in which the calibrator sources themselves are absorbed by Galactic HI around 0 velocity. It is *extremely important* that you assign a different qualifier to each frequency with the OBSERVE program. Then load the data with FILLM forcing a single value of FREQID by setting CPARM(7) = -1. In this case, the information that two observations with the same FREQID have different frequencies will be contained only in the qualifiers. Whatever data are loaded with the same value of FREQID will have the same reference frequency; it should be possible to average over the observing frequencies using the appropriate programs in *AIPS* (CLCAL and BPASS).

B.5 Additional recipes

B.5.1 Banana breeze pie

1. In a small saucepan, melt 1/3 cup **butter** or **margarine**. Add 1/4 cup **sugar** and 1/2 teaspoon **cinnamon**. Stir constantly over low heat until bubbles form around the edges of pan.
2. Remove from heat, add 1 cup **corn flake cereal** crumbs and mix well. Press mixture evenly into a 9-inch pie pan to form crust. Chill.
3. Beat 8 ounces softened **cream cheese** until light and fluffy. Add 1 15-ounce can **condensed milk** and blend thoroughly. Add 1/3 cup **lemon juice** and 1 teaspoon **vanilla**. Stir until thickened.
4. Slice 3 ripe **bananas** and line crust. Pour filling into crust and refrigerate for 2–3 hours or until firm. Do not freeze.
5. Slice 2 ripe **bananas**, dip in lemon juice and arrange on top of pie. Note. for a change of pace, use lime juice.

B.5.2 Banana cutlets

1. Peel 6 medium-ripe **bananas** and halve them crosswise.
2. Dip them in 1/3 cup **lemon juice** and then roll in 1 cup crushed **corn flake crumbs**.
3. Saute them in 3 tablespoons **butter** until a golden brown.
4. Serve on lettuce.

C A Step-by-Step Recipe for VLBA Data Calibration in *AIPS*

This appendix provides a step-by-step guide to calibrating many types of VLBA and HSA (High Sensitivity Array or Effelsberg, Arecibo, GBT, and phased VLA) experiments. Continuum strong-source or phase-referencing observations are included, as are simple spectral-line observations. This appendix applies specifically to data sets with full calibration transfer. There is an addendum (§ C.10) describing issues with flagging for non-VLBA data sets and other matters for cases in which not all calibration data are loaded by FITLD. It may often be used (with some modifications in loading amplitude data) for data sets containing other antennas. Simple VLBA utilities that go all the way up to and including fringe-fitting are described.

C.1 Quick Guide

This section is meant as a reminder for people who have some idea how to reduce straightforward VLBI data and a familiarity with *AIPS*. For further details see the rest of this appendix and/or Chapter 9. This section does not include things like CVEL or AVSPEC that might be useful for spectral line observations. Also, this section refers to procedures that can loaded by typing RUN VLBAUTIL. All these steps are performed, in this order, by the 31DEC25 VLBAPIPE (section § C.5) pipeline procedure.

1. VLBALOAD — load the data
2. VLBAFIX — fix the data (separate frequencies, merge tables, create index table, etc.)
3. VLBAEOPS — correct earth orientation parameters
4. VLBATECR — (if frequency < 10 GHz) remove dispersive delay from ionosphere
5. VLBAPANG — correct parallactic angle
6. VLBACCOR — correct sampler threshold errors from correlator
7. VLBAPCOR or VLBAMPCL — (if more than 1 IF) correct instrumental delays, use POSSM to inspect correction
8. VLBAFRNG or VLBAFRGP — fringe fit data
9. VLBABPSS — calibrate bandpass, use POSSM to inspect bandpass solutions
10. VLBAAMP — corrects any auto-correlation departure from unity caused by previous steps and performs a-priori amplitude calibration, use SNPLT to check that nothing has gone wrong in the amplitude calibration.
11. RFLAG — flag data that are too variable in time or frequency; optionally stop pipeline.
12. IMAGR and FRING or CALIB — do one round of self-calibration on each source in CALSOUR optionally
13. SPLIT — apply all calibration and possibly average data spectrally
14. IMAGR — image data

These steps, in a different order, are also performed by the older VLBARUN (section § C.5) pipeline procedure.

1. VLBALOAD — load the data
2. VLBAFIX — fix the data (separate frequencies, merge tables, create index table, etc.)
3. VLBAEOPS — correct earth orientation parameters
4. VLBATECR — (if frequency < 10 GHz) remove dispersive delay from ionosphere
5. VLBACCOR — correct sampler threshold errors from correlator
6. VLBAPCOR or VLBAMPCL — (if more than 1 IF) correct instrumental delays, use POSSM to inspect correction
7. VLBABPSS — calibrate bandpass, use POSSM to inspect bandpass solutions
8. VLBAAMP — corrects any auto-correlation departure from unity caused by previous steps and performs a-priori amplitude calibration, use SNPLT to check that nothing has gone wrong in the amplitude calibration.
9. VLBAPANG — correct parallactic angle
10. VLBAFRNG or VLBAFRGP — fringe fit data
11. RFLAG — flag data that are too variable in time or frequency
12. IMAGR and FRING or CALIB — do one round of self-calibration on each source in CALSOUR optionally
13. SPLIT — apply all calibration and possibly average data spectrally
14. IMAGR — image data

C.2 Table Philosophy

AIPS follows an incremental calibration process on multi-source data sets. Calibration solutions are written to SN ("Solution") tables, which can be inspected in various ways. CLCAL is used to apply an SN table and write a new CL ("Calibration") table, which stores the cumulative calibrations. If the procedure in VLBAUTIL is used, much of this is hidden, however most procedures will report the new SN andor CL that are produced. The actual visibilities are not altered until the final calibration is applied using SPLIT (or SPLAT), which produces single-source (or multi-source) data sets that can be imaged. With this philosophy, it is easy to back up a step or two if errors are made in processing. Users should keep track of which tables contain which solutions and calibrations as they go through the calibration process.

A key verb to be aware of is EXTDEST, which can delete any unwanted table. For example, to delete SN version 3 from the data set cataloged as data set 1 on disk 1, type `INDISK 1; GETNAME 1; INEXT 'SN'; INVER 3; INP EXTDEST; EXTDEST`. *Beware of the fact that once a table is deleted, there is no 'undelete' function.*

C.3 Data set assumed in this Appendix

This appendix assumes a VLBA-only data set observed at several frequency bands (*e.g.*, 1.6, 2.3, and 5.0 GHz). To include data from the non-VLBA antennas see § C.10. It is also assumed that phase-referencing programs have been observed according to the philosophy discussed in detail in VLBA Scientific Memo No. 24. The hypothetical observation considered here contains the following sources:

- 'CAL-BAND' — fringe-search and bandpass calibrator
- 'CAL-AMP' — amplitude-check source
- 'CAL-POL' — polarization position angle calibrator
- 'STRONG' — strong target source
- 'CAL-PHASE' — phase-reference source
- 'TARGET' — weak target source, to be calibrated with CAL-PHASE

In the text below, table versions, such as SN version 1, are referred to as SN 1.

C.4 VLBA Utilities

Note that there are simple VLBA procedures ("front ends" to standard tasks) that will take the user all the way from data loading up to and including fringe-fitting. These are tremendous labor-savers for those working with reasonably straightforward data sets. For spectral line, use the procedures to calibrate a lower spectral resolution version of the spectral line data and copy the final calibration to the line set. To access the utilities, type RUN VLBAUTIL from inside *AIPS*. The procedures that we will use in this appendix are:

- VLBALOAD: loads VLBA data with simplified inputs
- VLBASUMM: makes summary listings of your data set
- VLBAFIX: Fixes VLBA data
- VLBATECR: automatically downloads and applies ionospheric corrections
- VLBAEOPS: automatically downloads and applies corrections to the Earth Orientation Parameters used by the correlator
- VLBACCOR: determines amplitude corrections caused by errors in sampler thresholds
- VLBAPANG: determines phase corrections for parallactic angles
- VLBAPCOR: determines instrumental phase corrections using pulse cals
- VLBAMPCL: determines instrumental phase corrections using FRING
- VLBACPOL: calibrates cross polarization delays
- VLBABPSS: does bandpass calibration
- VLBAAMP: does adjustment due to normalization errors and a-priori amplitude calibration
- VLBAFRNG: does global fringe fit using FRING
- VLBAFRGP: does global fringe fit for phase referenced experiments using FRING
- VLBASNPL: plots the SN or CL tables versus time
- VLBACRPL: plots the cross-correlation spectrum

There are two additional procedures that can make life easier, called ANTNUM and SCANTIME. ANTNUM will return the antenna number of the antenna corresponding to a certain character string. For example, in many data sets, typing REFANT = ANTNUM ('BR') will be the equivalent of typing REFANT = 1. SCANTIME will return the time range of a given scan number, for use in various programs. Typing TIMERANG =

SCANTIME(4) will fill the eight-element array TIMERANG with the start and stop times of the $4^{\rm th}$ scan of a given data set.

Note that all of the VLBAUTIL procedures have HELP files with good discussions about when to use the simple procedures and when to use the tasks directly. Also, note that the procedures do not include data editing, which should be performed at appropriate points in the calibration process. You only need to RUN VLBAUTIL once to access all of the procedures. If you run it again for any reason, it is a good idea to type COMPRESS immediately afterward to avoid overflowing *AIPS*'s symbol memory.

C.5 VLBA Pipeline

VLBAPIPE and VLBARUN are two similar procedures which use the VLBA calibration procedures (from VLBAUTIL) and some logic to calibrate and image VLBA data. They both attempt to make intelligent decisions on defaults, so they can be run fairly automatically if the names of the sources are known. If desired, VLBAPIPE and VLBARUN will produce diagnostic plot files and write them to disk, creating an HTML file to ease examination of these files. Images will be produced but only one phase-only self-cal is done (if requested), so the images should be considered diagnostic in nature.

VLBAPIPE and VLBARUN do all the calibration steps described in § C.1 and the next few sections but in a different order. They do not do polarization calibration. They offers options to use CLIP and/or RFLAG to do flagging. They also offer options to flag data at low elevations and to try to clean up bad values in the system temperature table. Although VLBAPIPE and VLBARUN are intended for simple VLBI observations, the addition of non-VLBA antennas should work if all the non-VLBA system temperature and gain curve information are loaded into the first TY and GC tables.

The two piplines use essentially the same input adverbs, except for the meaning of DOFLAG parameter. Sample inputs for procedure VLBAPIPE are:

> RUN VLBAUTIL ↵	to acquire the procedures used by VLBAPIPE.
> RUN VLBAPIPE ↵	to acquire VLBAPIPE.
> INDISK n ; GETN ctn ↵	to specify the input data file.
> OPTYPE 'CONT' ↵	to say this is a continuum data set.
> CLINT 0 ↵	to use default.
> CHREFANT 'FD' ↵	to set reference antenna to Fort Davis.
> CHINC -1 ↵	to have the procedure determine an appropriate channel increment for FRING.
> TIMERANG 0 ↵	to have VLBAPIPE determine a good instrumental delay calibration scan.
> INVER 0 ↵	PC table to use, -1 if you want to force manual phase cal.
> CALSOUR 'CAL-BAND', 'CAL-PHASE', 'CAL-AMP' ↵	to list calibrators, bandpass and initial FRINGcalibrator *must* be first.
> SOURCES 'CAL-PHASE', 'TARGET', ... ↵	to list phase referencing and target pairs.
> INFILE '' ↵	set if you want to apply a zenith delay file produced by DELZN.
> SOLINT 0 ↵	to use 1 minute averages in FRING.
> IMSIZE 512 ↵	to make images and specify size of target images.
> FACTOR 0 ↵	to make calibrator images 128x128.
> DOCLIP fp, fc ↵	to run CLIP to flag a sample when the parallel-hand flux is above $\|fp\|$ or the cross-hand flux is above $\|fc\|$. If fp is positive, the clip is done after amplitude calibration; if negative, it is done before the bandpass and amplitude calibration. The appropriate values for fp and fc are different in these two cases.

> DOFLAG 1 ↵	to run RFLAG on each source individually.
> TYSMOLEV 100 ↵	to clip system temperatures above 100 K with median-window filtering at 10 K and replacing the clipped values with nearby good ones.
> ELEVLIM 15 ↵	to flag out all data for which the elevation is < 15 degrees.
> SELFCAL 1 ↵	to do one pass of self-cal, phase only on each calibrator prior to the final SPLIT¿
> GET2N *ctn2* ; NMAPS=1 ↵	to use a model for the bandpass calibration.
> BPASS5 -2 ↵	to normalize bandpass scans before the fit. Default is +1 to normalize the bandpass functions themselves without normalization of the data.
> BPSOLINT 0 ↵	to make a solution for each calibration scan. BPSOLINT=-1 averages all calibration scans.
> DOBAND 3 ↵	to interpolate the bandpass over time. DOBAND=1 (the default) averages all bandpass records.
> CALTASK 'CALIBS' ↵	to use FRING for the initial time series calibration but use CALIB for any self-cal. Other choices are 'CALIB' and 'FRING' (the default) to use the specified task for all time series calibrations.
> DOPLOT 1 ↵	to make some diagnostic plots. Use level 2 only if there are serious issues; it generates a lot of plots.
> OUTFILE *'/directory/name* ↵	to specify directory for output plots. If this is set, plots are written out from *AIPS* and organized in an HTML file for easy viewing. The plots, HTML file, and message file contents are put in a dated subdirectory below the specified directory.
> OUTTEXT *'email@somewhere.edu* ↵	to specify an e-mail address if the user wants to be notified when the pipeline is done.
> BADDISK 0 ↵	to specify which disks to avoid for scratch.
> VLBAPIPE ↵	to run the procedure.

The order of operations differs between the two pipelines. The most significant difference is that FRING is run before BPASS in VLBAPIPE. This should insure that the bandpass solutions, when there are multiple primary calirator scans, remain phase coherent. The only adverb that differs between the two pipeline scripts is DOFLAG. In VLBAPIPE, the pipeline will shut down after running RFLAG if $0 <$ DOFLAG < 10 to allow you to rerun the pipeline using the RFLAG flag tables. Presumably, the calibration in the second pass should be better. If DOFLAG ≥ 10 in VLBAPIPE or DOFLAG> 0 in VLBARUN the pipeline invokes RFLAG and then continues to the optional self-cal step followed by SPLIT and IMAGR. VLBARUN uses the value of DOFLAG as an initial clip level in RFLAG but only if it is greater than 100. VLBAPIPE does not use the initial clip option in RFLAG, leaving that to the DOCLIP adverb.

It is a good idea to examine your TY table with SNPLT before running VLBAPIPE; this will tell you what a reasonble upper limit to $T_{\rm sys}$ will be and whether there are bad values in need of correction. SNRMS provides a summary of the statistics of tables such as the TY table. It's maxima and rmses will make problems with the table very clear. Note that editing with CLIP is always a good idea. RFLAG appears to work well at low frequencies but to be overly enthusiastic at C-band and above. Note that the flagging is done after all calibration steps, so you may examine your data with and without the new flags to determine their correctness. RFLAG is run using baseline-dependent clip levels and so should work with arrays containing high-sensitivity telescopes as well as the VLBA. You may wish to use VLBAPIPE without RFLAG (or to discard the flags produced by RFLAG) and then run SPFLG to flag the visibility data interactively. Then use VLBAPIPE over again applying the manually produced flags. Procedure R_RESTART (P_RESTART in VLBARUN) will clear away all plot and calibration tables from your data set, leaving any flag tables. Do not delete version one of

the CL and FG tables since they contain the initial calibration and on-line flagging information.

VLBAPIPE and VLBARUN will then run and produce the requested number of diagnostic plots. For details on the plots produced for each level of DOPLOT see EXPLAIN VLBAPIPE. If an e-mail address is specified then a VLBAPIPE DONE or VLBAPIPE FAILED message will be sent. However, the VLBAPIPE FAILED message will only be sent if VLBAPIPE failed because of problems with the inputs, if the pipeline failed because a task it was running failed this message is not sent. It is highly recommended that the user read the explain file for VLBAPIPE.

A similar RUN file named VLBAMPHC is available to calibrate, flag, and image data sets for secondary phase stopping centers when DiFX has been used to correlate multiple centers from a single pointing. A pipeline called VLBARFI appeared in 31DEC21. It calibrates a data set concentrating on the autocorrelation data on which it performs a statistical analysis.

C.6 Data Loading and Inspection

1. Load the data using VLBALOAD (which is a very simplified FITLD). CLINT should be set so that there are several CL table entries for each self-calibration or fringe-fitting interval anticipated; this will minimize interpolation error during the calibration process. However, setting CLINT too short will result in a needlessly large table. Somewhere between CLINT = 0.25 and CLINT = 1.0 is about right. A FITLD parameter that is set automatically in VLBALOAD is WTTHRESH = 0.7, which results in irrevocable discarding of all data with playback weight less than 0.7. The only way around this is to use FITLD explicitly. The data will be assigned weights that reflect the playback weight plus the integration time and channel bandwidth. If you run FITLD explicitly you may turn this off with DOWEIGHT <= 0.

2. Correct data with VLBAFIX. *If necessary*, VLBAFIX sorts (with MSORT), splits into different frequencies (with UVCOP), fixes the polarization structure (with FXPOL), and indexes (with INDXR) the data. VLBAFIX will also correct for subarrays (with USUBA), but you must tell it to do so. There are only 2 inputs of interest in VLBAFIX, CLINT, the CL table interval, and SUBARRAY which should be set to 1 if there are subarrays and 0 if not. This is a very benign procedure, it can be run on every data set read into AIPS and will only perform the necessary fixes. Note that, if the data are split into different frequencies, the flag table is applied and deleted.

3. At this point it is a good idea to get a listing of the antennas and scans in your data by running VLBASUMM. VLBASUMM runs PRTAN over all antenna tables and LISTR with OPTYPE='SCAN' and gives a choice of writing a text file to disk or sending the listing to a printer.

4. Apply corrections to the Earth Orientation Parameters (EOPs). VLBI correlators must use measurements of the Earth Orientation Parameters (EOPs) to take them out of the observations. These change slowly with time and therefore the EOPs used by the correlator must be continually updated, and they are generally best two weeks or more after the observation. Since we try to correlate observations as quickly as possible, it is likely that the EOPs used are not the most accurate. Therefore it is recommended that all phase-referencing experiments be corrected for this possible error. The procedureVLBAEOPS will do this correction. VLBAEOPS automatically downloads a file with correct EOPs and runs CLCOR to correct the EOPs. If the automatic download does not work, VLBAEOPS can use a file which you have downloaded. *Warning: this procedure does not work well over concatenated data sets containing multiple days; uss* CLCOR *explicitly and set* CLCORPRM(2).

5. Apply ionospheric corrections, if desired, with VLBATECR. This procedure automatically downloads Global Positioning System (GPS) models of the electron content in the ionosphere and uses them to correct the dispersive delays and Faraday rotation caused by the ionosphere. If the automatic download does not work, VLBATECR can also use a file which you have downloaded, This is

particularly important for phase referencing and polarization experiments at low frequency. We recommend VLBATECR for all experiments at 8 GHz or lower. VLBATECR is only as good as the ionospheric model, so it is a very good idea to compare the corrected and uncorrected phases using VPLOT. To inspect the phases using VPLOT, use options BPARM = 0, 2; APARM=0; DOCAL=1; GAINUSE=*highest* CL *table*. The phases should not wind as much (although they will probably not be completely flattened), when the corrected CL table is applied. To see the corrections themselves, use SNPLT on the new CL table setting OPTYPE = 'DDLY'. In 31DEC23, tasks MFIMG and TEPLT may be of interest after running TECOR.

6. For a simple spectral-line data set, or any data set with high spectral resolution, it is a very good idea to average the data set to a smaller number of channels before deriving the calibration parameters. Otherwise, the calibration tasks may take forever to run. It is recommended that you quickly inspect the channels of interest for your line data (*e.g.*, with UVPLT) for high points. Remove obviously high amplitudes with CLIP (or *e.g.*, UVMLN, SPFLG) before averaging. Inspect the full resolution data also for high delays and fringe rates. Spectral averaging in such cases may not be acceptable. Continue calibration on the averaged data set as if it were a continuum set. There is a better method to calibrate spectral line data described in § 9.5.6 and § 9.5.7.10, but the one used here is simpler and will usually give acceptable results. To reduce the data-set size, run the task AVSPC with AVOPTION = 'SUBS'. For example, to average IFs with $N_{\rm chan}$ down to 32 channels, set the adverb CHANNEL = $N_{\rm chan}/32$ (*e.g.*, to average from 2048 to 32 channels, use CHANNEL = 64).

C.7 Amplitude and Delay Calibration

Amplitude calibration uses measured antenna gains and system temperatures ($T_{\rm sys}$), as well as finding a correction for voltage offsets in the samplers. Even though this is substantially still the case, we now recommend a somewhat different amplitude calibration procedure than in the past based on VLBA Scientific Memo #37 (Walker 2015).

Before amplitude calibration is done, there must be information for all antennas in the gain curve (GC), system temperature (TY), and weather (WX) tables (weather tables are needed for the opacity correction in APCAL). Missing $T_{\rm sys}$ and gain curve information can usually be obtained in ANTAB format and loaded with the *AIPS* task ANTAB. § C.10 has information on including non-VLBA calibration information, if they are not already included in the data. § C.11 has details on how to incorporate the pre-EVLA VLA $T_{\rm sys}$ and gain curves. Otherwise consult § 9.5.2.3.

1. Correct sampler offsets and apply amplitude calibration by running VLBACCOR. The procedure VLBACCOR runs ACCOR, SNSMO, and CLCAL. ACCOR uses the autocorrelation to correct the sampler voltage offsets. After ACCOR creates an SN table, SNSMO smooths the table in order to remove any outlying points. Then the SN table is applied to the highest CL table using CLCAL (using INTERPOL='2PT'), and a new CL table is created.

2. Next, the instrumental delay residuals must be removed. These offsets or "instrumental single-band delays" are caused by the passage of the signal through the electronics of the VLBA baseband converters or MkIII/MkIV video converter units. There are two different methods to remove these instrumental delays, one for the case where you have pulse-cal information for some, but not necessarily all, of your antennas; and one for the case where you have no pulse-cal information at all. Note that the preferred method for continuum experiments is to use the pulse-cals, since they correct the instrumental delay over the whole experiment, rather than on a short scan. Spectral-line observers would have switched off the pulse-cals as they interfere with line observations, so they are forced to use the second (strong source) method. For VLBA continuum experiments before April 1999, you can load the pulse-cal data using PCLOD; consult the § 9.5.2.

- For the case where you have some pulse-cal information, run VLBAPCOR. VLBAPCOR is another procedure which runs quite a few tasks, PCCOR, CLCAL, FRING (sometimes) and CLCAL again (sometimes). PCCOR extracts pulse-cal information from the PC table and creates an SN table. Then CLCAL is run to apply that SN table to the highest CL table, creating a new CL table. If there are antennas that do not have information in the PC table, or their PC entries are wrong, then VLBAPCOR can run FRING on a short calibrator scan (input TIMERANG). The SN table from FRING contains corrections for the antennas left out of the PC table, and is applied to CL table without corrections from PCCOR, and added to the CL table with the PC corrections. For the simplest case of all VLBA antennas, the inputs for VLBAPCOR should be TIMER=*time range on* CAL-BAND *with good fringes for all baselines*; REFANT=n; SUBARRAY=0; CALSOUR='CAL-BAND'; GAINUSE=0; OPCODE=''; ANTENNAS=0. For the case where you have the VLA (in this example antenna 11), which does not have pulse-cals, your inputs should be the same as above except OPCODE='CALP'; ANTENNAS=11,0. For the second case it is important that there are no "Failed" solutions from FRING; if there are failed solutions, then you should delete the tables that were created and find another TIMERANG with good fringes to the REFANT. Also see EXPLAIN VLBAPCOR for a detailed description of the steps involved with using pulse-cals and FRING without using VLBAPCOR.
- The alternate method is to solve for the phase cals "manually" with VLBAMPCL. This method uses the fringes on a strong source to compute the delay and phase residuals for each antenna and IF. VLBAMPCL runs FRING to find the corrections and then CLCAL to apply them. If there is no calibrator scan that includes all antennas then there is an option to run FRING and CLCAL again on another source and/or time range in order to correct the antenna(s) not corrected by the first scan. For the simplest case where all antennas have strong fringes to CAL-BAND, set TIMERANG = *time range of scan on calibrator with strong fringes to all antennas*; REFANT = n; CALSOUR = 'CAL-BAND'; GAINUSE = *highest* CL *table*; OPCODE = ''. There must be no "Failed" solutions from FRING, if there are any failed solutions the data from that antenna/IF will be completely deleted from the data set.
- Now you *must* check the results of correcting your instrumental delays using VLBACRPL or POSSM. Set GAINUSE=*highest* CL *table*, and plot cross-correlations (VLBACRPL will do this for you). The plotted cross-correlations should show the phase slope removed from each IF and there should no longer be a phase jump between IFs, although the phase does not have to be at 0°. If you do a "manual" instrumental delay correction (*i.e.*, you used FRING, not PCCOR); then the phases far in time from the scan on which FRING was performed may have a small slope and a small phase jump between the IFs. Also non-zero phase slopes still may be seen at low elevations, where the atmosphere causes additional delay residuals, or for low-frequency observations where the ionospheric delay varies. If you see significant phase slopes, or phase jumps between IFs on *any* baseline, then the instrumental phase corrections have not worked and you need to figure out why and start again.

3. Next you must calibrate the bandpass shapes. To do this, run VLBABPSS on the bandpass calibrator, CAL-BAND. Make sure that the spectral line data for the bandpass calibrator are clean and devoid of high points, using UVPLT or SPFLG. Inputs for VLBABPSS are CALSOUR = 'CAL-BAND' and a model for your calibrator if you have one. Then you must examine the BP table using POSSM by setting APARM(8)=2.

4. Now it is time to finish the amplitude calibration by running VLBAAMP. This procedure runs several tasks: ACSCL, SNSMO, CLCAL, APCAL and CLCAL. After the previous steps the calibrated autocorrelation amplitudes will be offset from unity; ACSCL corrects this offset. ACSCL creates an SN which is then smoothed by SNSMO and CLCAL is run to apply the calibration to the next CL table. Then to finalize the amplitude calibration, APCAL is run on the highest TY and GC tables, and a new SN table is created. Adverb DOFIT controls whether APCAL also uses the weather tables to fit and correct for opacity. It is desirable to perform an atmospheric opacity correction at high frequencies, particularly if very accurate source fluxes are needed. See §9.5.4.6 for a more detailed discussion of APCAL. Lastly, VLBAAMP runs CLCAL to apply the amplitude calibration SN table to the CL created by the last run on

CLCAL. VLBAAMP will print messages telling you about the new tables it has created. To keep track of your tables, it is important to copy these messages.

5. At this point it is a *very* good idea to examine your calibration.

   - Run the task SNPLT or procedure VLBASNPL (which is a very simplified SNPLT) to examine the tables created by ACCOR. Use INEXT='CL'; OPTYPE= 'AMP'; INVERS=CL-*table-with-sampler-offsets*; DOTV=1 (to display to the TV; for a hardcopy use DOTV=-1 and LWPLA to print the plot files). The solutions that SNPLT plots should be close to 1000 milligain or 1 gain.
   - Run SNPLT or VLBASNPL to examine the amplitude calibration. This time look at the SN table that APCAL created. Use INEXT='SN'; OPTYPE= 'AMP'; INVERS=*highest* SN *table*; DOTV=1 for VLBASNPL; or to inspect IF m, use BIF = m; EIF = m; OPTYP = 'AMP'; INVER = 0; INEXT = 'SN'; OPCODE = ' '; NPLOT = 10; DOTV = 1; GO SNPLT. For a hard copy, use DOTV = -1; GO SNPLT; GO LWPLA. Plotted amplitudes are the square-roots of the system-equivalent flux densities (SEFDs), in Jansky, where the SEFD is the flux density of a source that would double the system temperature. (Low numbers are good!) At centimeter wavelengths, VLBA antennas have SEFDs near 300 Jy, so gains above 30° elevation should be near 17–18 and should vary slowly and smoothly with time (*i.e.*, change in elevation) for an individual source. To look at the input system temperatures, run SNPLT with OPTYP = 'TSYS'; INEXT = 'TY'; INVER = 0. On rare occasions, you might find clearly discrepant points that have leaked in from a different frequency band. In that case, you can use task SNEDT, or the clipping option of SNSMO, to get rid of the bad points. You may notice that at low elevations the gains on individual antennas are high. All data below a given elevation can be flagged by running UVFLG; *e.g.*, to flag all data below 10°, run UVFLG with APARM(4)=0 and APARM(5) = 10. Note that FG tables are not applied to tables, so flagged data still may have points plotted by SNPLT. The $T_{\rm sys}$ measurements are also a very good diagnostic of bad data from poor weather, equipment failures, *etc.*. If there are time ranges of unusually high or low $T_{\rm sys}$ you may consider flagging those time ranges using UVFLG or EDITA. Be particularly suspicious of patches of unusual gains at only one IF or STOKES of an antenna. Remember, one of the best things you can do for your final result is to get rid of bad data.
   - You may wish to use your favorite method of inspecting data for flagging (*e.g.*, EDITR, SPFLG, TVFLG, IBLED). On-line flags are already included in FG 1 unless they were applied as the data were split into separate frequencies. For example, run EDITR with inputs SOURCES='CAL-BAND','' (do each source separately); DOCAL=1; GAINUSE=*highest* CL *table*; FLAGVER=0; OUTFGVER=0; DOTWO=1; ANTUSE=1,2,3,4,5,6,7,8,9,10. Once you gain experience you might want to set CROWDED=2 ; DO3COL=1 which allows 3-color displays of all polarizations and IFs in one plot with a faster method for the TV displays; this can speed up editing significantly. Look for anomalously high or low amplitudes, remember there can be a slow change in amplitude with time due to source structure. Some people do no additional flagging at this stage, but later use the results of fringe-fitting and visibility plots of calibrated data to point the way to bad data.

6. For spectral-line experiments needing velocity accuracy better than 1 km/s, a Doppler correction should be performed. Use CVEL; see § 9.5.4.5 and § 9.5.5 for details.

7. This is a useful time to run TASAV to save all your ancillary tables to another file. If you foul up the calibration, the relevant tables can be copied back using TACOP.

C.8 Rate and Phase Calibration

Now that the data have calibrated amplitudes, the next step is to do the calibration of the antenna rates and phases. This section describes that process.

1. Correct the antenna parallactic angles, if desired, using VLBAPANG. The RCP and LCP feeds on alt-az antennas will rotate in position angle with respect to the source during the course of the observation (all VLBA and VLA antennas are alt-az). Since this rotation is a simple geometric effect, it can be corrected by adjusting the phases without looking at the data. You *must* do this correction for polarization experiments and phase referencing experiments. Parallactic angles are important for phase referencing because the parallactic angle difference between calibrator and target is different at different stations which leads to an extra phase error which can be corrected. VLBAPANG copies the highest numbered CL table with TACOP and then runs CLCOR (OPCODE = 'PANG'; CLCORPRM = 1,0). VLBAPANG has no inputs that require discussion. Be sure to correct the parallactic angles before any of the following steps. Again keep track of which CL tables add which correction.

2. Now you must remove global frequency- and time-dependent phase errors using FRING or one of the procedures which use this task, VLBAFRNG or VLBAFRGP. This cannot be done simply for spectral-line sources, so the practice here is to determine delay and rate solutions from the (continuum) phase-reference sources and interpolate them over the spectral line observations. The procedures run FRING along with CLCAL. VLBAFRNG and VLBAFRGP use FRING, with VLBAFRGP specifically for phase referencing. For all these procedures, if the SOURCES adverb is set, then CLCAL is run once for each source in SOURCES. For the phase-referencing procedure (VLBAFRGP), any source that is in the SOURCES list that is *not* in the CALSOUR list will be phase referenced to the *first* source in the CALSOUR list. These procedures will produce new (highest numbered) SN and CL tables. Since it is probably best to run CLCAL on each source separately, SOURCES should always be set. To use VLBAFRGP for a simple phase referencing experiment (remember that CAL-PHASE is the phase reference calibrator), set CALSOUR='CAL-PHASE','CAL-BAND','CAL-AMP','CAL-POL','STRONG'; GAINUSE=*highest* CL *table*; REFANT=n; SEARCH 9 4 1 3 5 6 7 8 10; SOLINT=*coherence time*; DPARM(7)=1 *(if a polarization experiment)*; SOURCES='CAL-PHASE', 'CAL-BAND', 'CAL-AMP', 'CAL-POL', 'STRONG','TARGET'; INTERPOL='SIMP'. For this example, FRING will be run on the sources in CALSOUR and then CLCAL will be run 6 times, with all of the sources except TARGET referenced to themselves and TARGET referenced to CAL-PHASE, using interpolation method SIMP. For a non-phase-referencing experiment you would use VLBAFRNG with inputs the same as above except for SOURCES, which would not contain TARGET. The results will be the highest SN and CL tables. The INTERPOL to use is a personal preference. You might want to restrict the channel range slightly using BCHAN and ECHAN, since the channels at the high end of each IF will have lower SNR, due to the cutoffs in the bandpass filters. For a data set with 16 channels per IF, numbered from 1 to 16, setting ECHAN to 14 or 15 may be worth trying. Note that some people like to run CALIB rather than FRING or KRING for this stage of phase-referencing observations, but fringe fitting is recommended, as it solves for rates.

   The above fringe fit may take a bit of time, depending on the computer and the spectral resolution. Then, use SNPLT or VLBASNPL to inspect the solutions in the SN table. It's not totally out of the question that some data will be found that need flagging, which can be done with UVFLG. In that case, it's a good idea to delete the last SN and CL table and re-run VLBAFRGP or VLBAFRNG.

   This fringe-fitting stage is the most likely place where things can go wrong, for reasons that are not immediately apparent to the observer. Below, a few common examples are listed.

   - **Many solutions failed.** The source may be too weak, or the coherence time too short. Try increasing or decreasing SOLINT. Or narrow the search window. For most VLBA data, DPARM(2) = 400 and DPARM(3) = 60 should be a good first step, though the rate window specified in DPARM(3) is proportional to the observing frequency, and may need to be larger at 22 GHz and above. For more options you could try running FRING and reduce the SNR threshold with

APARM(7) or averaging the RR and LL (APARM(3)=1). One last thing to try is to abandon FRING and solve for the phases in CALIB, obviously not ideal since you will not get rates or delays, but it is a hit worth taking if the the calibration can be salvaged.

- **Some antenna has low SNR, and may cause an entire set of solutions to go bad.** This typically happens because an antenna should have been flagged. A common cause is when OV is looking at the White Mountains, and neither the on-line system nor the astronomer has flagged the data. Then, you need to run UVFLG and re-run VLBAFRGP or VLBAFRNG.
- **There are discrepant delay/rate solutions.** Look at the solutions you believe, and try VLBAFRGP or VLBAFRNG again with DPARM(2) and DPARM(3) specified appropriately. Full widths are specified, so if the good solutions fall between $+15$ mHz and -15 mHz, use DPARM(3) = 30. (Actually, you should use a value somewhat larger to allow some margin.) It may be that an antenna is suffering from radio-frequency interference, so some channels and/or IFs will need to be flagged.
- **Some solutions are outside the specified delay/rate range.** This can happen because the initial coarse fringe search uses the range specified by DPARM(2) and DPARM(3), but the least-squares solution can take off from there and go elsewhere.
- **Delays and rates for some station change rapidly near the beginning or end of the observation.** This may be caused by low elevation at the relevant station. Depending on how desperate you are to include low-SNR data, you may wish to flag some time range, or flag all data at elevations below 5°, 10° or even 20° (particularly at high frequencies or for phase referencing) with UVFLG.
- **Phases wrap rapidly, particularly on the phase-reference source, CAL-PHASE.** There may not be a lot you can do about this initially, because it's possible that the tropospheric delay just changed too fast for the cycle time used in the observation, especially at low elevation. However, you may wish to note the times and antennas when the phase connection is best (typically the southwestern antennas near transit). Later, when imaging the program source, it can be helpful to image with a subset of antennas and time ranges, then use that initial image to self-calibrate the rest of the data.

3. Use SNPLT or VLBASNPL to inspect the interpolation of the phases in the CL table. When you inspect the CL table notice any phase wraps that seem out of place. The human eye is better at pattern matching than a computer and these phases may be in error. You can use SNBLP to examine the phase solutions on a baseline basis (as they actually affect the visibilities). This may make problems more apparent. If so you might want to run CLCAL independently and try another interpolation method or you might want to edit the CL table. Remember that this is your last calibration table; you want to get rid of any bad calibration now before applying it to the data. Getting rid of spurious wraps in the final CL table (SNEDT) or flagging the data associated with fast changing phases (SNFLG) will improve your final image more consistently than anything else, *particularly* for phase referencing.

C.9 Final Calibration Steps

1. If you used AVSPC to reduce the size of the data set used in determining calibration, you must copy your final calibration tables back to the full-size data set. This can be done with task TACOP. For bookkeeping purposes, it may be best to copy over all the CL tables with the same table numbers in both the averaged and un-averaged data sets. Copy the FG table as well, since any data which are bad in the averaged data set will be bad in the full resolution data set. After inspecting the data with UVPLT or VPLOT, run EDITR, TVFLG, CLIP, SPFLG, or other data editor to edit the bad data from the calibrated spectral-line data set.

2. After you have corrected the bandpass for spectral line data, you may want to correct for the change in frequency by the motion of the antennas with respect to the Sun *etc.*. This is done with CVEL, after the source velocities are entered in the SU table with SETJY. For a detailed description see § 9.5.5.

3. Polarization calibration still remains, if desired, and if all the appropriate calibration sources were observed. This can be done in a variety of ways; see § 9.5.7 for details.

4. Finally, apply the calibration to the visibility data and make single-source data sets using SPLIT. (Some people might wish to use SPLAT to average over time as well as spectral channel.) Inputs for a continuum observation are SOURCES = ' '; BIF = 0; EIF = 0; DOPOL = -1 (or 1 if polarization calibration was attempted); DOBAND = 1; DOUVCOMP = 1; NCHAV = 0; APARM = 2,0; DOCALIB = 1; GAINUSE = *highest* CL *table*. For a spectral-line observation, set APARM = 0, because you don't want to average over frequency. Use OUTDISK and OUTCLASS as appropriate for your computer and record-keeping purposes.

The single-source data sets are now ready for imaging and possible self-calibration. At this point, it is a good idea to look at the amplitude check source 'CAL-AMP' using tasks such as UVPLT, WIPER or VPLOT in order to see if there are any antenna gain calibrations that must be adjusted. Doing a WIPER for each target source is a good idea also, because there may be discrepant amplitude points due to interference or poor fringe fits (among other things). The task WIPER makes UVPLT like plots but is designed to do flagging.

C.10 Incorporating non-VLBA antennas

Many non-VLBA telescopes have their amplitude calibration incorporated in the tables loaded by FITLD. We retain the following sections for the case where some telescopes are not included in calibration tables for whatever reason.

The phased-EVLA came on-line in February 2013. Its calibration is very different from the old VLA. With the phased-EVLA either an ANTAB style file was provided to the observer or the calibration information is attached to the data. This calibration is generated from the switched power measurements attached to the EVLA only data. There is no need for special calibration steps. However, the observer may wish to improve the calibration by editing the switched power and recreating the ANTAB style file. More information on this can be found in § 9.5.2.4. There is a separate section (§ C.11) dealing with pre-EVLA VLA data.

C.10.1 Loading $T_{\rm sys}$ and Gain Curves

Most telescopes that do not have $T_{\rm sys}$ and gain curve information attached to the data will provide the user with a text ANTAB format file that can be loaded with ANTAB. Please see EXPLAIN ANTAB for more information on the ANTAB format. Follow the steps below to load calibration information and amplitude calibrate the data.

1. To prevent any chance of having to re-run FITLD, first save the VLBA TY and GC (and all your other) tables by running TASAV. Then run ANTAB with CALIN set to the ANTAB file and setting TYVER and GCVER to the highest numbered tables (usually 1). You don't want to leave TYVER and GCVER equal to 0 because that causes ANTAB to create new TY and GC tables with only the antennas in the file you are loading. If ANTAB fails, it is most likely caused by the input file not being formatted correctly. Perhaps you needed to add an INDEX entry or the file wasn't in true ANTAB format. Also, if ANTAB fails check that the TY and GC tables have not been corrupted. If they have you should delete the tables and copy the original tables from the TASAV'ed file with TACOP.

2. Now, run VLBAAMP as described in § C.7 to combine the gain and system temperature information for all antennas into the appropriate SN and CL tables. Then, use SNPLT or VLBASNPL as described in § C.7 to make sure that the resulting SN now contains amplitude calibration for all the antennas and IFs included in the project.

C.10.2 Pointing Flags

The VLA, GBT, AR and possibly other telescope's on-line systems produce only recorder-related flags, not pointing flags. Thus, for example, there are no flags for when the telescope is not on-source. This can lead to a large amount of bad data especially if you are changing source frequently. However, the VLBI scheduling program SCHED creates a *.flag file which contains estimates of how long it takes these antennas to slew. The *.flag file will also contain flags for all antennas in the experiment, so it is best to remove the flags that pertain to the VLBA antennas. The flag file can be read in with UVFLG using the INTEXT adverb.

C.11 Pre-EVLA VLA data

The method described in this section is only meant for VLA data from before February 2013 (before the phased-EVLA came on-line). The observation being calibrated may have incorporated either a single VLA antenna or the phased VLA, but the amplitude calibration parameters for the VLA were not transferred automatically. You will need to obtain an input text file for the VLA then run ANTAB before APCAL. Before February 2013, the gains and system temperatures for this file, in an appropriate format, were supplied in a file called *xxxxx*cal.y.gz', where '*xxxxx*' is the observation code (*e.g.*, 'bm120'), located at https://www.vlba.nrao.edu/astro/VOBS/astronomy/mmmyy/xxxxx/. That file contains instructions on editing the file to get correct inputs. For a phased array or a 1.3-cm observation in which 3 antennas are used, follow the instructions in § C.11.2; for a single antenna, use § C.11.1.

C.11.1 Single VLA Antenna

The method described in this section is only meant for VLA data from before February 2013. Depending on the age of your observation, you may have to add an INDEX entry within the TSYS card (**Do not separate the INDEX entry from the TSYS entry by a "/" !!!**), un-comment the GAIN line for your particular observing frequency, and un-comment the TSYS line. There are examples of INDEX entries in the comments at the head of the file.

Beginning in June 2003, the INDEX, GAIN, and TSYS information in this table are reformatted to be directly acceptable to *AIPS*. You should check the times in the text file to make sure that your observation has been properly described. Only a few special cases will require editing of the file; in most cases you are able to invoke ANTAB with no editing. Once you are satisfied with the ANTAB file, load the data following the directions in § C.10.1.

C.11.2 Phased VLA

The method described in this section is only meant for phased-VLA data from before February 2013. The VLA may be phased on a program source ('STRONG'), or may be phased on a phase-reference source ('CAL-PHASE'), with the resulting solutions applied to the program source ('TARGET'). Rather

than recording a system temperature, the VLA system will record a ratio of antenna temperature to system temperature, which will vary as the array phases up. In order to convert the ratio of antenna and system temperatures to a usable gain, the flux density of some source will be needed.

1. Load and calibrate the VLA data by standard means (see Chapter 4 and Appendix O). Determine the flux density of a relevant strong source, usually either 'STRONG' or 'CAL-PHASE'. Then, on the VLBI data set, insert the flux density of this source into the SU table using SETJY. For example, if the source is 'CAL-PHASE' and its flux density is 0.432 Jy, run SETJY with SOURCES = 'CAL-PHASE'; BIF = 0; EIF = 0; ZEROSP = 0.432,0; OPTYPE = ' '.

2. Edit the input file as indicated above for a single VLA antenna. Again, an INDEX line, a GAIN line, and a TSYS line must be checked (after June 2003) or be created or un-commented. The GAIN line is independent of observing band (the source flux is used to determine the gain), and the TSYS line should include the parameter 'SRC/SYS', indicating that the ratio of antenna temperature to system temperature is being supplied.

3. Run ANTAB to read in the input file of amplitude calibration parameters. Then run VLBACALA to put this in an SN table. Both steps are essentially the same as for a single VLA antenna (see § C.11.1). The most likely problem is that APCAL in VLBACALA will fail because you forgot to enter a source flux density using SETJY, although the error message may not always make this obvious.

4. Run VLBASNPL or SNPLT to inspect the resulting SN table, as for the single VLA antenna. In this instance, you should see that the phased VLA is very sensitive. If the phasing worked well at centimeter wavelengths, the amplitude should be near 4 or 5 instead of the value of 17 or 18 seen for a single VLBA antenna. At the start of scans where the VLA is being phased, you may see a rapid change in the amplitude gain (toward smaller numbers) as the antenna phases are brought into alignment. The SN table should be inspected very carefully, because there may be data that should be flagged when the VLA phasing did not work well. Three possible reasons for poor phasing are (1) the source is too weak; (2) the troposphere is misbehaving; or (3) there was radio-frequency interference at the VLA.

C.12 Summary for non-VLBA antennas

Following the insertion of the amplitude solutions for the non-VLBA antennas, you can return to follow the standard path for calibration of VLBA data. Although the procedures from here on are identical to the VLBA-only case, the observer may wish to pay attention to several issues.

1. Many non-VLBA antennas are more sensitive than a single VLBA antenna, it can be a good idea to use the most sensitive antenna as the reference antenna for fringe-fitting.

2. Non-VLBA antennas may have a larger delay offset than VLBA antennas. The user should pay close attention to the fringe fits, and be aware of the possibility that non-VLBA antennas may have larger residual delays and rates than a VLBA antenna.

3. Many non-VLBA antennas do not slew as rapidly as the VLBA. The FG table supplied by calibration transfer may not include all the on-line flags, and may not incorporate information about the pointing of non-VLBA antennas, and when they arrive on source. Therefore, some judicious flagging by the user may be necessary. See § C.10.2 for a discussion of applying flags produced by the scheduling software SCHED.

4. The elevation limit for non-VLBA antennas is generally different and usually higher than the VLBA's limit ($> 2^\circ$) which may cause sources to rise and set at different antennas at different times. For example, the VLA has an elevation limit of 8°, this means that a source will rise later and set earlier at the VLA than it does at Pie Town or Los Alamos.

C.13 Some Useful References

1. Chatterjee, S., "Recipes for Low Frequency VLBI Phase-referencing and GPS Ionospheric Correction," VLBA Scientific Memo No. 22, May 1999.
`https://library.nrao.edu/vlbas.html`

2. Ulvestad, Jim, "VLBA Calibration Transfer with External Telescopes, Version 1.1," VLBA Operations Memo No. 34, July 30, 1999. `https://library.nrao.edu/vlbao.shtml`

3. Ulvestad, Jim, "A Step-By-Step Recipe for VLBA Data Calibration in *AIPS* Version 1.3" VLBA Scientific Memo No. 25 (the basis of this appendix), January 2, 2001.
`https://library.nrao.edu/vlbas.html`

4. Ulvestad, Jim, Greisen, Eric W., Mioduszewski, Amy "*AIPS* Procedures for Initial VLBA Data Reduction, Version 2.0" *AIPS* Memo No. 105 April 26, 2001.
`https://www.aips.nrao.edu/aipsdoc.html # MEMOS`

5. Wrobel, J. M., Walker, R. C., Benson, J. M., & Beasley, A. J., "Strategies for Phase Referencing with the VLBA," VLBA Scientific Memo No. 24, June 2000.
`https://library.nrao.edu/vlbas.html`

C.14 Additional recipes

C.14.1 Orange baked bananas

1. Mix in a saucepan 1/2 cup firmly packed **brown sugar**, 1 tablespoon **cornstarch**, 1/8 teaspoon **cinnamon**, and a few grains **salt**.
2. Add gradually, blending in 3/4 cup boiling water.
3. Bring rapidly to boiling and cook about 5 minutes or until sauce is thickened, stirring constantly.
4. Remove from heat and blend in $1\frac{1}{2}$ teaspoons grated **orange peel**, 1/4 cup **orange juice**, 1 teaspoon **lemon juice**, and 2 tablespoons **butter**.
5. Peel and cut into halves lengthwise 6 **bananas** with all-yellow or green-tipped peel.
6. Arrange halves cut side down in baking dish and brush with about 2 tablespoons melted **butter**.
7. Sprinkle 1/2 teaspoon **salt** over bananas and then pour the orange sauce over bananas.
8. Bake at 375° F for 10 to 20 minutes.

C.14.2 Mexican chicken vegetable soup with bananas

1. In large, covered kettle, over medium-low heat, simmer 4 pounds cut up **stewing chicken**, 1/c cup coarsely chopped **onion**, 1 teaspoon **salt**, and 4 cups of hot **water** for 2 hours or until chicken is tender.
2. Remove chicken to cutting board; cut meat from bones into chunks; discard bones. Skim any fat from surface of broth.
3. Add chicken, 1/2 cup chopped **celery**, 1 12-ounce can whole-kernel **corn** and 1 16-ounce can **tomatoes** to soup. Continue simmering, covered for 10 minutes. Season to taste.
4. Five minutes before serving, peel 4 firm (green-tipped) **bananas**, slice diagonally into 1-inch slices.
5. Add sliced bananas to soup, continue cooking just until bananas are tender. Serve immediately.

Thanks to Turbana Corporation (`www.turbana.com`).

C.14.3 Orange gingered bananas

1. Combine in a small saucepan 1/4 cup **orange juice** and 1/2 teaspoon **cornstarch**. Cook and stir over medium heat until boiling.
2. Add 1/4 cup **orange juice**, 1 1/2 teaspoons **honey**, and 1 1/2 teaspoons chopped **crystallized ginger** and cook, stirring, until thoroughly heated.
3. Place 2 peeled, green-tipped **bananas** in a shallow baking dish and cover with sauce.
4. Bake at $350\,\mathrm{deg}$ about 15 minutes or until the bananas are tender (but not soft), basting with the sauce several times.

C.14.4 Frozen Push-Ups

1. Peel 2 **bananas** and slice into blender or food processor.
2. Add 1 6-ounce can frozen **orange juice** (thawed), 1/2 cup instant non-fat **dry milk**, 1/2 cup **water**, and 1 cup plain low-fat **yogurt**.
3. Cover and blend until foamy. Pour into small paper cups and freeze.
4. To eat, squeeze bottom of cup.

Thanks to Ruthe Eshleman *The American Heart Association Cookbook*.

C.14.5 Banana mallow pie

1. Combine 2 cups **vanilla wafer** crumbs and 1/3 cup melted **butter**. Press into 9-inch pie plate and bake at 375° F for 8 minutes.
2. Prepare a 3 1/8 ounce package **vanilla pie filling** using 1 3/4 cup **milk**. Cover surface with transparent wrap and chill.
3. Fold 1 1/2 cups **mini-marshmallows** and 1 cup **Cool Whip** into pie filling.
4. Slice 2 **bananas** into pie crust, pour filling over bananas, and chill several hours or overnight.

D Hints for Reducing High-Frequency VLA Data in *AIPS*

D.1 Complications at higher frequencies

High-frequency data (22 or 43 GHz) from the VLA may be reduced occasionally with the standard centimeter-wavelength recipe given in this *CookBook*, particularly in the smaller arrays. However, quire frequently,, the standard recipe will be inadequate for such data, particularly in the larger (A and B) array configurations. Nevertheless, VLA data taken at these high-frequencies in the largest array configurations can be calibrated in almost all cases with only a few minor adjustments to the centimeter wavelength recipe.

One reason for more complicated calibration is the high resolution, which resolves the standard flux density calibrators, particularly 3C48. However, most of the problems are caused by the atmosphere, where the troposphere introduces rapid phase fluctuations between the antenna elements of the interferometer. Both effects scale with baseline length expressed in units of wavelength, but the latter also heavily depends on the current weather; phases are sometimes observed to wind on time scales of less than a minute. This causes decorrelation during your calibrator and target source scans, and requires you to determine phase-only calibration, before the flux density (*i.e.*, gain) calibration should be attempted.

In this appendix, an approach to reducing high-frequency VLA data in *AIPS* is described which should help to overcome the most common problems. It is assumed that the reader has some experience with reducing data in *AIPS*, and is familiar with the "standard recipe" (*e.g.*, Appendix A), tools to examine the data, to apply self-cal, and if appropriate, to deal with spectral-line and polarization calibration issues. If not, you should read the *AIPS CookBook* first (in particular Chapter 4).

High-frequency calibration begins when loading the data, requiring specific parameters in FILLM to be set (ideally one should use these FILLM inputs for all frequencies).

Run FILLM with:

> DOWEIGHT 1 ↵	to apply T_{sys} weights for each individual IF and polarization.
> DOUVCOMP −1 ↵	to store the data without compression, which discards individual IF weights.
> CPARM 0 ; CPARM(8) 0.05 ↵	to use a short time interval in the CL table entries (in min); 0.05min = 3s.
> BPARM 0 ; BPARM(10) 0.75 ↵	to apply opacity and gain curve corrections with zenith opacity weighted 75% by the measured weather and 25% by seasonal averages.

This creates a CL table that can be interpolated over very short intervals, hopefully short enough to cover the atmospheric phase fluctuations accurately. The default CL table interval is 5 minutes, which may be fine for centimeter wavelengths, but is much too long for proper interpolation of high-frequency phases. Also, you have "nominal sensitivity" weights for individual IF/Pol entries, which reflect sensitivity differences between the receivers, IFs, etc. To retain this "nominal sensitivity" weighting you are required to set DOCALIB=1 (actually $0 < \text{DOCALIB} \leq 99$ and a non-negative value for GAINUSE) in all the calibration tasks during the remainder of the data calibration.

The importance of the CL table interval is illustrated in Figure D.1 and Figure D.2. On the large scale, the phases look beyond redemption. But, on a relatively short time scale, the phases are relatively well behaved and may be calibrated easily.

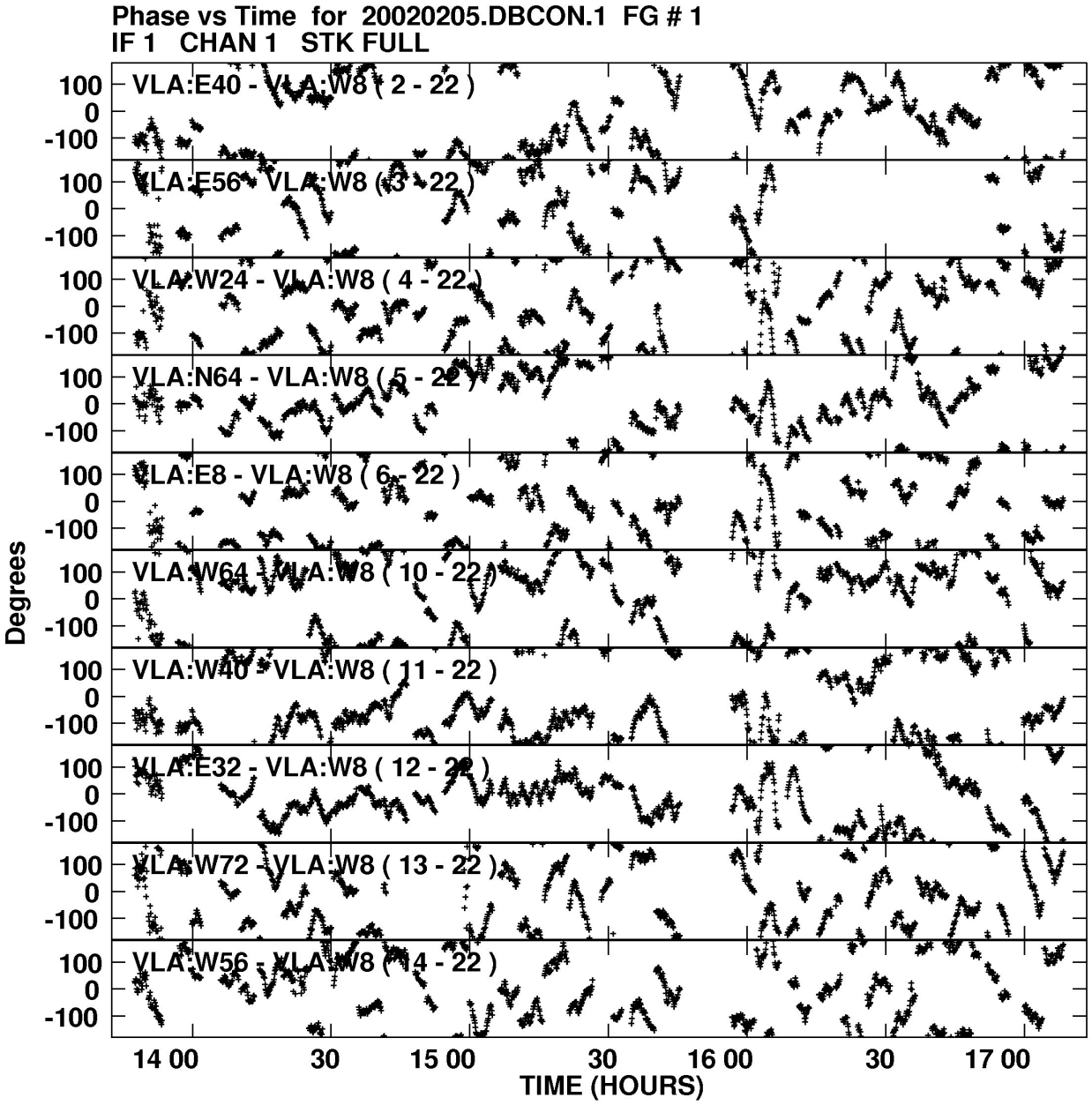


Figure D.1: Uncalibrated VLA A-array 43 GHz phases to the reference antenna in the center of the array (22) over 3 hours on a strong source (frequency switched every 6 minutes) as a function of time plotted by VPLOT look very volatile. Around 16:00 hours they even wrap 360 degrees within one minute. They cannot be calibrated using the default CL table interval of 5 minutes.

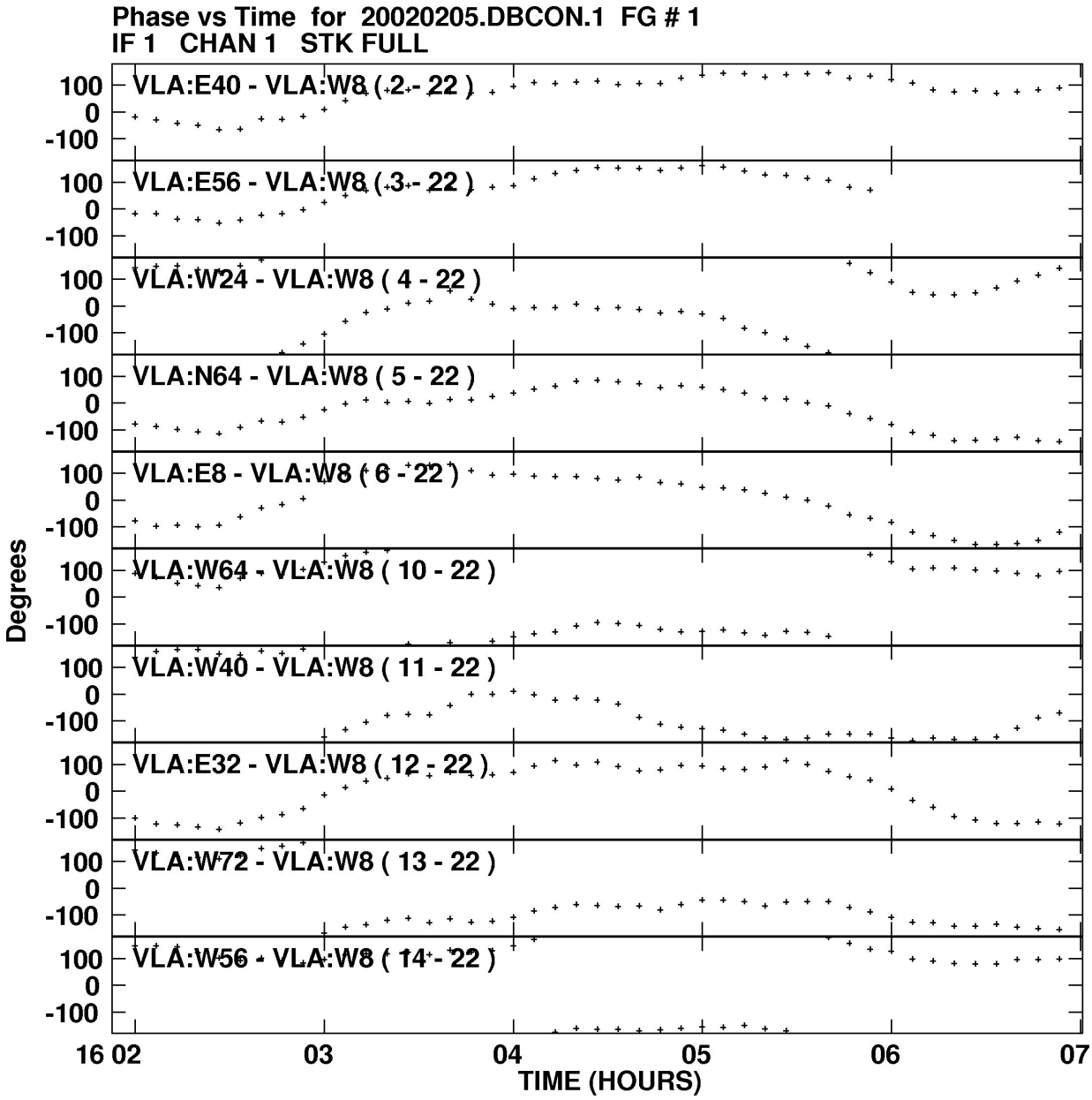


Figure D.2: A blow up from VPLOT of the time range 16:02 to 16:07 from Figure D.1. The uncalibrated visibility phases are seen to be well behaved, albeit on a short time scale. They can be calibrated if the SN table solutions are found on these typical short time scales and are interpolated with a CL table that has sufficiently short spacing between entries to allow for interpolation of the rapid phase fluctuations. An interval of 20 seconds would be okay here.

After loading your data, check your CL table entries, *e.g.*, LISTR with OPTYPE 'GAIN', PRTAB with DOHMS 1, or SNPLT on a short (few minutes) time range with OPTYPE 'AMP'. Make sure the entries are at the interval you expect (much less than a minute) and that the opacity and gain curve corrections have been applied (gains deviating from one by a few percent). Inspect your continuum or "channel 0" data (gains, system temperatures), and flag bad data. For example, you also may wish to flag antenna 1, which is known to have bad optics at 43 GHz, and the antennas without a 43 GHz receiver (currently in December 2002, antennas 9 and 15, but for earlier observations you may want to check the receiver status page (`https://www.vla.nrao.edu/astro/guides/highfreq/`), or your observation log — these antennas may have been left present in your data when you first do a pointing scan in X-band). Standard tools for data inspection and flagging are described in Chapter 4 of the *AIPS CookBook* (LISTR, UVPRT, VPLOT, SNPLT, SNBLP, UVFLG, TVFLG, EDITR, EDITA, and many more). Make sure that at least your calibrators are "clean." Run VPLOT on your calibrators with a reference antenna close to the center of the array (determined by using PRTAN) to get an indication how rapidly your phases fluctuate; use ANTENNA *reference_antenna* 0; SOLINT 0; BPARM 0 2 0 (for phase only). If your program source is too weak to allow self-calibration and the phase change from one scan on your calibrator to the next is of the order of 180°, you probably want to flag the source data in between the calibrator scans. Task SNBLP will let you examine how the calibration affects each baseline and task SNFLG may be useful for doing this flagging.

Note that fast-switching, when used, will have changed the source names you used in making the observe schedule file. Your sources will have been renamed to their J2000 positions, making it difficult to recognize the calibrator and target scans when you run LISTR (OPTYPE 'SCAN').

Run SETJY on your absolute flux density calibrator: 3C286 = J1331+305 = B1328+307, or 3C48 = J0137+331 = B0134+329. And maybe it is a good idea to make a copy of your correct CL table number one (actually all tables) with TASAV before continuing, so in case of accidents, you have CL table one with the opacity/gain corrections applied. (INDXR may be used to re-create a CL table, including the opacity and gain corrections made by FILLM.

Run VLANT to correct phases for improved estimates of the antenna positions. Note that this task requires your computer to be connected to the Internet if your data are recent (within past 18 months or so). Otherwise, apply baseline corrections following advice at `www.vla.nrao.edu/astro/archive/baselines/`.

Run CALIB, at this stage to correct for phase only, with a small solution interval (depending on your signal to noise, *e.g.*, 20 seconds) on all your calibrator sources. You should use the Clean-components models for 3C286 or 3C48 provided with *AIPS*. See § 4.3.9.1 for information on CALDIR and CALRD. Then run CALIB on these sources separately using the appropriate model. There are also models for 3C147 and 3C138.

Inputs to the first pass of CALIB:

> CALSOUR '*cal1*', '*cal2*', ... ↵	to define your calibrators; **all but those for which you plan to use a model, *e.g.*, 3C48**.
> DOCALIB 1 ↵	to apply nominal sensitivities, essential that $0 <$ DOCALIB ≤ 99.
> GAINUSE 0 ↵	apply latest CL table (is version 2 after VLANT.
> REFANT *reference_antenna* ↵	to pick a well behaved antenna in the array center.
> SOLINT 20/60 ↵	to solve every 20 seconds; may have to try some values.
> SOLMODE 'P' ↵	to do phase calibration only at this stage.
> SNVER 1 ↵	to collect all solutions in SN table one.

And, if you have 3C48, 3C138, 3C147, and/or 3C286 as absolute flux density calibrator(s), you should re-run CALIB, one calibrator source at a time, with the previous/above values plus:

> CALSOUR '*ssss*', ' ' ↵	to specify the name you have used for the calibrator source.
> IN2DISK *d2* ↵	to specify the disk with the source model.

> GET2NAME *ctn2* ↵	to specify the CC model to be used by its catalog number.
> INVERS 0 ↵	to use the model's latest CC-version (*i.e.*, one).
> NCOMP 0 ↵	to use all the CC-components of the model.

This may work, but there is no guarantee. Some tricks to apply, in no particular order, in your data set or CALIB to obtain a larger relative portion of good versus bad solutions would be:

- Flag some more bad data points on your calibrator sources.
- Discard antennas with uncertain baseline positions (see observing log file).
- Choose a different reference antenna (the one you have might be misbehaving).
- Decrease the UVRANGE to weight short baselines (centrally located antennas) more in the solution.
- Use SOLTYPE 'L1' to be less sensitive to outlying points.
- Use FRING instead of CALIB with a larger SOLINT to solve for the phase rates, switching off the delay search with DPARM(2) = -1.
- Increase or decrease SOLINT; increase for weak, decrease for strong sources.
- Decrease the SNR cutoff APARM(7) (default 5) to include more noisy but possibly valid solutions.
- Decrease the number of antennas required for a solution (APARM(1), default 6) to require fewer antennas
- Recreate 'CH 0' from 'LINE' to get up to 25% more bandwidth on calibrators.

Note that at 43 GHz in A-array the unprojected *uv*-distance between the outer two antennas on one arm is 0.5 Mega-wavelengths, and the outer 6 antennas — the default for APARM(1) — require good solutions out to 2 Mega-wavelengths for CALIB to accept the solution for your outermost antenna. Hence, it is a good idea to set APARM(1) to *e.g.*, four (or three, if you're willing to check the output SN table carefully).

Check the resulting SN table number one with LISTR (OPTYPE 'GAIN', DPARM 1 0) or SNPLT (INEXT 'SN', OPTYPE 'PHAS'), and judge whether you have enough solutions and whether you believe the phases shown are likely to reflect the variation caused by the troposphere. If not, fiddle around with your data and/or parameters in CALIB as suggested above and try again. In case the majority of solutions are fine, you may want to edit spurious points in your SN table with *e.g.*, SNEDT, SNCOR, or SNSMO.

Once you are satisfied with the phases in your SN table, you want to apply phase corrections to minimize decorrelation in your calibrator scans before you determine the absolute flux density scale. To insert the corrections, run CLCAL with:

> SOURCES '' ↵	to correct phases for all sources.
> CALSOUR '*cal1*', '*cal2*', ... ↵	to include **all** your calibrators.
> INTERPOL '2PT' ↵	to interpolate between solutions ('SIMP' will average phases over a scan).
> SNVER 1; GAINVER 1;GAINUSE 2 ↵	to apply SN#1 to CL#1, creating CL#2.
> REFANT *reference_antenna* ↵	to select the same antenna as used in CALIB.

In less straightforward observations you may not be able to run CLCAL only once, *e.g.*, when you are switching frequencies. If in doubt, consult Chapter 4 of the *AIPS CookBook*. It is however very simple to run CLCAL multiple times. Inspect your new CL table two (three after VLANT) for unexpected dubious interpolations and extrapolations (LISTR with OPTYPE 'GAIN', DPARM 1 0, or SNPLT with INEXT 'CL',

OPTYPE 'PHAS') and backtrack possible problems. SNBLP should help you to find problem solutions as they affect your data.

Now re-run CALIB with the corrected phases to obtain the flux density scale. Begin with those sources not requiring models:

> CALSOUR '*cal1*', '*cal2*', ... CR	to identify your calibrators; all, except 3C48 and 3C286.
> DOCALIB 1 CR	to apply antenna gain/opacity and antenna location corrections to the data and their weights.
> GAINUSE 0 CR	to apply latest CL table (is 3 here).
> REFANT *reference_antenna* CR	to pick the same antenna as used before.
> SOLINT 0 CR	to average over the full scan; remember that phase variations are corrected by DOCALIB.
> SOLMODE 'A&P' CR	to do full calibration to get flux densities and residual phases.
> SNVER 2 CR	to collect solutions in a new SN table (two).

Then run CALIB again for the absolute flux density calibrator 3C286, or 3C48, using a model and the previous/above values plus:

> CALSOUR '*sssss*', ' ' CR	to specify the name you have for the source.
> UVRANGE 0 CR	to use the full uv range without restrictions.
> IN2DISK *d2* CR	to specify the disk with the source model.
> GET2NAME *ctn2* CR	to specify the model to be used by its catalog number.
> INVERS 0 CR	to use the model's latest CC-version (*i.e.*, one).
> NCOMP 0 CR	to use all the CC-components of the model.

The same tricks may be used as for phase-only to improve the ratio of good to bad solutions. Check your SN table 2 thoroughly; the phases **must** be zero or very close to zero (therefore INTERPOL '2PT' is preferred over INTERPOL 'SIMP' in CLCAL), and you want to make sure the gains of your reference antenna do not scatter too much for individual sources. Before GETJY the flux density scale is not fixed so the average gain will depend on source. GETJY corrects this so that the gains for each antenna should be similar for all sources. If you can identify misbehaving antennas, flag them, delete SN table 2 (as it does not overwrite values for which data have been deleted), and re-run CALIB as many times as needed to re-create SN table two. Cautious users will start from the beginning.

Run GETJY to obtain the secondary calibrator flux densities:

> SOURCES '*cal1*', '*cal2*', ... CR	to specify the unknown sources; *not* the primary calibrator(s).
> CALSOUR '3C48', ' ' CR	to specify the source name(s) you have used in SETJY.
> SNVER 2 CR	to point to the flux density/gain solution table.

If you used a model for the flux density calibrator, your flux scale is tied to the flux density of that calibrator in the SU table. The model Clean components are scaled to match that flux.

Carefully note the flux densities reported by GETJY and do not trust these values blindly. LISTR or SNPLT may point out problematic antenna solutions, requiring you to flag some more data and start over. If you flag data, it is best to delete SN table #2. Solutions at the times of deleted data will not be overwritten. It is helpful to know what flux you expect for your secondary calibrators. See the full source list at aips2.nrao.edu/vla/calflux.html. It is particularly helpful to use one or more of the sources regularly monitored by NRAO staff; see html://www.aoc.nrao.edu/~smyers/calibration/ for total flux as well as polarization information. You want to check the values, because sometimes the flux densities deviate considerably from the expected values and make no sense. This could be the case if the pointing solutions that were determined prior to your primary calibrator scan are inappropriate for this particular primary calibrator scan *e.g.*, when it is windy or when the cloud cover on your single primary calibrator scan differs from the cloud cover on the secondary calibrators, or, even worse, when these combine. In some cases

you may be forced to approximate the flux density scale by entering a (recent) flux density for one of your secondary calibrators, ignoring the primary calibrator scan and accepting an introduced flux density uncertainty. If you decide you have to restart, do not forget to delete your SN and CL tables (except for CL table 1 and 2 if you used VLANT) and to reset the flux densities of all your calibrators with SETJY (OPTYPE 'REJY'), before entering a ZEROSP for a new flux density calibrator source (also with SETJY). Re-iterate until you are happy with the flux-density scale.

The final flux density calibration table is obtained by running CLCAL again:

> SOURCES ' ' ↵	to calibrate flux densities for all sources.
> CALSOUR '*cal1*', '*cal2*', ... ↵	to include calibrators to use for your targets.
> INTERPOL '2PT', or 'SIMP' ↵	(to specify the interpolation method: no real difference for per-scan solutions).
> SNVER 2; GAINVER 2; GAINUSE 3 ↵	to apply SN#2 to CL#2, creating CL#3.
> REFANT *reference_antenna* ↵	to select the same antenna as used in CALIB.

From here you are almost ready to follow the usual "standard recipe," *i.e.*, polarization and bandpass calibration if appropriate, and splitting into single-source data sets. However, remember to set DOCALIB = 1 in all these tasks as long as you are working on the multi-source data set and haven't applied initial phase, flux density (including polarization, bandpass) and "nominal sensitivity" calibration with SPLIT. After SPLIT, the individual weights will have been entered in the data, properly scaled by the latest CL table you've made. Using your single source calibrated data, set DOCALIB = -1 in your subsequent imaging and analysis tasks, unless you do self-calibration.

If you anticipated checking your fast-switching calibration by including a "check source" (a moderately strong source observed a few times with the same fast-switching parameters at about the same distance from your fast-switching source as your target source, but not necessarily in the same direction), you can now assess a snapshot of your calibration by imaging this source. If the fast-switching has worked perfectly, your check source has the expected morphology, flux density, and position. Any position error on the check source should indicate the accuracy of the astrometry on your target source. If you did not include a check source, all but the astrometry and spatial dependence of the calibration can be inferred from your fast-switching source by imaging a scan (use a modified SN/CL table by skipping the calibration on this scan) with the calibration derived from the two neighboring scans.

D.2 Additional recipes

D.2.1 Chocolate chip banana bread

1. Blend 2 cups mashed **bananas**, 1 tablespoon grated **orange peel**, and 1/3 cup **orange juice** in a bowl. Beat in 3 **eggs**. Stir in 1 cup packed **brown sugar** and 1/3 cup **vegetable oil**.
2. Combine 2-1/2 cups **all-purpose flour**, 1 cup **chocolate chips** 2 teaspoons **baking powder**, 1/2 teaspoon **baking soda**, 1/2 teaspoon **salt**, and 1/2 teaspoon **nutmeg**.
3. Stir dry ingredients into banana mixture just until blended. Pour into 4 greased 5-3/4 x 3-1/4-inch loaf pans.
4. Bake in 350° F oven for 45 to 55 minutes or until tester inserted comes out clean. Let cool in pans on rack for 10 minutes. Remove from pan and let cool completely on rack.

Thanks to Tim D. Culey, Baton Rouge, La. (tsculey@bigfoot.com).

D.2.2 Banana bran muffins

1. Preheat oven to 400° F.
2. Grease 12 2.75-inch muffin cups.
3. In bowl, combine 1/2 cup crushed **cereal** (1.5 cups un-crushed Multi-Bran Chex recommended), 1.5 cups all-purpose **flour**, 1/2 cup **sugar**, 1/3 cup chopped **nuts** (optional), 2.5 teaspoons **baking powder**, and 1/2 teaspoon **baking soda**.
4. In a separate bowl, combine 3 large mashed **bananas** (1.5 cups), 1 **egg** slightly beaten, 1/4 cup vegetable **oil**, 2 tablespoons **water**, and 1 teaspoon **vanilla extract**.
5. Add to cereal mixture and stir just until moistened. Do not over-mix.
6. Divide evenly among muffin cups.
7. Bake 18–20 minutes, or until tester inserted in center comes out clean.

Thanks to Ralston Purina Company.

D.2.3 Banana-pineapple rum bread

1. Place 1/2 cup **white rum** and 1/2 cup diced **dried pineapple** in a bowl, cover, and let sit for at least one hour.
2. In a mixing bowl, beat together 4 tablespoon **butter** or margarine and 3/4 cup **sugar**. Add 1 extra large **egg** and continue beating until light and fluffy.
3. Add 2 large mashed ripe **bananas** and mix well. Beat in 1/3 cup plain **yogurt** — curdling of the mixture is normal.
4. In another mixing bowl, combine 2 cups **all-purpose flour**, 1/2 tablespoon **baking soda**, 1 teaspoon ground **cinnamon**, 1 teaspoon ground **nutmeg**, 1 teaspoon ground **allspice**, and 1/2 teaspoon **salt**.
5. Add the wet ingredients and mix until well blended. Drain the pineapple and add. Fold in 1/2 cup coarsely chopped **pecans**.
6. Pour into liberally greased 9-inch loaf pan. Bake at 350° F for 45 to 55 minutes or until the bread passes the toothpick test. Remove the pan from the oven and let it sit for 10 minutes, before turning out on a rack to cool.

Thanks to Tim D. Culey, Baton Rouge, La. (`tsculey@bigfoot.com`).

D.2.4 Dulce Zacatecaño

1. Peel 3 large not-too-ripe **bananas** and slice lengthwise. Saute in 5 tablespoons **butter** until golden brown. Drain on paper, place in a shallow baking dish, and sprinkle with a little **sugar**.
2. Whip 1/2 cup **heavy sweet cream**. Add 1/4 cup **sugar**, 1/4 cup **dry sherry wine**, and 1 teaspoon **vanilla**. Pour over bananas covering them completely. Chill and serve very cold.

Thanks to Ruth Mulvey and Luisa Alvarez *Good Food from Mexico*.

D.2.5 Virginia's instant banana pie

1. Mix 1 cup **sour cream**, 1 cup **milk**, and 1 small package **instant vanilla pudding** until mixture thickens.
2. Slice 3 medium **bananas** into the bottom of a 9-inch **graham cracker pie crust**.
3. Pour the pudding over the bananas and refrigerate at least 2 hours.

O Special Considerations for Data from Older Telescopes in $\mathcal{AIPS}$

Chapter 4 on data calibration has been revised to describe the modern VLA data format and calibration in $\mathcal{AIPS}$. Similarly, old VLBI formats have been excised from Chapter 9 to avoid confusion with more modern practices. Nonetheless, the VLA archive for the pre-expansion VLA is widely and productively used and we suspect that there are VLBI users with older data files which may still need reduction and analysis. This appendix is used to address first the older VLA data format and initial reduction steps and then the older Mk III and Mk IV VLBI formats.

O.1 The Historical VLA

The NRAO archive contains all of the data from the historical (or pre-EVLA) Very Large Array. These data are frequently mined for uses above and beyond those of the original observers. There is a new archive at `https://data.nrao.edu`; the old archive is no longer available. Log in at the above web address using your `my.nrao.edu` account and then select Show Search Inputs. This has all sorts of ways to find what you may desire. You may specify one or more of telescope, project code, date range, target name, target coordinates and search radius, observing band, or telescope configuration. You may submit your query and the tool will show you all data sets that match your specification. You may then request the data sets that you actually want and they will be downloaded to a publicly accessible disk area. All historical VLA data have been in the public domain for a long time, so no special permission should be needed. You will be notified when your data are available and told how to access them.

The old archive contained less well known content which is not available in the new archive. Instead the web site `https://www.vla.nrao.edu/astro/nvas/` describes an archive of $\mathcal{AIPS}$-processed legacy VLA data sets containing calibrated data and images. There is a description of this service and, at the left, buttons to connect to various web pages of informations and also a search tool.

For VLA data from the archive, use `FILLM` to read one or more disk files; see § O.1.1. The VLA format was changed on January 1, 1988, but all older data were translated and archived in the modern format. On July 1, 2007, the ModComps were replaced with modern computers and the format had an essential change made to it. Use `FILLM` to read data from both the ModComp and post-ModComp eras.

For VLA calibration, there are several useful procedures described in this chapter and Chapter 4. They are contained in the `RUN` file called `VLAPROCS`. Each of these procedures has an associated `HELP` file and inputs. Before any of these procedures can be used, this `RUN` file must be invoked with:

> RUN VLAPROCS ↵ to compile the procedures.

There is a "pipeline" procedure designed to do a preliminary calibration and imaging of ordinary VLA data sets. This provides a good first look at the data. Nonetheless, the results are still not likely to be of publishable quality. To run the pipeline, enter

> RUN VLARUN ↵ to compile the procedures.
> INP VLARUN ↵ to review the input adverbs and, when ready,
> VLARUN ↵ to execute the pipeline.

See Appendix A for a simplified summary of data reduction suitable to data from the historic VLA. Much of the calibration of historic VLA data is similar to that of the modern EVLA, although the latter is fundamentally multi-channel and wide band. See Chapter 4 for a discussion of the steps used in calibrating

all VLA data.

O.1.1 Reading from VLA archive files using FILLM

The NRAO Archive makes available, among other things, data from the EVLA, which began observing in January 2010, and from the old VLA which ceased observing a few days into 2010. To load data from the EVLA into *AIPS* consult Chapter 4 for information about BDF2AIPS and other options. The following describes how to read old VLA data into *AIPS*.

The Archive now serves VLA data in the form of one or more "MOdComp" format disk files. To load these into *AIPS*, enter

> TASK 'FILLM' ; INP ↵	to review the inputs needed.
> DATAIN 'MYDATA:AC238_' ↵	to read from the disk area pointed at by the logical MYDATA and the data from program ID AC238.
> NFILES 0 ↵	to start with file AC238_1. If your first file is *e.g.*, AC238_4, set NFILES = 3.
> NCOUNT 3 ↵	to read three data files AC238_1 through AC238_3.
> OUTNA ' ' ↵	to take the default output file name.
> OUTDI 3 ↵	to write the data to disk 3 (one with enough space).
> DOUVCOMP -1 ↵	to write visibilities in uncompressed format. VLA files are small by modern standards, so saving space is not worth the costs.
> DOWEIGHT 1 ↵	Data weights will depend on the "nominal sensitivity" and should be calibrated along with the visibility amplitudes (DOCALIB = 1).
> CPARM 0 ↵	to do no averaging of the data in FILLM.
> CPARM(6) 1 ↵	to select VLA sub-array 1.
> CPARM(7) 2000 ↵	to have observations within 2 MHz be regarded as being at the same frequency.
> CPARM(8) 1 ↵	to use a 1-minute interval for the CL table; default is 5 min.
> CPARM(9) 0.25 ↵	to use a 15-second interval for the TY table; default is the input data interval.
> DPARM 0 ↵	to have no selection by specific frequency.
> REFDATE '*yyyymmdd*' ↵	to specify the year, month, and day of the reference date. This should be the first date in the data set (or earlier). All times in *AIPS* will be measured with respect to that date and must be positive. The default is the first date included by the data selection adverbs, which may not be the desired one. Note that REFDATE is only a reference point; it does not affect which data are loaded from the the files.
> TIMERANG *db* , *hb* , *mb* , *sb* , *de*, *he* , *me* , *se* ↵	to specify the beginning day, hour, minute, and second and ending day, hour, minute, and second (wrt REFDATE) of the data to be included. The default is to include all times.
> INP ↵	to review the inputs.
> GO ↵	to run the program when you're satisfied with inputs.

In 31DEC21, FILLM first tries to open the file under the name in DATAIN. If that fails, it then tries the name in DATAIN with the file number appended as described above.

There are numerous adverbs including BAND, QUAL, CALCODE, VLAOBS. and VLAMODE to limit what data were loaded from magnetic tapes which could hold data from multiple projects. These adverbs still function, but are of little use today. Note that the values given above are illustrative and should not be copied verbatim in most cases.

Be careful when choosing the averaging time with CPARM(1). If you have a large data set, setting this time too *low* will make an unnecessarily large output file; this may waste disk space and slow the execution of subsequent programs. Setting it too *high* can, however, (1) smear bad data into good, limiting the ability to recognize and precisely remove bad data, (2) smear features of the image that are far from the phase center, and (3) limit the dynamic range that can be obtained using self-calibration. If you need a different (usually shorter) averaging time for the calibrator sources than for your program sources, use CPARM(10) to specify the averaging time for calibrators. See Lectures 12 and 13 in *Synthesis Imaging in Radio Astronomy*[1] for general guidance about the choice of averaging time given the size of the required field of view and the observing bandwidth.

CPARM(2) controls a number of mostly esoteric options. If your data include the Sun (see § O.1.8) or planets, you must set CPARM(2) = 16 to avoid having each scan on the moving source assigned a different name. The adverb DOWEIGHT = 1 has the same affect as CPARM(2) = 8 and both select the use of the nominal sensitivity to scale the data weights. When this is done, the weights will be $1/\sigma^2$ as they should for imaging, with σ in "Jy" in the same uncalibrated scale as the fringe visibilities. Having selected this option, you should apply any amplitude calibration to the weights as well as the visibilities. If you store the data in compressed form, only one weight may be retained with each sample. Any differences between polarizations and/or IFs in that sample will be lost. Uncompressed data require less CPU, but more real, to read but 2 to 3 times as much disk space to store.

CPARM(2)=2048 allows you to load data as correlation coefficients, which can be scaled to visibilities later with TYAPL (§ O.1.3). CPARM(3) controls which on-line flags are applied by FILLM, which now always writes an OF table containing information about these flags. That information can be viewed with PRTOF and applied selectively to the data at a later time with OFLAG.

FILLM writes a weather (WX) table to the output file. At the same time, it uses "canned" VLA antenna gain curves and a balance of the current with a seasonal model weather data to estimate opacity and gain corrections to be written into the first calibration (CL) table. These functions are controlled by adverbs CALIN and BPARM and may be turned off, although the default is to make the corrections. In subsequent tasks, set DOCALIB = 1 to use these initial calibration data. If, for some reason, the data weights do not depend on the nominal sensitivity, use DOCALIB=100 to apply calibration.

Where possible, FILLM will try to place all data in one file. However, in many cases this is not possible. For instance so-called "channel 0" data from a spectral-line observation will be placed in a separate file from its associated line data. Similarly, scans which have differing numbers of frequency channels will also be placed into separate files. Another case is observations made in mode LP, *i.e.*, one IF-pair is set to L band, the other to P band. In this case the two bands will be split into separate files. Yet another case arises when there are observations of different bandwidths. All of this should be relatively transparent to the user.

FILLM and many *AIPS* tasks are able to handle multiple, logically different, frequencies within a multi-source data set. FILLM does this by assigning an FQ number to each observation and associating a line of information about that frequency in the FQ file associated with the data set. Users should note that this concept can become quite complicated and that not all tasks can handle it in full generality. In fact, most tasks can only process one FQ number at a time. Polarization calibration works only on one FQ at a time since the antenna file format allows for only one set of instrumental polarization parameters. Therefore, it is *strongly* advised that you fill continuum experiments which involve multiple frequencies into separate data sets. FILLM will separate bands automatically, but you will have to force any remaining separation. To

[1] *Synthesis Imaging in Radio Astronomy*, Astronomical Society of the Pacific Conference Series, Volume 6 "A Collection of Lectures from the Third NRAO Synthesis Imaging Summer School" eds. R. A. Perley, F. R. Schwab and A. H. Bridle (1989)

do this, (a) use the QUAL adverb in FILLM, assuming that you have used separate qualifiers in OBSERVE for each frequency pair; (b) use the DPARM adverb array in FILLM to specify the desired frequencies precisely; or (c) use the UVCOP task to separate a multiple FQ data set into its constituent parts. Note that the first two options require multiple executions of FILLM, while the third option requires more disk space.

Spectral-line users and continuum observers using different frequencies in the same band should be aware of the FQ entry tolerance. Each frequency in a *uv* file will be assigned an FQ number as it is read from disk by FILLM. For spectral-line users, the observing frequency will normally change as a function of time due to Doppler tracking of the Earth's rotation, or switching between sources or between spectral lines; in general, this will cause different scans to have different FQ numbers. FILLM assigns an FQ number to a scan based on the FQ tolerance adverb CPARM(7) which defines the maximum change of frequency allowed before a new FQ number is allocated. If CPARM(7) < 0, the the same FQ number is assigned to all data in spectral-line data sets. If CPARM(7) is positive, a scan's will be assigned to an existing FQ number if

$$\|\nu_{current} - \nu_{firstFQ}\| < \texttt{CPARM}(7)$$

where $\nu_{firstFQ}$ is the frequency of the first sample to which the particular FQ number was assigned. If no match is found, then a new FQ number is created and assigned and another line added to the FQ table file. Alternatively, if CPARM(7) is zero, then the FQ tolerance is assumed to be half of the maximum frequency difference caused by observing in directions 180 degrees apart (*i.e.*, $\Delta\nu = 10^{-4} \times \nu$).

An example: if an observer observes the 1612, 1665 and 1667 MHz OH masers in VY CMa and NML Cygnus, then presumably he would like his data to have 3 FQ numbers, one associated with each OH transition. However, running FILLM with CPARM(7) set to 0 would produce 6 FQ numbers because the frequency difference between the masers in VY CMa and NML Cygnus is greater than the calculated tolerance of 160 kHz. Therefore, in order to ensure that only 3 FQ numbers are assigned, he should set CPARM(7) to 1000 kHz. Setting CPARM(7) < 0 would result in all data having the same FQ number, which is clearly undesirable.

For most continuum experiments the FQ number will be constant throughout the database. Normally any change in frequency should be given a new FQ number. To achieve this, FILLM treats CPARM(7) differently for continuum. If CPARM(7) $\leq$ 0.0, then FILLM assumes a value of 100 kHz. A positive value of CPARM(7) is treated as a tolerance in kHz as in the spectral line case.

Note: *If your uv database contains several frequency identifiers, you should go through the calibration steps for each* FQ *code separately.*

FILLM can still read from magnetic tape. Set DATAIN to blanks, mount your tape (adverb INTAPE), index the tape with PRTTP, and use the adverbs to limit the data loaded to that portion of your project in which you are interested.

If FILLM is executing correctly, your message terminal will report the number of your observing program, the VLA archive format revision number, and then the names of the sources as they are found in the data files. Once FILLM has completed, you can find the database on disk using:

> INDI 0 ; UCAT ↵

This should produce a listing such as:

```
Catalog on disk  3
Cat Usid Mapname      Class  Seq  Pt     Last access      Stat
  1  103 25/11/88    .X BAND.   1 UV 26-JAN-2018 12:34:16
```

You might then examine the header information for the disk data set by:

> INDI 3 ; GETN 1 ; IMHEAD ↵

This should produce a listing like:

```
AIPS 1: Image=MULTI     (UV)         Filename=25/11/88    .X BAND.   1
AIPS 1: Telescope=VLA                Receiver=VLA
AIPS 1: Observer=AC238               User #=   36
AIPS 1: Observ. date=25-NOV-1988     Map date=26-JAN-2018
AIPS 1: # visibilities   2613887     Sort order  TB
AIPS 1: Rand axes: UU-L-SIN  VV-L-SIN  WW-L-SIN  BASELINE  TIME1
AIPS 1:            SOURCE  FREQSEL
AIPS 1: ----------------------------------------------------------------
AIPS 1: Type    Pixels   Coord value     at Pixel     Coord incr   Rotat
AIPS 1: COMPLEX      1   1.0000000E+00       1.00  1.0000000E+00    0.00
AIPS 1: STOKES       4  -1.0000000E+00       1.00 -1.0000000E+00    0.00
AIPS 1: FREQ         1   8.4110000E+09       1.00  1.2500000E+07    0.00
AIPS 1: IF           2   1.0000000E+00       1.00  1.0000000E+00    0.00
AIPS 1: RA           1    00 00 00.000       1.00       3600.000    0.00
AIPS 1: DEC          1    00 00 00.000       1.00       3600.000    0.00
AIPS 1: ----------------------------------------------------------------
AIPS 1: Coordinate equinox 1950.00
AIPS 1: Maximum version number of extension files of type HI is   1
AIPS 1: Maximum version number of extension files of type AN is   1
AIPS 1: Maximum version number of extension files of type NX is   1
AIPS 1: Maximum version number of extension files of type SU is   1
AIPS 1: Maximum version number of extension files of type FQ is   1
AIPS 1: Maximum version number of extension files of type CL is   1
AIPS 1: Maximum version number of extension files of type TY is   1
AIPS 1: Maximum version number of extension files of type WX is   1
AIPS 1: Maximum version number of extension files of type OF is   1
AIPS 1: Keyword = 'CORRMODE'  value = '        '
AIPS 1: Keyword = 'VLAIFS  '  value = 'ABCD    '
AIPS 1: Keyword = 'CORRCOEF'  value =           -1
```

This header identifies the file as a multi-source file (`Image=MULTI`) with 2613887 floating-point visibilities in time-baseline (`TB`) order. There are two entries on the `IF` axis. These correspond to the old VLA's "AC" and "BD" IF-pairs respectively. The description of the frequency (`FREQ`) axis shows that the first IF ("AC") is at 8411 MHz and has 12.5 MHz bandwidth. The parameters of the second IF-pair ("BD") are determined from the data in the `FQ` table file and cannot be read directly from this header; these values are shown in the `'SCAN'` listing from `LISTR`. The header shown above indicates that the data are in compressed format since the number of pixels on the `COMPLEX` axis is 1 and the `WEIGHT` and `SCALE` random parameters are not present. Uncompressed data does not use these random parameters and has 3 pixels on the `COMPLEX` axis.

The term "IF" can be confusing. At the VLA, IFs "A" and "C" correspond to right-hand and left-hand circularly polarized (RHC and LHC) signals, respectively, and are normally for the same frequency in an observing band. Such pairs, if at the same frequency, are considered to be one "IF" in *AIPS*. An observation which was made in spectral line mode "`2AC`" is considered at the VLA to have two "IFs" whereas within *AIPS* this would be filled as one "IF" with two polarizations if they were both observed with the same frequency, the same number of channels, and the same channel separation. If these conditions do not hold, then they are filled into separate *uv* files, each with a single IF and a single polarization. The term "sub-array" is also confusing. At the VLA — and in task `FILLM` — sub-array means the subset of the 27 antennas actually used to observe your sources. (The historic VLA allowed up to 5 simultaneous sub-arrays in this sense.) In the rest of *AIPS*, sub-array refers to sets of antennas used together at the same time. If observations from separate times (*e.g.*, separate array configurations) are concatenated into the same file, then *AIPS* will regard the separate sets of antennas as different "sub-arrays" whether or not the same physical antennas occur within more than one of these sub-arrays.

Besides the main *uv* data file, this header listing shows that there are numerous "extension" files attached to the data. These are, in order, the history, antenna, index, source, frequency, calibration, system temperature,

weather, and on-line flags tables. LISTR with OPTYPE='SCAN' provides a useful summary of the index, source, and frequency tables.

If your experiment contains data from several bands FILLM will place the data from each band in separate data sets. Also, if you observed with several sets of frequencies or bandwidths in a given observing run these will be assigned different FQ numbers by FILLM. You can determine which frequencies correspond to which FQ numbers from the 'SCAN' listing provided by LISTR. Line data are divided into the "channel 0" (central $3/4$ of the of the observing band averaged) and the spectra. Data observed in the "LP" mode (or any other two-band mode) will be broken into separate data sets, one for each band.

O.1.2 Reading old spectral-line data

If your spectral-line data are in a VLA archive disk file, they should be read into *AIPS* using FILLM, as described in § O.1.1. FILLM will fill a typical line observation into two files, a large one containing the line data only, and a smaller file containing the "channel-0" data. (Note that FILLM computes channel-0 from the line data rather than using the channel-0 provided by the on-line system.) The standard calibration and editing steps are performed on channel 0 and the results copied over to the line data set. *You must be careful with the tolerance you allow* FILLM *to use in determining the* FQ *numbers. If you desire all of your data to have the same* FQ *number, so that you can calibrate it all in one pass, then set* CPARM(7) *in* FILLM *to an appropriately large value.* If you wish to retain spectral-line autocorrelation data, you must set DOACOR to true.

By default for the VLA, the channel-0 data are generated by the vector average of the central *3/4* of the observing band. If this algorithm is not appropriate for your data, you may generate your own channel-0 data set by averaging only selected channels. You may now select different spectral channels in different IFs. To do this, use the task AVSPC:

> TASK 'AVSPC' ↵	
> INDI *n* ; GETN *m* ↵	to specify line data set.
> OUTDI *i* ; OUTCL 'CH 0' ↵	to specify output "channel-0" data set disk and class.
> ICHANSEL 10, 30, 1, 0, 31, 55, 2, 1 ↵	for example, to average every channel between 10 and 30 in all IFs and also every other channel between 31 and 55, but only in IF 1.
> GO ↵	to create a new channel-0 data set.

You might find this necessary when observing neutral hydrogen at galactic velocities. Most calibrator sources have some absorption features at these frequencies.

O.1.3 Applying nominal sensitivities to historic VLA data

FILLM scales the correlation coefficients by the instantaneous measured "nominal sensitivities," producing data approximately in deci-Jy. The VLA nominal sensitivities are stored in the TY table as "system temperatures" (T_{sys}). For calibration purposes, it is best to have the nominal sensitivities applied, but it may be better to use a clipped and/or time-smoothed version of those sensitivities. If you want to do this, load the T_{sys} data into the TY table with the highest time resolution possible by setting CPARM(9)=0 in FILLM. FILLM can also be told not to apply the nominal sensitivities and therefor produce correlation coefficients by setting CPARM(2)=2048, but this is not strictly necessary. In order to smooth and clip the TY table use the task TYSMO. If you have done editing such as QUACK, it may help to copy the data with UVCOP, applying your flag table not only to the visibilities but also to the TY table (UVCOPPRM(6)=3) before running TYSMO to remove questionable values at the start of scans. Alternatively, SNEDT will apply the data flags to the table allowing you to write a new, cleaned-up version of the table. Then a TY table may be applied (and/or removed) from a data set with TYAPL:

> TASK 'TYAPL' ; INP ↵	to review the inputs needed.
> INDI n ; GETN m ↵	to select the correct data set.
> FREQID 1 ↵	to select FQ number 1.
> INVERS 1 ↵	TY table to remove from data, will only work if data are not already correlation coefficients.
> IN2VERS 2 ↵	smoothed TY table to apply to data, will only work if data is in correlation coefficient form — either initially or after removal of INVERS.
> INP ↵	to re-check *all* the inputs parameters.
> GO ↵	to start the task.

O.1.4 Calibrating historic VLA data

There are many similarities to the calibration of historic and modern VLA data. After reading in the data, you should with both run LISTR for the scan listing, PRTAN for the antenna layout information, and VLANT to correct antenna positions; see § 4.2.2 and § 4.3.1 for details. The general discussion on flagging (§ 4.3.10) applies to both types of data. However, with historic VLA data one does a purely continuum calibration and editing initially. If the observation is of spectral lines, the flag and calibration tables are copied to the line uv data file following the calibration of "channel 0." Thus, the spectral editing tasks such as SPFLG, FTFLG, and RFLAG become useful only in the second half of spectral-line reductions. TVFLG (§ O.1.6 below) is the preferred interactive editing task and editing with LISTR plus UVFLG (§ O.1.5 below) is actually a reasonable option.

O.1.5 Editing with LISTR and UVFLG

Data may be flagged using task UVFLG based on listings from LISTR. To print out the scalar-averaged raw amplitude data for the calibrators, and their *rms* values, once per scan in a matrix format, the following inputs are suggested:

> TASK 'LISTR' ; INP ↵	to review the inputs needed.
> INDI n; GETN m ↵	to select the data set, $n = 3$ and $m = 1$ above.
> SOURCES ' ' ; CALCODE '*' ↵	to select calibrators.
> TIMER 0 ↵	to select all times.
> ANTENNAS 0 ↵	to list data for all antennas.
> OPTYPE 'MATX' ↵	to select matrix listing format.
> DOCRT FALSE ↵	to route the output to printer, not terminal.
> DPARM 3 , 1 , 0 ↵	amplitude and *rms*, scalar scan averaging.
> BIF 1; EIF 0 ↵	to select all IFs, LISTR will list IFs separately.
> FREQID 1 ↵	to select FQ number 1 (note that FQ numbers must also be done separately).
> INP ↵	to review the inputs.
> GO ↵	to run the program when inputs set correctly.

For unresolved calibrators, the VLA on-line gain settings normally produce roughly the same values in all rows and columns within each matrix. At L, C, X, and U bands, these values should be approximately 0.1 of the expected source flux densities. At P band, the factor is about 0.01. The factors for other bands are unspecified. Any rows or columns with consistently high or low values in either the amplitude or the *rms* matrices should be noted, as they probably indicate flaky antennas. In particular, you should look for

- In the amp-scalar averages, look for *dead* antennas, which are easily visible as rows or columns with small numbers. Rows or columns that differ by factors of two or so from the others are generally fine. Such deviations mean only that the on-line gains were not set entirely correctly.
- In the *rms* listings, look for discrepant high values. Almost all problems are antenna based and will be seen as a row or column. Factors of 2 too high are normally okay, while factors of 5 high are almost certainly indicative of serious trouble.

The next step is to locate the bad data more precisely. Suppose that you have found a bad row for antenna 3 in right circular polarization in IF 2 between times (*d1*, *h1*, *m1*, *s1*) and (*d2*, *h2*, *m2*, *s2*). You might then rerun LISTR with the following new inputs:

> SOURCES '' C_R	to select all sources.
> TIMER *d1 h1 m1 s1 d2 h2 m2 s2* C_R	to select by time range.
> ANTENNAS 1 , 2 , 3 C_R	to list data for antenna 3 with two "control" antennas.
> BASEL 1 , 2 , 3 C_R	to list all baselines with these three antennas.
> OPTYPE 'LIST' C_R	to select column listing format.
> DOCRT 1 C_R	to route the output to terminal at its width.
> DPARM = 0 C_R	amplitude only, no averaging.
> STOKES 'RR' C_R	to select right circular.
> BIF 2 C_R	to specify the "BD" IFs.
> FLAGVER 1 C_R	to choose flag table 1.
> GO C_R	to run the program.

This produces a column listing on your terminal of the amplitude for baselines 1–2, 1–3 and 2–3 at every time stamp between the specified start and stop times. The '1–2" column provides a control for comparison with the two columns containing the suspicious antenna.

Note that "amp-scalar" averaging ignores phase entirely and is therefore not useful on weak sources, nor can it find jumps or other problems with the phases. To examine the data in a phase-sensitive way, repeat the above process, but set DPARM(2) = 0 rather than 1. Bad phases will show up as reduced amplitudes and increased *rms*'s.

Once bad data have been identified, they can be expunged using UVFLG. For example, if antenna 3 RR was bad for the full interval shown above, it could be deleted with

> TASK 'UVFLG' ; INP C_R	to select the editor and check its inputs.
> TIMER *d1 h1 m1 s1 d2 h2 m2 s2* C_R	to select by time range.
> BIF 2 ; EIF = BIF C_R	to specify the "BD" IFs.
> BCHAN 0 ; ECHAN 0 C_R	to flag all channels.
> FREQID 1 C_R	to flag only the present FQ number.
> ANTEN 3 , 0 C_R	to select antenna 3.
> BASEL 0 C_R	to select all baselines to antenna 3.
> STOKES 'RR' C_R	to select only the RR Stokes (LL was found to be okay in this example).
> REASON = 'BAD RMS WHOLE SCAN' C_R	to set a reason.
> OUTFGVER 1 C_R	to select the first (only) flag table.
> INP C_R	be careful with the inputs here!
> GO C_R	to run the task when ready.

Continue the process until you have looked at all parts of the data set that seemed anomalous in the first matrix listing, then rerun that listing to be sure that the flagging has cleaned up the data set sufficiently. If

there are lots of bad data, you may find that you have missed a few on the first pass. If you change your mind about a flagging entry, you can use UVFLG with OPCODE = 'UFLG' to remove entries from the flag table. All adverbs of UVFLG are used when removing entries, so you may use REASON along with the channel, IF, source, et al. adverbs to select the entries to be removed. OPCODEs 'REAS' and 'WILD' may be used to undo an entry solely based on the REASON. If the table becomes hopelessly messed up, use EXTDEST to delete the flag table and start over or use a higher numbered flag table. The contents of the flag table may be examined at any time with the general task PRTAB and entries in it may also be removed with TABED and/or TAFLG. Two flag tables can be merged using TAPPE.

O.1.6 Editing with TVFLG

If your data are seriously corrupted, contain numerous baselines, and you like video games, TVFLG is the visibility editor of choice. The following discussion assumes that you have read § 2.3.2 and are familiar with using the *AIPS* TV display. The latest versions of TVFLG and SPFLG are described in detail in an *AIPS* Memo[2] which should be consulted for current details. The following inputs are suggested:

> TASK 'TVFLG' ; INP ↵	to review the inputs needed.
> INDI n ; GETN m ↵	to select the data set, $n = 3$ and $m = 1$ above.
> SOURCES ' ' ↵	to select all sources.
> TIMER 0 ↵	to select all times.
> STOKES 'FULL' ↵	to select all four or 'HALF' to select the two parallel-hand polarizations; you can then toggle between them interactively.
> FREQID 3 ↵	Select FQ entry 3.
> BIF 1 ; EIF 2 ↵	to specify both VLA IFs; you can then toggle between the two interactively.
> ANTENNAS 0 ↵	to display data for all antennas.
> BASELINE 0 ↵	to display data for all baselines.
> DOCALIB 1 ↵	to apply initial calibration to the data.
> FLAGVER 1 ↵	to use flag (FG) table 1.
> OUTFGVER 0 ↵	to create a new flag table with the flags from FG table 1 plus the new flags.
> DPARM = 0 ↵	to use default initial displays and normal baseline ordering.
> DPARM(6) = 30 ↵	to declare that the input data are 30-second averages, or to have the data averaged to 30 seconds. Note that one can interactively increase the time averaging, in integer units of DPARM(6), after the master grid is created.
> DPARM(5) = 10 ↵	to expand the flagging time ranges by 10 seconds in each direction. The times in the master grid are average times and may not encompass the times of the samples entering the average without this expansion.
> DOCAT 1 ↵	to save the master grid file.
> INP ↵	to review the inputs.
> GO ↵	to run the program when inputs set correctly.

If you make multiple runs of TVFLG, it is important to make sure that the flagging table entries are all in one version of the FG table. The easiest way to ensure this is to set FLAGVER and OUTFGVER to 0 and keep it that way for all runs of TVFLG. If you make a mistake, two flag tables may be merged with the task TAPPE.

TVFLG begins by constructing a "master grid" file of all included data. This can be a long process if you

[2] Greisen, E. W. 2025, "Interactive flagging in *AIPS*", *AIPS* Memo 127, https://www.aips.nrao.edu/aipsdoc.html

include lots of data at once, even though a faster, large-memory method of gridding is now usually used. It is probably better to use the channel selection (including averaging channels with NCHAV), IF selection, source selection, and time range selection adverbs to build rather smaller master grid files and then to run TVFLG multiple times. It will work with all data included, allowing you to select interactively which data to edit at any one moment and allowing you to resume the editing as often as you like. But certain operations (such as undoing flags) have to read and process the entire grid, and will be slow if that grid is large. The master grid file is always cataloged (on IN2DISK with class TVFLGR), but is saved at the end of your session only if you set DOCAT = 1 (actually > 0) before starting the task. To resume TVFLG with a pre-existing master grid file, set the adverb IN2SEQ (and IN2DISK) to point at it. When resuming in this way, TVFLG ignores all of its data selection adverbs since they might result in a different master grid than the one it is going to use. If you wish to change any of the data selection parameters, *e.g.*, channels, IFs, sources, times, or time averaging, then you must use a new master grid.

Kept with the master grid file is a special file of TVFLG flagging commands. This file is updated as soon as you enter a new flagging command, making the master grid and your long editing time virtually proof from power failures and other abrupt program terminations. These flagging commands are not entered into your actual *uv* data set's flagging (FG) table until you exit from TVFLG and tell it to do so. During editing, TVFLG does not delete data from its master grid; it just marks the flagged data so that they will not be displayed. This allows you to undo editing as needed during your TVFLG session(s). When the flags are transferred to the main *uv* data set, however, the flagged data in the master grid are fully deleted since undoing the flags at that point has no further meaning. When you are done with a master grid file, be sure to delete it (with ZAP) since it is likely to occupy a significant amount of disk.

TVFLG keeps track of the source name associated with each row of data. When averaging to build the master grid and to build the displayed grids, TVFLG will not average data from different sources and will inform you that it has omitted data if it has had to do so for this reason. For multi-source files, the source name is displayed during the CURVALUE-like sections. However, the flagging table is prepared to flag *all* sources for the specified antennas, times, *etc.* or just the displayed source. If you are flagging two calibrator scans, you may wish to do all sources in between as well. Use the SWITCH SOURCE FLAG interactive option to make your selection before you create flagging commands. Similarly, you will need to decide whether flagging commands that you are about to prepare apply only to the displayed channel and/or IF, or to all possible channels and/or IFs. In particular, spectral-line observers often use TVFLG on the pseudo-continuum "channel-0" data set, but want the resulting flags to apply to all spectral channels when copied to the spectral-line data set. They should be careful to select all channels before generating any flagging commands. Each flagging command generated is applied to a list of Stokes parameters, which *does not have to include* the Stokes currently being displayed. When you begin TVFLG and whenever you switch displayed Stokes, you should use the ENTER STOKES FLAG option to select which Stokes are to be flagged by subsequent flagging commands.

If you get some of this wrong, you can use the UNDO FLAGS option in TVFLG if the flags have not yet been applied to the *uv* data set. Or you can use tasks UVFLG, TABED or TAFLG to correct errors written into the FG table of your multi-source *uv* data set. Flag tables are now used with both single- and multi-source data sets.

TVFLG displays the data, for a single IF, channel (average), and Stokes, as a grey-scale display with time increasing up the screen and baseline number increasing to the right. Thus baselines for the VLA run from left to right as 1–1, 1–2, 1–3, ..., 2–2, 2–3, ..., 27–27, 27–28, and 28–28. An input parameter (DPARM(3) = 1 allows you to create a master grid and display baselines both as, say 1–2 and 2–1. An interactive (switchable) option allows you to order the baselines from shortest to longest (ignoring projection effects) along the horizontal axis.

The interactive session is driven by a menu which is displayed on a graphics overlay of the TV display. An example of this full display is shown on the next page. Move the cursor to the desired operation (noting that the currently selected one is highlighted in a different color on many TVs) and press button A, B, or C

to select the desired operation; pressing button D produces on-line help for the selected operation. The first (left-most column) of choices is:

OFFZOOM	turn off any zoom magnification
OFFTRANS	turn off any black & white enhancement
OFFCOLOR	turn off any pseudo-coloring
TVZOOM	interactive zoom as in AIPS
TVTRANSF	black & white enhancement as in AIPS
TVPSEUDO	many pseudo-colorings as in AIPS
TVPHLAME	many flame-like pseudo-colorings as in AIPS
OFMCOLOR	30 pseudo-colorings as in AIPS
DO WEDGE ?	switches choice of displaying a step wedge
LOAD xxxx	switch TV load transfer function to xxxx
LIST FLAGS	list selected range of flag commands
UNDO FLAGS	remove flags by number from the FC table master grid
REDO FLAGS	re-apply all remaining flags to master grid
SET REASON	set reason to be attached to flagging commands
DO LABEL ?	turn axis labeling on and off
CHAR MULT	to control TV character size as in AIPS

Note: when a flag is undone, all cells in the master grid which were first flagged by that command are restored to use. Flag commands done after the one that was undone may also, however, have applied to some of those cells. To check this and correct any improperly un-flagged pixels, use the REDO FLAGS option. This option even re-does CLIP operations! After an UNDO or REDO FLAGS operation, the TV is automatically re-loaded if needed. Note that the UNDO operation is one that reads and writes the full master grid.

The load to the TV for all non-phase displays may be done with all standard transfer functions: LINear, LOG, SQRT, LOG2 (more extreme log). The menu shows the next one in the list (xxxx above) through which you may cycle. The TV is reloaded immediately when a new transfer function is selected.

Column 2 offers type-in controls of the TV display and controls of which data are to be flagged. In general, the master grid will be too large to display on the TV screen in its entirety. The program begins by loading every $n^{\rm th}$ baseline and time smoothing by m time intervals in order to fit the full image on the screen. However, you may select a sub-window in order to see the data in more detail. You may also control the range of intensities displayed (like the adverb PIXRANGE in TVLOD inside AIPS). The averaging time to smooth the data for the TV display may be chosen, as may the averaging time for the "scan average" used in some of the displays. Which correlators are to be flagged by the next flagging command may be typed in. All of the standard Stokes values, plus any 4-bit mask may be entered. The spectral channel and IF may be typed in. Flagging may be done only for the current channel and IF and source, or it may be done for all channels and/or IFs and/or sources. Note that these controls affect the next LOADs to the TV or the flagging commands prepared after the parameter is changed. When the menu of options is displayed at the top of the TV, the current selections are shown along the bottom. If some will change on the next load, they are shown with a trailing asterisk. Column 2 contains

ENTER BLC	Type in a bottom left corner pixel number on the terminal
ENTER TRC	Type in a top right corner pixel number on the terminal
ENTER AMP PIXRANGE	Type in the intensity range to be used for loading amplitude images to the TV
ENTER PHS PIXRANGE	Type in the phase range to be used for loading phase images to the TV
ENTER RMS PIXRANGE	Type in the intensity range to be used for loading images of the rms to the TV

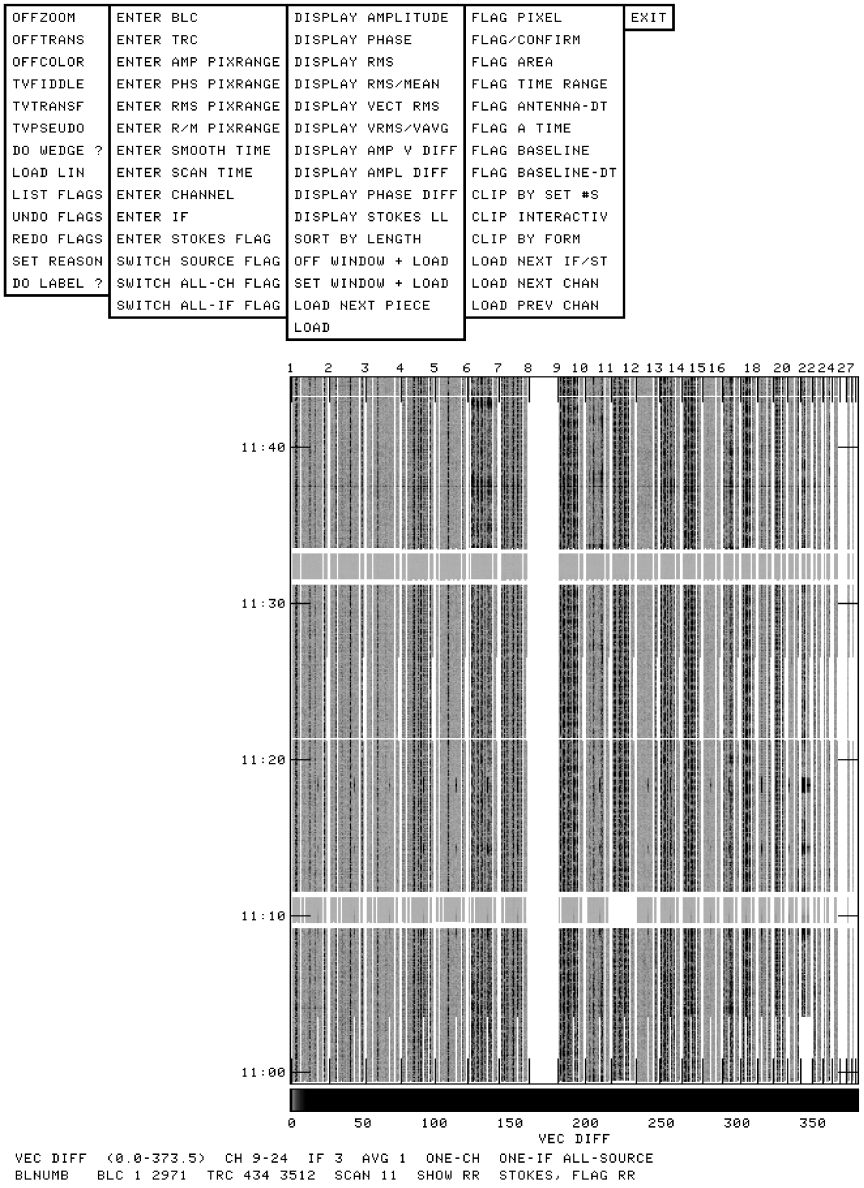


Figure O.1: A display of a sample TV screen from TVFLG, made using the *AIPS* task TVCPS to produce a negative black-and-white display. The TVFLG menu (in the boxes) is displayed in a graphics plane which is normally colored light green. The status line(s) at the bottom and optional axis labels are displayed in a graphics plane that is normally cyan in color. The data are grey scales in a TV memory and may be enhanced in black-and-white or pseudo-colored. The particular display chosen is the amplitude of the vector difference between the sample and a running vector average of samples surrounding it. This particular parameter is sensitive to both phase and amplitude problems and may save you the extra time of looking at phase and amplitude separately. It requires that there be data to average, but does not blur the flagging by the averaging interval (as the RMS method does). The visibility data are from the JVLA. All baselines are shown once only in baseline number order. Antenna 8 and 27 are missing for all times, while antennas 11 and 22 are missing for some times in the piece displayed here. The displayed data are the RR Stokes samples and have been windowed to exclude some times. Flag commands generated at the moment illustrated will flag all source names, one spectral channel (actually channels 9-24 averaged here), one IF, and Stokes RR only (from a 2-Stokes data set). The step wedge and labeling options have been selected. This figure was produced with an older version of TVFLG; consult *AIPS* Memo 127 for the latest details.

ENTER R/M PIXRANGE	Type in the value range to be used for loading rms/mean images to the TV
ENTER SMOOTH TIME	Type in the time smoothing (averaging) length in units of the master grid cell size
ENTER SCAN TIME	Type in the time averaging length for the "scan average" in units of the master grid cell size
ENTER CHANNEL	Type in the desired spectral channel number using the terminal
ENTER IF	Type in, on the terminal, the desired IF number
ENTER STOKES FLAG	To type in the 4-character string which will control which correlators (polarizations) are flagged. Note: this will apply only to subsequent flagging commands. It should be changed whenever a different Stokes is displayed.
SWITCH SOURCE FLAG	To switch between having all sources flagged by the current flag commands and having only those sources included in this execution of TVFLG flagged. The former is desirable when a time range encompasses all of 2 calibrator scans.
SWITCH ALL-CH FLAG	To reverse the flag all channel status; applies to subsequent flag commands
SWITCH ALL-IF FLAG	To cycle the flag all IFs status; applies to subsequent flag commands
AUTO Y ZOOM ?	To allow interpolative zoom in the time axis

The all-channel flag remains true if the input data set has only one channel and the all-IF flag remains true if the input data set has no more than one IF. If there are more than 2 IFs, the SWITCH ALL-IF FLAG cycles between flagging one IF, flagging a range of IFs, and flagging all IFs. When going to the range of IFs, it will ask you to enter the desired range.

An extra word should be said about the "scan average" to which reference was made above. This is used solely for displaying the difference of the data at time T and the average of the data at times near T. This average is computed with a "rolling buffer." Thus, for a scan average time of 30 seconds and data at 10-second intervals, the average for a set of 7 points is as follows:

time	average of	times	
00	00	10	20
10	00	10	20
20	10	20	30
30	20	30	40
40	30	40	50
50	40	50	60
60	40	50	60

The third column of options is used to control which data are displayed and to cause the TV display to be updated. The master grid must be converted from complex to amplitude, phase, the rms of the amplitude, or the rms divided by the mean of the amplitude for display. It may also be converted to the amplitude of the vector difference between the current observation and the "scan average" as defined above or the absolute value of the difference in amplitude with the scalar-average amplitude or the absolute value of the difference in phase with the vector scan average. Furthermore, the baselines may be reordered in the TV display by their length rather than their numerical position. This column has the options:

DISPLAY AMPLITUDE	To display amplitudes on the TV
DISPLAY PHASE	To display phases on the TV
DISPLAY RMS	To display amplitude rms on the TV
DISPLAY RMS/MEAN	To display amplitude rms/mean on the TV

DISPLAY VECT RMS	To display vector amplitude rms on the TV
DISPLAY VRMS/VAVG	To display vector amplitude rms/mean on the TV
DISPLAY AMP V DIFF	To display the amplitude of the difference between the data and a running (vector) "scan average"
DISPLAY AMPL DIFF	To display the abs(difference) of the amplitude of the data and a running scalar average of the amplitudes in the "scan"
DISPLAY PHASE DIFF	To display the abs(difference) of the phase of the data and the phase of a running (vector) "scan average"
DISPLAY STOKES *xx*	To switch to Stokes type *xx* (where *xx* can be RR, LL, RL, LR, etc. as chosen by the STOKES adverb).
SORT BY *xxxxxxxx*	To switch to a display with the *x* axis (baseline) sorted by ordered by LENGTH or by BASELINE number
OFF WINDOW + LOAD	Reset the window to the full image and reload the TV
SET WINDOW + LOAD	Interactive window setting (like TVWINDOW) followed by reloading the TV
LOAD LAST PIECE	Reload the TV with the previous piece of the total time range.
LOAD NEXT PIECE	Reload the TV with the next piece of the total time range.
LOAD	Reload TV with the current parameters

SET WINDOW + LOAD is "smarter" than TVWINDOW and will not let you set a window larger than the basic image. Therefore, if you wish to include all pixels on some axis, move the TV cursor outside the image in that direction. The selected window will be shown. When there are more times than will fit on the TV screen at the current smoothing (averaging) time, the task divides the data up into overlapping time-range "pieces." When it has done so, the LOAD LAST PIECE and LOAD NEXT PIECE menu items will appear. This lets you view one piece after the next, rotating through all pieces, to edit each time interval at full resolution. Note that a FLAG BASELINE will flag that baseline through all pieces.

The fourth column is used to select the type of flagging to be done. During flagging, a TV graphics plane is used to display the current pixel much like CURVALUE in AIPS. Buttons A and B do the flagging (except A switches corners for the area and time-range modes). Button C also does the flagging, but the program then returns to the main menu rather than prompting for more flagging selections. Button D exits back to the menu without doing any additional flagging. Another graphics plane is used to show the current area/time/baseline being flagged. All flagging commands can create zero, one, two, or more entries in the flagging list; hit button D at any time. There are also two clipping modes, an interactive one and one in which the user enters the clip limits from the terminal. In both, the current image computed for the TV (with user-set windows and data type, but not any other windows or alternate pixels etc. required to fit the image on the TV) is examined for pixels which fall outside the allowed intensity range. Flagging commands are prepared and the master file blanked for all such pixels. In the interactive mode, buttons A and B switch between setting the lower and upper clip limits, button C causes the clipping to occur followed by a return to the main menu, and button D exits to the menu with no flagging. The options are

FLAG PIXEL	To flag single pixels
FLAG/CONFIRM	To flag single pixels, but request a yes or no on the terminal before proceeding
FLAG AREA	To flag a rectangular area in baseline-time
FLAG TIME RANGE	To flag all baselines for a range of times
FLAG ANTENNA-DT	To flag all baselines to a specific antenna for a range of times
FLAG A TIME	To flag all baselines for a specific time
FLAG BASELINE	To flag all times for a specific baseline
FLAG BASELINE-DT	To flag a time range for a specific baseline
CLIP BY SET #S	To enter from the terminal a clipping range for the current mode and then clip high and low samples

CLIP INTERACTIV	To enter with the cursor and LUTs a clipping range for the current mode and then clip data outside the range.
CLIP BY FORM	To clip selected channels/IFs using the "method" and clipping range of some previous clip operation
LOAD NEXT IF/ST	Load TV with the next IF or polarization.
LOAD NEXT CHAN	To load the next spectral channel to the TV with current parameters
LOAD PREV CHAN	To load the previous spectral channel to the TV with current parameters

The CLIP BY FORM operation allows you to apply a clipping method already used on one channel/IF to other channels and/or IFs. It asks for a command number (use LIST FLAGS to find it) and applies its display type (amp, phase, rms, rms/mean, differences), averaging and scan intervals and clip levels to a range of channels, IFs and Stokes (as entered from the terminal). To terminate the operation, doing nothing, enter a letter instead of one of the requested channel or IF numbers. To omit a Stokes, reply, if requested for a flag pattern, with a blank line. You may watch the operation being carried out on the TV as it proceeds.

The right-most column has only the option:

EXIT	Resume AIPS and, optionally, enter the flags in the data

Before the flags are entered in the data, TVFLG asks you whether or not you actually wish to do this. You must respond yes or no. Note that, if the master grid is to remain cataloged, there is no need to enter the flagging commands every time you decide to exit the program for a while. In fact, if you do not enter the commands, you can still undo them later, giving you a reason not to enter them in the main *uv* data set too hastily.

The two most useful data modes for editing are probably amplitude and amplitude of the vector difference. The former is useful for spotting bad data over longer time intervals, such as whole scans. The latter is excellent for detecting short excursions from the norm. For editing uncalibrated data, rms of two time intervals is useful, but the rms modes require data to be averaged (inside TVFLG) and therefore reduce the time resolution accuracy of the flagging. If you edit by phase, consider using the pseudo-coloration scheme that is circular in color (option TVPSEUDO followed by button B) since your phases are also circular.

Using TVFLG on a workstation requires you to plan the real estate of your screen. We suggest that you place your message server window and your input window side-by-side at the bottom of the screen. Then put the TV window above them, occupying the upper 70–90% of the screen area. (Use your window manager's tools to move and stretch the TV window to fill this area.) Instructions and informative, warning and error messages will appear in the message server window. Prompts for data entry (and your data entry) appear in the input window. Remember to move the workstation cursor into the input window to enter data (such as IF, channel, antenna numbers, and the like) and then to move the cursor back into the TV area to select options, mark regions to be flagged, adjust enhancements, and so on.

O.1.7 Spectral-line calibration

The calibration of spectral-line data is very similar to that of continuum data with the exception that the antenna gains have to be determined and corrected as a function of frequency as well as time. The model used by *AIPS* is to determine the antenna gains as a function of time using a pseudo-continuum ("channel-0") form of the data. Then the complex spectral response function ("bandpass") is determined from observations of one or more strong continuum sources at or near the same frequency as the line observation. In general, the channel-0 data are calibrated using the recipes in the previous sections of this chapter. The sub-sections below are designed to bring out the few areas in which spectral-line calibration differs from continuum.

O.1.7.1 Spectral-line aspects of SETJY

The LISTR output with OPTYPE = 'SCAN' will show information from the source table including spectral-line parameters. VLA data from FILLM are normally supplied with adequate information regarding the source velocity, line rest frequency, and velocity reference (radio versus optical, LSR versus barycentric). However, data from the EVLA and other telescopes may be missing these parameters. SETJY must then be used to fill in the needed values. An OPTYPE='VCAL' option computes the velocity of the reference channel from first principles. It is recommended over inaccurate guesses of adverb values in other SETJY OPTYPEs. It may be run over all sources after the rest frequencies and velocity reference information has been filled in.

O.1.7.2 Editing the spectral data

You should follow the steps outlined in § 4.3.5 to edit the calibrator data using the channel-0 data set. Even though channel-0 data is continuum, be careful to have TVFLG and UVFLG generate the flagging commands for all channels, not just channel 1. Then, copy the resulting FG table to the line file. Use TACOP:

> TASK 'TACOP' ⏎
> INDI *n* ; GETN *m* ⏎ to specify channel-0 data set.
> OUTDI *i* ; GETO *j* ⏎ to specify the line data set.
> INEXT 'FG' ⏎ to copy the FG table.
> INVER 1 ⏎ to copy table 1.
> NCOUNT 1 ⏎ to copy only one table.
> OUTVER 1 ⏎ to copy it to output table 1
> INP ⏎ to review the inputs.
> GO ⏎ to run the program when inputs set correctly.

Specifying the "ALL-CH" setting in TVFLG and specifying BCHAN 1 ; ECHAN 0 ⏎ in UVFLG cause all channels to be flagged when the FG table is copied to the line data set.

Spectral-line observers should also use SPFLG (§ 10.2.2) to examine and, perhaps, to edit their data. This task is very similar to TVFLG described in § O.1.6, but SPFLG displays spectral channels for all IFs on the horizontal axis, one baseline at a time. If you have a large number of baselines, as with the VLA, then you should examine a few of the baselines to check for interference, absorption (or emission) in your calibrator sources, and other frequency-dependent effects. Use the ANTENNAS and BASELINE adverbs to limit the displays to a few short spacings and one or two longer ones as well. If there are serious frequency-dependent effects in your calibrators, use SPFLG and UVFLG to delete them. (You might wish to delete the FG table with EXTDEST to begin all over again.) Then use AVSPC to build a new channel-0 data set and repeat the continuum editing. Note that you should not copy the FG table from the spectral-line data set to the new continuum one. The reason for this is the confusion over the term "channel." If you have flagged channel 1, but not all channels, in the spectral-line data set — a very common occurrence — then a copied FG table would flag all of the continuum data since it has only one "channel." When you have flagged the channel-0 data set, you can merge the new flags back into the spectral-line FG table with task TABED.

> TASK 'TABED' ⏎
> INDI *n* ; GETN *m* ⏎ to specify channel-0 data set.
> OUTDI *i* ; GETO *j* ⏎ to specify the line data set.
> INEXT 'FG' ⏎ to copy the FG table.
> INVER 1 ⏎ to copy table 1.
> OUTVER 1 ⏎ to copy it to output table 1.
> BCOUNT 1 ; ECOUNT 0 ⏎ to copy from the beginning to the end.

> OPTYPE 'COPY' ↵	to do a simple copy appending the input table to the output table.
> TIMER 0 ↵	to copy all times.
> INP ↵	to review the inputs.
> GO ↵	to run the program when inputs set correctly.

If the channel-0 data set is meaningful for your program sources, you might consider doing a first-pass editing of them along with your calibrators before copying the FG table back to the line data set. If your program sources contain significant continuum emission, then this is a reasonable operation to perform. If they do not, then the standard channel-0 data set is not useful for editing program sources. You can use SPFLG to edit all channels, or if the signal is strong in a few channels, you could run TVFLG on those channels from the spectral-line data set or average those with the BCHAN, ECHAN, and NCHAV adverbs.

O.1.7.3 Calibrating the spectral data

The channel-0 data set should be calibrated as described above for continuum data (§4.3.5 and §4.3.10). When you are satisfied with your results, you should copy the relevant CL table over to the line data set with TACOP:

> TASK 'TACOP' ↵	
> INDI *n* ; GETN *m* ↵	to specify the channel-0 data set.
> OUTDI *i* ; GETO *j* ↵	to specify the line data set.
> INEXT 'CL' ↵	to copy a CL table.
> INVER 0 ↵	to copy highest numbered table from CLCAL step.
> NCOUNT 1 ↵	to copy only one table.
> OUTVER 0 ↵	to create new output table.
> INP ↵	to review the inputs.
> GO ↵	to run the program when inputs set correctly.

If you copy SN, TY, or SY tables, you may apply a flagging table to the table values.

Then one runs BPASS on the bandpass calibration source(s). This is the same process as for the modern VLA; see §4.3.8 for details.

At this point it is often useful to examine your fully calibrated data using POSSM:

> TASK 'POSSM' ↵	
> INDI *i* ; GETN *j* ↵	specify line data.
> SOURCES '*source1*' , ' ' ↵	to specify the source of interest.
> ANTENNAS 0 ↵	to plot all antennas.
> BCHAN 10 ; ECHAN 55 ↵	to plot spectrum for this channel range only.
> DOCALIB 1 ↵	to apply the antenna gain to both visibilities and weights (if appropriate). calibration.
> GAINUSE 0 ↵	to use most recent CL table.
> DOBAND 3 ↵	to apply the bandpass calibration time smoothed.
> BPVER 1 ↵	to use BP table 1.
> FREQID 1 ↵	to use only one FQ value.
> APARM 0 ↵	to do vector averaging of amplitudes and self-scale the plots.

> SMOOTH 5 , 0 C_R	to apply Hanning smoothing in the spectral domain after bandpass calibration is applied. Use 13,0 to smooth after the data are averaged, which is faster and less prone to oddities due to channel-dependent flagging. Use 1,0 only if the data were Hanning smoothed when BPASS was run.
> INP C_R	to review the inputs.
> GO C_R	to run the program when inputs set correctly.
> GO LWPLA C_R	to send the plot to the (PostScript) printer/plotter.

If you have multiple FQ entries in your data set, you should repeat the calibration for each additional FQ entry. Bookkeeping is simplified if you eliminate all extant SN tables before calibrating the data associated with each frequency identifier. However, it is not essential to do this.

O.1.8 Solar data calibration for the historic VLA

The calibration of solar *uv* data differs from normal continuum and spectral-line calibration in one critical respect: the system temperature correction to the visibility data is applied by the observer in *AIPS*. See Lecture 21 in *Synthesis Imaging in Radio Astronomy* for a discussion of the system temperature correction as it applies to VLA solar visibility data. The system temperature correction is embodied in a quantity referred to as the "nominal sensitivity," an antenna-based numerical factor normally applied in real time to the scaled correlation coefficients before they are written to the VLA archive. With the exception of X and L band, only a handful of VLA antennas are equipped with so-called "solar CALs." The nominal sensitivity is only computed for those antennas so-equipped, namely (for the old VLA) antennas 5, 11, 12, and 18 (at K, U,and C bands) and antennas 7, 12, 21, and 27 (at P band). The system-temperature correction for those antennas without solar CALs must, therefore, be bootstrapped from those antennas which do have solar CALS. This is accomplished through two tasks for the old VLA and two other tasks for the new VLA.. For the old VLA, FILLM fills the uncalibrated visibility data to disk and places the nominal sensitivities in a TY extension table. Then, SOLCL applies the nominal sensitivities to calibration parameters in the CL table. For the new VLA, BDF2AIPS reads the data in, writing a SysPower (SY) and CalDevice (CD) table. Then SYSOL applies the gain and weight corrections. See § 4.1.1 and § 4.3.15.

O.1.8.1 Reading solar data from the VLA archive

To load a solar *uv*-data file to disk from an old VLA archive data set follow the general instructions given above (§ O.1.1 and § O.1.2) with the following additions:

> VLAMODE 'S ' C_R	to indicate solar mode observing.
> CPARM(2) 16 C_R	to indicate that moving sources are allowed without renaming.

If your experiment involved observing active solar phenomena, (*e.g.*, flares), you may wish to update the system-temperature correction every integration time. For example, if you observed a flare with an integration time $\tau = 1.67$ seconds, choose

> CPARM(8) 1.67 / 60 C_R	for 1.67 sec CL and TY table intervals.

Loading an entire solar *uv*-data set to disk with the minimum integration time results in large disk files which make all subsequent programs take a longer time to run. By modern standards, historic VLA data sets are relatively small, so the following may no longer be necessary. A useful strategy is to load the data with relatively low time resolution (20–30 seconds for observations of active solar phenomena) and to proceed with the usual continuum data calibration, deferring the system temperature correction. When a

satisfactory calibration is obtained, the relevant SN table may be saved using TASAV. (Note that you must save the SN table, before running CLCAL rather than the final CL table.) Then run CLCAL and inspect the data for interesting periods of activity — try UVPLT with BPARM = 11, 1 for plots of amplitude versus time or TVFLG, displaying amplitudes as a function of baseline length and time. Use FILLM to load the relevant time ranges of solar *uv* data to disk with no averaging. The saved SN table is then copied to each high-time resolution data set. Assess, and possibly edit, the nominal sensitivities (§ O.1.8.2) and then apply the system-temperature corrections (§ O.1.8.3). Finally, apply the saved/copied SN table to the CL table of each using CLCAL.

O.1.8.2 Using SNPLT and LISTR to assess the nominal sensitivities

When solar *uv* data are written to disk, FILLM writes the nominal sensitivities of those antennas equipped with solar CALs into the TY table. Before bootstrapping the system temperature correction for antennas without solar CALs from those which do, it is always wise to examine the nominal sensitivity for each of the solar CAL antennas for each of the IFs. The tools available for this purpose include: SNPLT, which plots the nominal sensitivities in graphical form, LISTR or PRTAB, which allow one to inspect the values directly, and EDITA, which provides an interactive display of the TY data and allows you to edit the data. To make plots:

> TASK 'SNPLT' ; INP ↵	to review the inputs needed.
> IND *m* ; GETN *n* ↵	to specify the input *uv* file.
> INEXT 'TY' ↵	to plot data from TY extension table.
> INVERS 0 ↵	to use the highest version number.
> SOURCES 'SUN' , ' ' ↵	to plot solar source only.
> TIMERANG 0 ↵	to select all times.
> ANTENNAS 5 11 12 18 ↵	to select only CAL-equipped antennas; this sample list for K, U, or C band.
> PIXRANG 0 ↵	to self-scale each plot.
> NPLOTS 4 ↵	to do 4 plots on a page.
> FACTOR 2 ; SYMBOL 5 ↵	to use triangles to mark the data and enlarge them by a factor of 2. The symbols may even be connected by lines.
> XINC 1 ↵	to plot every XINC$^{\rm th}$ point.
> OPTYPE 'TSYS' ↵	to plot nominal sensitivities.
> INP ↵	to review the inputs.
> GO ↵	to run the program when you're satisfied with inputs.

SNPLT produces a PL extension file which may be plotted using LWPLA, TKPL, or TVPL — or you could set DOTV TRUE in SNPLT and get the display directly (and temporarily) on the TV. Then to inspect the values over some limited time range in detail, run LISTR (assuming the adverbs set above and):

> TASK 'LISTR' ; INP ↵	to review the inputs needed.
> OPTYPE 'GAIN' ↵	to list quantities in a calibration file.
> INEXT 'TY' ↵	to select the sensitivities.
> TIMER *d1 h1 m1 s1 d2 h2 m2 s2* ↵	to select by suspect time range.
> DOCRT -1 ↵	to route output to the printer.
> DPARM 10 0 ↵	to list nominal sensitivities.
> INP ↵	to review the inputs.
> GO ↵	to run the program when you're satisfied with inputs.

Task SNIFS is similar to SNPLT except that it plots IF on the x axis to compare solutions across them. It has numerous binning options to control the otherwise excessive plotting.

The use of EDITA with TY tables is described extensively in § 4.3.11 and need not be described further here.

O.1.8.3 Using SOLCL to apply the system-temperature correction

For the old VLA, once you have identified the appropriate subset of reference solar CAL antennas for each source and IF, you are ready to bootstrap the system-temperature correction of the remaining antennas. It is recommended that you run SOLCL before applying any other calibration to the CL table. In this way, you can easily verify that the appropriate corrections have been made to each antenna. Then you apply the system-temperature correction to version 2 and correct mistakes by deleting and recreating version 2. To run SOLCL:

> TASK 'SOLCL' ; INP C_R	to review the inputs needed.
> SOURCES '*' C_R	to correct all sources.
> STOKES ' ' C_R	to correct both polarizations.
> TIMERANG 0 C_R	to correct all times.
> ANTENNAS 5 11 12 18 C_R	to use the listed antennas as references.
> SUBARRAY 1 C_R	to modify sub-array 1.
> GAINVER 2 C_R	to write corrected entries to CL table version 2.
> INP C_R	to review the inputs.
> GO C_R	to run the program when you're satisfied with inputs.

After applying the system temperature correction, you may proceed with the usual *AIPS* data calibration procedures outlined in previous sections, including the special solar tactics described in § O.1.8.1.

O.2 Old VLBI format data

For data from a MkIII correlator, run MK3TX, MK3IN, MSORT, DBCON, UVAVG, TAMRG, SBCOR, and INDXR as needed (§ O.2.1.1–§ O.2.1.7). However, observations which require multiple passes through the correlator (including MkIII Modes A, B, and C observations) will have one file per observing mode *per correlation pass*. Data from separate correlator passes can be concatenated using task VBGLU and/or merged with task VBMRG.

O.2.1 Loading data from a MkIII/MkIV correlator

O.2.1.1 Running MK3IN

Data from a MkIII correlator, such as that in Bonn, Germany or Haystack, Massachusetts, can also be read into *AIPS*. To do this you need to be supplied with the so called "A" tape output, also known as "type 52's." These data tapes can be read and translated by the task MK3IN. The process of reading MkIII correlator data into *AIPS* and preparing it for further processing is more cumbersome than the equivalent process for VLBA correlator data. This simply reflects the manner in which data are generated on a baseline-based correlator with a limited number of playback drives. MkIII data may also appear in the form of a Unix tar file. For such data, use M3TAR and TFILE rather than MK3IN and AFILE, respectively.

Before running MK3IN, run the task MK3TX to extract the text files from the MkIII archive tape. These text files contain information about the correlated scans in the data set. MK3TX will first provide an index of all the text files and then ask you to select files for loading onto disk. It then asks you interactively for the desired

destination of the text files. It is important to load and concatenate all the "A" files, *i.e.*, those files having names like Atttt. The meaning of the other text files is described in the MK3TX Explain file. Sometimes the text files are not on the tapes, which means that you cannot select sub-sets of the data using the A-files, but is not otherwise catastrophic.

If the A-files are present and have been loaded onto the disk, use AFILE to sort and edit these files to produce a list of scans to be loaded by MK3IN. Use APARM settings in AFILE to establish criteria for selecting between any duplicate scans which may appear on the archive. If the data set contains data at multiple frequencies, you should edit the resulting output text file so that there is a version for each frequency, containing only those scans at that frequency.

The final step before running MK3IN is to create another text file which provides the commands for the task. This step is necessary since some information that is needed by *AIPS* is not present on the tape. Ideally, in this text file (as shown below), the parameter STATIONS should be a list of all the stations correlated, with the exact name used at correlation. If you do not have such a list, you can instead specify a list containing STATIONS 'ANY', 'ANY' ... Note that there must be at least as many 'ANY' entries as there are stations in the data set or some of the stations will not be loaded. The parameters in this text file are:

STOKES='RR','LL'	the Stokes range of the output file. The standard abbreviations are used to select the polarization range. The largest consistent range is used. For example: STOKES='RR','LL' will cause only RR and LL to be written. STOKES='LL' will cause just LL to be written. STOKES='RR','LR' will cause all four circular polarization combinations to be in the output file, since RR and LR span the range of allowed *AIPS* Stokes values.
FREQCODE='R','L','r','l'	the polarization codes used by MkIII correlators are anything but standard and they need to be supplied to MK3IN using the parameter FREQCODE. The one character polarization identifiers are expected in the order RR, LL, RL, and LR. The usual correlator convention is 'R'=RR, 'L'=LL, 'r'=RL, 'l'=LR and this is the default assumed by MK3IN. However, other codes are possible. For example FREQCODE = 'A', 'B', 'C', 'D' will interpret 'A' as RR, 'B' as LL and so forth, while FREQCODE = 'R', 'C', 'r', 'l' will use the default abbreviations except that 'C'=LL. If MK3IN encounters an unidentified polarization code the task will report: AT20XX: Unidentified Stokes parameter: 'X'. In this case, modify the FREQCODE parameter to include this polarization identifier. This will ensure that polarizations are not misidentified inadvertently.
NO_POL=2	the number of polarization correlations (*e.g.*, RR, LL, RL and LR), the default is 1.
STATIONS='NRAO','VLA','OVRO','FDVS','MPI'	station names.
/	keyin style delimiter.

Then, from inside AIPS, mount the tape (§ 3.9) and run MK3IN:

> TASK 'MK3IN' ; INP ↵	to review the inputs.
> INFILE 'MYVLB:PARAM.LIS' ↵	to define the text control file.
> IN2FILE 'MYVLB:AFILE.LIS ↵	to point to a file containing a list of scans to be loaded as produced by AFILE
> INTAPE 4 ↵	to specify the tape drive number.
> NFILES 0 ↵	to skip no files on tape.

> OUTNA 'EXP 86-34' ↵	to select the output file name.
> OUTCL 'MK3IN' ↵	to select the default output class name.
> REFDATE '12/11/89' ↵	to tell MK3IN the start date of the observations — get this right or you may get negative times.
> SOURCES '' ↵	to accept all sources found.
> TIMERANG 0 ↵	to accept data from all times found.
> DOUVCOMP 1 ↵	to write data on disk in compressed format.
> APARM 1, 0 ↵	to set the time increment in the CL table entries in minutes.
> APARM(7) 1 ↵	to separate sidebands into separate *AIPS* IFs; the default is to store both USB and LSB in the same IF.
> GO ↵	to run the program.

If the data are contained on more than one Exabyte or DAT tape, load the second tape and re-run MK3IN, setting DOCONCAT = 1 ↵ so that the data are appended to the previous output file. Before running MK3IN a second time, it is important to set the list of STATIONS in the control file to exactly those found when loading the first tape; use PRTAN on the output file to obtain this list. Also leave additional 'ANY' entries after the list for any stations that are on the second tape but which were not on the first tape. The use of DOUVCOMP = 1 is recommended for most data sets, see Appendix F.

O.2.1.2 Sorting MkIII/IV data

The *AIPS* data files created by MK3IN will be in an arbitrary sort order. Use UVSRT or MSORT to sort them into time-baseline order:

> TASK 'UVSRT' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to select the input file.
> OUTNA INNA ; OUTCL 'TBSRT' ↵	to specify the output file.
> SORT 'TB' ↵	to sort to time-baseline order.
> GO ↵	to make the sorted *uv* file.

O.2.1.3 Concatenating MkIII/IV data

If you did not set DOCONCAT=1 when running MK3IN and as a result several files were loaded from tape for one observation, use DBCON to concatenate them together. In order to have the concatenated data all appear in a single subarray, both input files for DBCON must have the same reference day number and identical antenna numbers. That is, the antennas extension (AN) files with each input *uv* data file must be the same. MATCH may be used to repair discrepancies.

You may list the contents of AN files using PRTAN. To run DBCON:

> TASK 'DBCON' ; INP ↵	to review the inputs.
> INDISK *n1* ; GETN *ctn1* ↵	to select the 1st input file.
> IN2DISK *n2* ; GET2N *ctn2* ↵	to select the 2nd input file.
> OUTNA INNA ; OUTCL 'DBCON' ↵	to specify the output file.
> DOARRAY 1 ↵	to force DBCON to mark the output data records as being in the same sub-array. For this to work properly, both of the input files must have the same reference day and have identical antennas files.
> GO ↵	to concatenate the two files.

Task DBAPP may be used to avoid the 2^n proliferation of files, but only if the files are fairly similar in antennas, subarrays, and frequency IDs.

O.2.1.4 Labeling correlator polarization data

Most MkIII and MkIV VLBI setups reverse the polarizations and assign odd-numbered bands to LCP and even-numbered bands to RCP. In this case BANDPOL should be set to '*(LR) ' and the output data set will again be of equal size to the input data with two polarizations and half the number of IFs. This case normally applies if LISTR shows pairs of IFs with the same frequency and QHEADER shows one pixel on the STOKES axis with coordinate value LL, but there may be exceptions to this rule when non-VLBA antennas are used.

To use FXPOL directly, typical inputs are:

> TASK 'FXPOL' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> BANDPOL '*(LR)' ↵	to specify the normal MkIII and MkIV polarization structure.
> GO ↵	to run the program.

Consult HELP FXPOL for further information about more complicated cases. Note that FXPOL has to write a new output file since the structure of the data is being changed. All standard extension files are also converted, but it is still a good idea to run FXPOL before running the calibration tasks.

O.2.1.5 Merging MkIII/IV data

MkIII VLBI correlators usually produce redundantly correlated data. You must merge the data using UVAVG:

> TASK 'UVAVG' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> OUTNA INNA ; OUTCL 'UVMRG' ↵	to specify the output file.
> YINC 4.0 ↵	to set the averaging interval of the input data records (in seconds).
> OPCODE 'MERG' ↵	to direct the task to perform the merge operation.
> GO ↵	to run the program.

The CL table should only contain one entry for each antenna at each time stamp. But, due to the merging process described above and the fact that redundant correlations may have been performed, there is one step to follow before you have consolidated your database fully. You must run TAMRG to remove the redundant CL entries:

> TASK 'TAMRG' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> INEXT 'CL' ↵	to specify the table type to merge.
> INVER 1; OUTVER INVER ↵	to process the input table in place.
> APARM 4, 1, 4, 0, 1, 1, 1, 0 ↵	to control the merging: don't ask why, just do it!
> BPARM 1, 4 ↵	to set compared columns — again, don't ask.
> CPARM 1.157e-5, 0.2 ↵	to set degree of equality — ditto.
> GO ↵	to run the program.

O.2.1.6 Correcting MkIII/IV sideband phase offsets

If your observation contains a mixture of VLBA and non-VLBA antennas and you have not stored the sidebands as separate IFs, there will be a phase offset of about 130° between the upper and lower sidebands on baselines from VLBA to non-VLBA antennas. A correction for this offset is achieved using the task SBCOR:

> TASK 'SBCOR' ; INP ↵	to review the inputs.
> INDISK *n1* ; GETN *ctn* ↵	to specify the input file.
> OUTNA INNA ; OUTCL 'SBCOR' ↵	to specify the output file.
> BCHAN 1 ↵	to specify the lowest channel of lower sideband.
> ECHAN 4 ↵	to specify the highest channel of lower sideband.
> APARM(1) 0 ↵	to apply the default phase offset (*i.e.*, $-130^{\rm deg}$.)
> ANTENNAS = VLBA ; INP ↵	to specify the VLBA antenna numbers; the = sign is required here. The verb VLBA reads the antenna file to find VLBA antennas.
> GO ↵	to run the program.

If you have loaded the VLBAUTIL procedures, then you may use a procedure called ANTNUM to translate a station name into a station number. Thus ANTENNAS = ANTNUM('BR'), ANTNUM('FD'), ... The verb VLBA in is easier.

O.2.1.7 Indexing MkIII/IV data

Next, you must index your data. The NX table is useful as a summary of the file for you, and is also used by the calibration programs to provide quick access for reading data. Create this file with INDXR:

> TASK 'INDXR' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> CPARM 0, 30, -1 ↵	to allow ≤ 10-minute time gaps within scans, to limit scans to ≤ 30 minutes, and to not create a new CL table.
> GO ↵	to run the program.

Files with suffixes .SCAN and .MKIII contain scan summaries and MkIII information and are for information purposes only.

Other than these initial loading and merging steps, the reduction of MkIII and MkIV correlator data is identical to that of VLBA correlator data.

For MkIII data from the Bonn correlator, phase-cal measurements are incorporated directly into the first CL table produced by MK3IN — this is another strong reason to protect the first CL table.

L Special Considerations for EVLA P-band Data Processing in *AIPS*

The EVLA has been equipped with new, wide-band receivers in the range 230 MHz to 470 MHz, known as P-band. Most of the data reduction and imaging at this band are similar to those at other bands, so the following will assume some degree of familiarity with the rest of the *CookBook*, particularly Chapter 4. This guide was initially prepared by Minnie Mao from an earlier guide written by Susan Neff. Enough has changed with the P-band system that a thorough revision of the Appendix has been done.

Note that the choice of file names, disk assignments, catalog numbers, and the like depends on your circumstances and will almost certainly not be the same as those shown in the examples below. Note also that experienced (but still reasonable) EVLA users disagree on the details of data reduction. Some compromise in these details has been attempted below.

RFI, especially in spectral-line observations, has tended to accumulate at the phase center for reasons not well understood. It may be best to observe your sources somewhat offset from the phase center.

L.1 P-band calibration and editing in *AIPS*

1. Download your data set from the archive found at
   `https://data.nrao.edu/`
   making sure that you select SDM-BDF data set (all files) as the download data format.

2. Before starting `AIPS` and while in the terminal window you intend to use for `AIPS`, enter
   > cd *my_data_directory* CR — to change to the data directory containing the top-level directory of your data set.
   > export MYAREA=‘pwd‘ CR — for bash shells, or
   > setenv MYAREA ‘pwd‘ CR — for tcsh and c shells.

   Note the back ticks around `pwd` which cause the output of `pwd` to appear in the typed line.

3. Start `AIPS` with, for example,
   > aips tv=local pr=3 CR — to use Unix sockets for the TV and printer 3

   and enter your user number when prompted.

4. Look at the SDM-BDF file contents with `BDFLIST` — make sure you do *not* have a quote at the end of the file name, or else the lower case characters get changed into upper case and the file won't be found.
   > default bdflist CR — initialize all adverbs
   > dowait 1; docrt 1 CR — to get output on terminal as well as log file.
   > asdmfile(1) = 'MYAREA: CR — two parts are concatenated.
   > asdmfile(2) = 'TSUB0001.sb23643640.eb23666175.56454.14034020834 CR
   > bdflist CR — to examine file contents.

   Pay particular attention to the configuration numbers and number of spectral channels. These refer to each unique correlator *and* antenna setup, not just the size of the current VLA. Each configuration and number of channels that you want will need to be loaded individually.

5. Load data into *AIPS* with BDF2AIPS — We put everything on disk d because that's where there's more room. Note, do not use DEFAULT if you wish to retain the adverbs from the previous step. Loading data may take a while depending on the size of your data.

> outn 'my-name' ↵	to select a meaningful file name.
> config c ↵	to select configuration c.
> outdi d ↵	to select the disk.
> bdf2aips ↵	to translate the data into *AIPS*.
> nchan N ↵	to limit to data with N spectral channels; some spectral-line configurations can contain more than one number of channels but AIPS can only load one number at a time.

Do this as often as needed to load all desired configurations. Spectral-line set-ups can have spectral windows with different numbers of channels within the same configuration. *AIPS* can only handle one at a time, so specify nchan N ↵ to load the data with N channels and repeat as needed.

6. Look at header of the *AIPS* file written by BDF2AIPS

> indi d; getn 1 ↵	to select disk and catalog number, assuming you had no files on disk d previously.
> imh ↵	to see the details of the uv-data set header.

7. look at where the telescopes were during your observation with PRTAN

> default prtan ↵	
> indi d; getn 1 ↵	
> docrt = -1 ↵	to make a printer listing to keep at your terminal.
> go prtan ↵	

8. Look at the observation summary with LISTR

> default listr ↵	
> indi d; getn 1 ↵	
> opty 'scan' ↵	to list scans and sources.
> docrt = -1 ↵	to make a printer listing to keep at your terminal.
> go ↵	

At this stage, Minnie's summary recommends separating P-band from 4-band data. At present, only P-band is recorded so that step is not needed. Then, Minnie recommends using POSSM to find dead antennas plus those with crossed polarization. Then she recommends UVFLG to flag the dead ones and FIXRL to uncross the mis-wired ones. These problems have been essentially eliminated. In addition, P-band data are now delivered by default in a number of relatively narrow spectral windows (called IFs in *AIPS*) which helps minimize data lost to RFI and eliminates the NOIFS and MORIF steps from Minnie's guide.

9. However, the requantizer gains are now allowed to vary and must be corrected.

> default tyapl ↵	to select the needed task
> indi d; getn 1 ↵	to select the input file.
> optype 'PGN' ↵	to select requantizer gain correction.
> inext 'SY' ; in2ver = 1 ↵	to apply the SysPower table, the version number is not defaulted.
> outdis d ; go ↵	to stay on one disk.

10. Narrow-band RFI will normally cause ringing in P-band data. Use POSSM to examine some baselines from a calibration source to check this if you want to be careful. The default setup delivers data from outer IFs for which the sensitivity of the P-band receiver is poor. Use SPLAT to drop the insensitive IFs (those centered on 232, 248, 264, and 472 MHz) and to do Hanning smoothing to reduce the ringing.
    > default splat ↵ to select the needed task
    > indi *d*; getn 2 ↵ to select the (new) input file.
    > flagver 1 ↵ to apply the on-line flags.
    > smooth = 1, 0 ↵ to do Hanning smoothing once and for all.
    > bif = 4; eif = 15 ↵ to select the more sensitive IFs.
    > outdis *d* ; go ↵ to put the output on the same disk.

11. There is also a Navy satellite that makes a mess of data from 360 MHz to 380 MHz. Delete this with UVFLG in one of 2 ways. For the standard setup, you can specify the channels as
    > default uvflg ↵
    > indi *d*; getn 3 ↵
    > bif 6 ; eif 6; bchan 65; echan 128; ↵
    > go ↵
    > bif 7 ; eif 7; bchan 1; echan 97; ↵
    > go ↵

    For any setup, you may make a text file containing the line
    ```
    BFREQ = 360 EFREQ = 380 /
    ```
    and then use it with UVFLG as
    > default uvflg ↵
    > indi *d*; getn 3 ↵
    > intext = *'MYWORK:pband.flg* ↵ to give file name
    > go ↵

12. Edit UV data with EDITA and the SY table. *The* SY *table* $P_{\rm sum}$ *parameter is sensitive to the occasional bursts of RFI which render the data useless. Flag all seriously discrepant points but do not flag really tightly since noise and even abnormalities in this parameter are "normal."*
    > default edita ↵
    > indi *d*; getn 3 ↵
    > inext 'SY' ↵
    > go ↵

13. At this time, it is best to identify those spectral channels affected by RFI throughout the run and to flag them once and for all. FTFLG will display the data combining all included baselines in a single display (per polarization), while the very similar SPFLG allows much better discrimination (and more work) by displaying the baselines separately. Be sure to flag all sources if you flag channels over all times.
    > default ftflg ↵
    > indi *d*; getn 3 ↵
    > calcode '*' ↵ to examine calibrators only
    > dparm(6) = *x* ↵ to specify the basic integration time in the data in seconds.
    > go ↵

    If you use SPFLG, you should limit the times, sources, and/or baselines examined. The flagging then is best done with UVFLG from notes made while you study the data with SPFLG.

> default uvflg C_R
> indi d; getn 3 C_R
> bif = bi ; eif = ei C_R to set range of spectral windows.
> bchan = bc ; echan = ec C_R to set range of spectral channels.
> antennas = $a1, a2, a3$ C_R to select antennas to flag.
> baseline = $b1, b2, b3, b4$ C_R to limit flags to the baselines from the pairs $a1$ through $a3$ with $b1$ through $b4$.
> reason 'bad channels' C_R to set a reason in the flag table.
> go C_R and repeat as needed with different IFs, channels, antennas, etc.

14. This step is optional and should be considered only if you have several P-band data sets to reduce. There is a special capability in *AIPS* developed for lowband data. The task FGTAB was written to read a flag table and write to a text file those flags that apply to all times, sources, antennas, subarrays, and frequency IDs. The channel specifications are written in frequency units which enables UVFLG to read the text file and apply the same flags to the channels to which they pertain in the second data set. Of course, if the two data sets have identical frequency structure, then one could simply copy the FG table from one data set to the other. Task TACOP would be used if you want the flag file by itself. TAPPE would be used if you want to append the flag table of the first file to one of the second. Note that these two tasks transfer all flags in the input table, not just those that flag the general RFI environment.

15. Calibrate for the delay errors with FRING. Choose a time range in the middle of the observation where you know the antennas have settled down. A short time range is all you need — look at LISTR — because you don't want time variations to come into play. This is also why you should choose something bright — like 3C286, which is used here as the name for the generic fringe- and bandpass-fitting calibration source. It would be wise to check these data for RFI with POSSM before running FRING.
> default fring C_R
> indi d; getn 3 C_R
> calso '3C286' ' C_R to select the primary calibration source.
> timerang 0 3 40 0 0 3 40 55 C_R to select < 1 minute of good data from the calibration source.
> solint 1 C_R to average over all included times.
> aparm(5) 1 ; aparm(6) 1 C_R to fit a single delay over all IFs and to print some extra detail in the fitting.
> dparm(9) 1 C_R to avoid any fitting for rates — important.
> refant 22 C_R to set a reference antenna, if one is known to be better than others.
> go C_R

This print level will display the delays found, which should all be a few nano-seconds although larger (correct) delays sometimes arise.

16. Apply these delay solutions to the calibration (CL) table with CLCAL
> default clcal C_R
> indi d; getn 3 C_R
> calso '3C286'," C_R
> snver 1 C_R to specify the SN table version written by FRING.
> go C_R to copy CL table 1 to version 2 adding the delay correction, taken to be constant in time.

17. You should flag out any bad data on your bandpass calibrator. Some users feel that a quick-and-dirty flagging is enough at this stage. Use CLIP with a carefully chosen, perhaps IF-dependent, cutoff level. A more detailed editing is done by other users. Here are options to use RFLAG and SPFLG. Maybe you'd like to RFLAG first and then eyeball with SPFLG. The options are endless! The first execution of RFLAG determines flagging level parameters and makes plots on the TV for you to be sure that they are reasonable, The second execution applies these levels and determines new levels in case you want to run RFLAG even more times.

> default rflag ↵
> indi d; getn 3 ↵
> sources(1) '3C286' ↵
> docal 1 ↵ — to correct delays and a-priori gains.
> avgchan 11 ↵ — to set the width of the median-window filter run across the spectral channels in each IF
> fparm = 3, x, -1, -1 ↵ — to do a small rolling time buffer, set the sample interval, and to have NOISE and SCUTOFF set IF-dependent cutoff levels.
> fparm(13) = 1000 ↵ — to clip amplitudes at 1000 Jy.
> doplot 12 ; dotv 1 ↵ — to do the 2 most useful plots on the TV.
> go ↵

RFLAG will return cutoff levels to the appropriate adverbs. Therefore, to apply the flags using the flagging levels found above, do *not* do a `tget rflag` which will reset the levels to zero. Instead, simply do

> doplot = -12 ↵ — to apply the cutoff levels and determine new ones, plotting on the TV.
> go ↵

RFLAG will make a new flag table each time it determines flags (when DOPLOT $\leq$ 0). RFLAG has numerous FPARM options you may choose. FPARM(9) and FPARM(10) scale the rms for NOISE and SCUTOFF that RFLAG computes. If the fraction of a channel that is flagged exceed FPARM(7), the entire channel is flagged. FPARM(6) is especially useful at P-band as it allows channels adjacent to an RFI peak to be flagged. Similarly, FPARM(8) allows channels between strong RFI peaks to be flagged. Read the help file for detailed information. Flagging calibrator scans aggressively is recommended especially for spectral-line observations.

Check the results with

> default spflg ↵
> indi d; getn 3 ↵
> sources(1) '3C286' ↵ — to examine only the bandpass calibrator.
> dparm(6) = x ↵ — to set sampling time in seconds
> docal 1 ↵ — to correct delays and a-priori gains.
> go ↵

It is at this point that experienced users differ significantly in their approach. If you have only one bandpass-calibration scan and the ionosphere was reasonably behaved (phases nearly constant through that scan), then you should run BPASS on the calibration source in place. However, when one has multiple bandpass scans and/or a badly behaved ionosphere, it may be best to split the calibration scans out and self-calibrate them on a short time interval. The addition of multiple scans where the source has been phased up, but the remaining RFI has not, will reduce the effect of that RFI substantially. This scheme is described next.

18. Split off your calibrator for bandpass calibration — this is only necessary for self-calibrating the BP calibrator.
    > default split ↵
    > indi d; getn 3 ↵
    > outdi indi ; source '3C286' ' ↵ — to select calibrator and stay on the same disk.
    > outcl 'bpspl' ↵ — to indicate the use for this file.
    > docalib 1 ↵ — to apply delay and a-priori calibration.
    > go ↵ — To make a single-source file.

19. Self-cal on your bandpass cal with CALIB to solve for wiggles in phase as a function of time.
    > default calib ↵
    > indi d; getn 4 ↵
    > refant 22 ↵ — to stay with the chosen reference antenna.
    > solint y ↵ — to average n records in each solution interval, set $y = nx$.
    > solty 'L1R' ; solmode 'P' ↵ — to do phase-only solutions.
    > aparm(6) 1 ↵ — to get some diagnostic messages.
    > go ↵ — to write a new SN table.

    Eyeball this new table with SNPLT — should be flat, not all over the place. But EDITA lets you look at more things at once.
    > default edita ↵
    > indi d; getn 4 ↵
    > inext 'sn' ; dotwo 1 ↵ — to plot the phases in the new table.
    > crowded 1 ; do3col 1 ↵ — to combine all IFs in each colorful plot.
    > go ↵

    If these plots are too crowded, you can interactively select individual IFs and polarizations. This task will let you flag data if desired. If you do, then you should re-run CALIB.

20. Use BPASS to calibrate the bandpass shape — corrects for wiggles in phase and amplitude as a function of frequency
    > default bpass ↵
    > indi d; getn 4 ↵
    > docalib 1 ↵ — to apply the new SN table phases.
    > refant 22 ↵
    > solty 'l1r' ; solint = -1 ↵ — to average the entire time range.
    > bpassprm(5) = 1; bpassprm(10) = 3 ↵ — to normalize the solutions at the end rather than the data record-by-record.
    > go ↵ — To generate a new BP table containing one record per antenna.

21. Eyeball the resulting bandpass (BP) table — APARM(8) = 2 plots the BP table in POSSM. If you see odd bandpass shapes or spikes in the bandpass, you may need to edit your data more extensively and repeat steps 17 through 20.
    > tget possm ↵
    > indi d; getn 4 ↵
    > solint = -1 ↵
    > aparm(8) 2 ; aparm(9) 1 ↵ — to plot all IFs together for each bandpass.

> nplots 2 ↵ — 2 per page is enough here.
> go ↵

Alternatively, BPEDT lets you use the bandpass table to guide additional flagging of the bandpass calibrator(s). It is similar to EDITA in that it lets you look at multiple antennas at the same time for comparison.

> default bpedt ↵
> indi d; getn 4 ↵
> inext 'bp' ; GO ↵

22. Copy the bandpass table back to your main data with TACOP.

> default tacop ↵
> indi d; getn 4 ↵
> inext 'bp' ↵
> outdi indi ; geton 3 ↵ — to select output name parameters
> go ↵

23. Since RFLAG probably made a large number of flags, let us copy the data file with UVCOP and apply the flags.

> default uvcop ↵
> indi d; getn 3 ↵
> outdi indi ; outn inna ↵ — to stay on d.
> outcl 'fixcp' ↵ — with a meaningful class name
> flagver n ↵ — to apply the highest flag table (n); use IMHEAD if unsure. Version 0 means no flagging here.
> go ↵

24. Use SETJY to compute the flux for the primary flux calibrator (3C286), storing it in the SU table.

> default setjy ↵
> indi d; getn 5 ↵
> source '3C286' ' ↵ — specify all "known" calibrator sources.
> opty 'calc' ↵ — to calculate and display fluxes from the frequencies plus tables of known sources.
> go ↵

25. Go through all of your calibration sources, one at a time, to check for RFI with POSSM and, if present, to address it with SPFLG and/or RFLAG

> tget possm ↵
> source 'calsrc_i' ' ↵ — to examine the i^{th} calibration source.
> indi d; getn 5 ↵
> docal 1 ; doband 1 ↵ — apply our hard-won calibration.
> solint 0 ↵ — to look at the average over all time.
> aparm(9) 1 ↵ — to plot all IFs together.
> nplots 2 ↵ — 2 per page is enough here.
> go ↵

If RFI is present (it almost certainly will be), two rounds or more of RFLAG may be needed.

> default rflag ↵
> indi d; getn 5 ↵
> source 'calsrc$_i$' ' ↵ to examine the $i^{\rm th}$ calibration source.
> docal 1 ; doband 1 ↵ apply the calibration.
> avgchan 11 ↵ to set the width of the median-window filter run across the spectral channels in each IF
> fparm = 3, x, -1, -1 ↵ to do a small rolling time buffer, set the sample interval, and to have NOISE and SCUTOFF set IF-dependent cutoff levels.
> fparm(13) = 1000 ↵ to clip amplitudes at 1000 Jy. There are many more FPARM options you may choose also.
> doplot 12 ; dotv 1 ↵ to do the 2 most useful plots on the TV.
> go ↵

RFLAG will return cutoff levels to the appropriate adverbs. Therefore, to apply the flags using the flagging levels found above, do *not* do a `tget rflag` which will reset the levels to zero. Instead, simply do

> doplot = -12 ↵ to apply the cutoff levels and determine new ones, plotting on the TV.
> go ↵

RFLAG will make a new flag table each time it determines flags (when DOPLOT ≤ 0). It may be good to check what RFLAG has done with POSSM. If more edits are needed, use SPFLG

> default spflg ↵
> indi d; getn 5 ↵
> source 'calsrc$_i$' ' ↵ to examine the $i^{\rm th}$ calibration source.
> dparm(6) x ↵
> docal 1 ; doband 1 ↵ apply the calibration.
> go ↵

26. Since RFLAG probably made a large number of flags, let us copy the data file once again with UVCOP and apply the flags.
> default uvcop ↵
> indi d; getn 5 ↵
> outdi indi ; outn inna ↵ to stay on d.
> outcl 'flaged' ↵ with a meaningful class name
> flagver n ↵ to apply the highest flag table (n); use IMHEAD if unsure. Version 0 means no flagging here.
> go ↵

27. Since all flag tables $< n$ were already merged into table n which we have applied, delete all flag tables in the output file.
> default extdest ↵
> indi d; getn 6 ↵
> inext 'fg' ; invers = -1 ↵ to delete all remaining versions of flag table.
> extdest ↵

28. Find the amplitude and phase calibration as a function of *time* using CALIB

> default calib ↵
> indi d; getn 6 ↵
> calcode '*' ↵ to select all calibration sources
> docalib 1 ; doband 1 ↵ to apply all existing calibration.
> refant 22 ↵ to keep our reference antenna.
> solint 0 ↵ One solution per calibrator scan is okay with good short-term phase stability. Otherwise, set to the longest time in minutes over which phase is stable.
> solty 'L1R' ; solmode 'a&p' ↵ to solve for amplitude and phase with an L1 method iterated robustly.
> aparm(6) 1 ↵ to print closure error statistics.
> go ↵

Pay attention to the fraction of failed solutions. If the fraction is large, try adjusting parameters such as SOLINT, etc.

29. The solution table (SN) produced by the previous step will contain amplitude gains at multiple levels, correct ones for those sources with fluxes in the source table (3C286) and incorrect ones for those which CALIB was forced to call 1 Jy. To solve for the unknown fluxes of the secondary calibrators

> default getjy ↵
> indi d; getn 6 ↵
> source 'calsrc$_1$', 'calsrc$_2$', 'calsrc$_3$', ... ↵ to list the secondary calibrators.
> calsour = '3c286' ' ↵ to list the known calibrator(s).
> snver = n ↵ to select only the SN table just written by CALIB. Use IMHEADER to find the maximum version number n.
> go ↵ to solve for the secondary fluxes as functions of IF.

Check the values displayed carefully to make sure that they are sensible. SOUSP offers tools to examine and adjust the fluxes and SN table n.

30. Look at both amplitude and phase solutions with EDITA or SNPLT.

> tvini ↵
> default snplt ↵
> indi d; getn 6 ↵
> inext 'sn' ; opty 'amp' ↵ to plot amplitudes.
> nplots 4 ; dotv 1 ↵ to plot 4 panels on each TV page.
> opco 'alif' ; do3col 1 ↵ to plot all IFs in each panel using color to distinguish them.
> go ↵

One can also plot IFs in separate panels or one at a time if these plots are too crowded. Then

> opty 'phas'; go ↵ To check the phases.

To use EDITA to look at both amplitude and phase at the same time for multiple antennas

> default edita ↵
> indi d; getn 6 ↵
> inext 'sn' ; dotwo 1 ↵ to plot the phases in the new table.
> crowded 1 ; do3col 1 ↵ to combine all IFs in each colorful plot.
> go ↵

If these plots are too crowded, you can interactively select individual IFs and polarizations. This task will let you flag data if desired. If you do, you should then re-run CALIB.

31. Apply the solutions to the calibration table with CLCAL
    - > default clcal ↵
    - > indi d; getn 6 ↵
    - > calcode '*' ↵ — to select all calibration sources
    - > sampty 'box' ; bparm 0.3, 0.3 ↵ — To smooth the SN table if you used SOLINT less than full scans. Leave this out if you used scan averages.
    - > refant 22 ↵
    - > go ↵

32. Use SNPLT to check that the solutions have been interpolated between calibrators.
    - > tget snplt ↵
    - > inext 'cl' ↵
    - > go ↵

33. Make an image of 3C286 as a sanity check using IMAGR
    - > default imagr ↵
    - > indi d; getn 6 ; outdi indi ↵
    - > docal 1 ; doband 1 ↵ — to apply the calibration.
    - > source = '3c286' ' ↵ — to select only the primary calibrator.
    - > bchan 5; echan = $n - 4$ ↵ — to omit edge channels in each IF, where n is the number of channels in an IF, now usually 128.
    - > nchav = echan-bchan+1; chinc nchav ↵ — to average all spectral channels into the image.
    - > cellsize 1. ↵ — to set the image cell spacing — depends on configuration.
    - > imsi 1024 ↵ — to make a largish image to see sources around 3C286.
    - > outn '3C286_quick' ↵
    - > niter 100 ; dotv 1 ↵ — to do some Cleaning and guide the progress with the TV
    - > go ↵

34. If you wish to image Stokes Q and U, it is now necessary to calibrate the polarization. This is still somewhat of a research question in *AIPS*, but the advice in §4.3.16 may work. You will need to run procedure VLATECR to correct for Faraday rotation and dispersion, task RLDLY to correct for the V minus H delay offset, task PCAL with SPECTRAL 1 to calibrate the polarization D terms, and then task VHDIF to calibrate the V minus H phase offset which is also a function of spectral channel. PCAL is best run with an unpolarized calibration source which is common at P band. However, VHDIF requires a calibration source with a useful flux in Stokes U. 3C345 has this, but, due to the rotation measure, not at all frequencies. Significant frequency smoothing may help. Task VHMOD models a point source with linear polarization which may help you model your source.

35. At this point, it may be simpler to SPLIT the data set and apply calibration to the target sources which we then can work with one at a time.
    - > defsult split ↵
    - > indi d; getn 6 ↵
    - > outdi indi ↵
    - > docal 1 ; doband 1 ↵

> dopol 1 ↵ — if you calibrated polarization
> sources = 'target$_1$', 'target$_2$', ... ↵ — to select only target sources.
> go ↵

To create a number of separate, single-source uv data sets each with separate target source.

36. Flag RFI and bad data out of the target data set with two or more rounds of RFLAG

> default rflag ↵
> indi d; getn 9 ↵
> avgchan 11 ↵ — Median window width on spectra.
> fparm = 3, x, -1, -1 ↵ — to do a small rolling time buffer, set the sample interval, and to have NOISE and SCUTOFF set IF-dependent cutoff levels.
> doplot 12 ; dotv 1 ↵ — To examine the two important plots on the TV.
> go ↵

and then apply the flags levels and determine new ones

> doplot -12 ; go ↵

and perhaps repeat this line for more flagging. *NOTE: for flagging spectral-line target sources, you should set* FPARM(4)=0 *since you should not flag in the spectral domain.* Check what RFLAG has done with SPFLG, which can be tedious.

> default spflg ↵
> indi d; getn 9 ↵ — to select the first of the split target data sets.
> dparm(6) x ↵ — to set integration time.
> go ↵

37. Because we used TYAPL only to apply the re-quantizer gains, your data weights simply reflect integration time. For better imaging, weights should reflect the actual data uncertainties. So re-weight your data with REWAY

> default reway ↵
> indi d; getn 9 ↵
> outdi indi ↵
> aparm 31,0,500 ↵ — to find rmses from rolling buffers 31 time intervals long on a baseline basis and then smooth them over 500 seconds.
> go ↵

Note that this applies the final SPFLG+RFLAG flag table. Delete the other flag table versions with

> default extdest ↵
> indi d; getn 10 ↵
> inext 'fg' ; invers = -1 ↵ — to delete all remaining versions of flag table.
> extdest ↵

L.2 P-band imaging in AIPS

38. The following steps relate to imaging your calibrated data. As the aim of this appendix is calibration, the imaging will be discussed only briefly here. The following items refer to continuum imaging. Make a dirty image of the center with IMAGR
    - > default imagr ↵
    - > indi d; getn 10; outdi indi ↵
    - > bchan 5; echan = $n-4$ ↵ — to omit edge channels in each IF, where n is the number of channels in an IF.
    - > nchav = echan-bchan+1; chinc nchav ↵ — to average all spectral channels into the image.
    - > cellsize 10 ↵ — to set a cell size which will depend on the VLA configuration.
    - > imsi 1024,1024 ↵
    - > niter 0 ↵ — to do no Cleaning
    - > go ↵

39. Clean image with faceting, auto-boxing, and TV display; we use DEFAULT IMAGR here to initialize all of IMAGR's many parameters
    - > default imagr ↵
    - > indi d; getn 10; outdi indi ↵
    - > task 'setfc' ↵
    - > cellsize 0 ; imsize 0 ↵ — to have SETFC determine the correct cell and image size for these data.
    - > bparm 2.2, 5, 2, 8, 0.3 ↵ — to image fully out to a radius of 2 degrees and to include 0.3 Jy NVSS sources out to a radius of 8 degrees.
    - > flux 1 ↵ — to include exterior sources above 1 Jy only.
    - > boxfile 'MYAREA:Target1.box ↵ — to list the facets found.
    - > go ↵

    The following does not use default since we need adverbs set for, and by, SETFC, such as inname, imsize, and cellsize
    - > task 'IMAGR' ↵
    - > flux 0 ↵
    - > bchan 5; echan = $n-4$ ↵ — to omit edge channels in each IF, where n is the number of channels in an IF.
    - > nchav = echan-bchan+1; chinc nchav ↵ — to average all spectral channels into the image.
    - > niter 5000 ↵ — to Clean 5000 iterations.
    - > do3dim true ; overlap 2 ↵ — to use true 3-D geometry, Cleaning one facet at a time in each cycle (recommended).
    - > dotv true ↵ — to control the imaging with the TV.
    - > oboxfile boxfile ↵ — to save all changes to Clean boxes.
    - > im2parm=2,0 ↵ — to allow up to 2 Clean boxes to be created automatically at each cycle.
    - > go ↵

    Note, the field may have enough spectral index variation to require imaging (and self-calibration) of each spectral window (IF) individually.

40. There is normally enough flux in a P-band field to do phase self-calibration and we recommend that you now do that. Use CALIB with the output of step 39 as the model including all NFIELD facets as the NMAPS images.

> default calib ↵
> indi d; getn 10; outdi indi ↵
> nmaps = nfield ↵ — to include all facets
> in2d = indisk; get2n M ↵ — to set the 2nd file name to the output image of class ICL001 (catalog number M).
> soltyp 'l1r' ; solmode 'p' ↵ — to start with phase-only solutions.
> solint = nx ↵ — to set a short integration time as n times the sampling interval.
> go ↵

41. Examine the output SN table with EDITA to see if there are times with bad solutions. If not, repeat steps 39 and 40 using

> tget imagr; docal = 1; go ↵ — to make new images.
> tget calib; in2seq = 2 ↵ — to point at the second set of images.
> solmode 'a&p' ↵ — to do amplitude as well *if there is enough flux in the field*.
> go ↵

Re-examine the new SN table with EDITA. In particular, watch for periods of time when the amplitude gain of an antenna gets rather larger than normal. It has been found that the ionosphere can defocus sources on scales as small as the antenna diameter. It is best to delete such data.

42. Spectral-line imaging is both simpler and more complicated. One may image each spectral channel individually. However, even for absorption experiments, it may be best to image all channels free of the spectral line together to make a continuum image and model. This will almost certainly require multiple facets and extensive cleaning but it will produce a single continuum model that can be applied to all spectral channels with UVSUB. Spectral index in the continuum may force you to do this one spectral window at a time. You can add the continuum image back to the cube with COMB.

At this point, run IMAGR on the output of UVSUB one spectral window at a time over all spectral channels of interest with NCHAV=1; CHINC=1. You may be able to limit the field of view imaged to one facet surrounding the target source.

43. The task TVSPC is a lovely way to explore the output image cube. To run this task, we must first transpose the cube

> default trans ↵
> indi d; getn C; outdi indi ↵ — where C is the catalog number of the output cube from IMAGR.
> transcod = '312' ↵ — to put frequency first.
> go ↵ — to make a transposed cube.

One also needs a "meaningful" single-plane image, perhaps the central facet of the continuum image made above. Or we can compute the first moment image.

> default xmom ↵
> indi d; getn T; outdi indi ↵ — where T is the catalog number of the output cube from TRANS.
> flux = f ↵ — to compute only with brightness $> f$ Jy/beam.
> go ↵ — to amke a first moment image.

Finally

> default tvspc ↵
> indi d; getn M; outdi indi ↵ where M is the catalog number of the meaningful image, perhaps the output plane from XMOM.
> in2di d; get2n T ↵ where T is the catalog number of the output cube from TRANS.
> in4di d; get4n C ↵ where C is the catalog number of the output cube from IMAGR.
> go ↵ Have fun!

This allows you to look at spectra from the cube at individual spatial pixels as well as the cube as an image at individual frequencies. Enter EXPLAIN TVSPC or see *AIPS*Memo 120 for details.

44. Make image(s) and final calibrated data sets into FITS files with FITTP. Exit AIPS and
> setenv FIT /lustre/mmao/Pband ↵ for tcsh and c shells
> export FIT=/lustre/mmao/Pband ↵ for bash shell

Then start AIPS again
> default fittp ↵
> indi d; getn M ↵ to select an image or data set
> dataout 'FIT:VirA.FITS ↵ to make an appropriate output name
> go ↵

And repeat with new M and DATAOUT values.

L.3 Additional Recipes

L.3.1 Banana storage

Bananas ripen after harvesting. They do it best at room temperature. Because of this there are three stages to banana storage.

1. **On the counter:** When you buy a bunch of bananas that are not exactly at the ripeness you want, you can keep them at room temperature until they are just right for you. Be sure to keep them out of any plastic bags or containers.
2. **In the refrigerator:** If there are any bananas left, and they are at the ripeness you like, you can put them in the refrigerator. The peel will get dusty brown and speckled, but the fruit inside will stay clear and fresh and at that stage of ripeness for 3 to 6 days.
3. **In the freezer:** If you want to keep your bananas even longer, you can freeze them. Mash the bananas with a little lemon juice, put them in an air tight freezer container and freeze. Once they're defrosted, you'll go bananas baking bread, muffins and a world of other banana yummies. Or, you can freeze a whole banana on a Popsicle stick. When it is frozen, dip it in chocolate sauce, maybe even roll it in nuts, then wrap it in aluminum foil and put it back in the freezer. Talk about a scrumptious snack.

F File Sizes

Your data reduction strategy will be more effective if you have an idea of how big the data and map file sizes will be on disk. Also, it will help you estimate just how many files you can backup to tape. In this appendix, we will discuss file sizes in bytes, which are 8 bits in size, rather than "blocks," which can vary in size between different computers. Inside $\mathcal{AIPS}$, the definition of "byte" is perverted, but all systems now use a 1024-byte (8192 bit) "block."

F.1 Visibility (*uv*) data sets

uv data files contain information about the coherence function of a wavefront at random locations and random times. Consequently, the way this information is stored on disk is different from that for images where the pixels are on a regular grid. AIPS *uv* data are stored on disk in a manner similar to the way that it would be organized on a FITS "random group" tape. The data are stored as logical records; each record (a "visibility") contains **all** the data taken on one baseline at a given time. Consequently, a record may contain information for several IFs, several frequencies at each of those IFs and more than one polarization combination for each frequency/IF. The first part of each logical record contains what are known as "random parameters" *e.g.*, spatial frequency coordinates and time. After the random parameters, there is a small, regular array of data.

For a multi-source data set such as might be created by FILLM, the random parameter group will include the following. UU-L-SIN, VV-L-SIN, and WW-L-SIN give the spatial frequency coordinates, computed with a sine projection in units of wavelengths at the reference frequency. TIME1 is the label for the time in days. BASELINE is the baseline number ($256ant_1 + ant_2 + subarray/100.$) and SOURCE is the source number corresponding to an entry in the source table. If you have frequency table identifiers (which is usually the case these days), then there will be an additional random parameter, FREQSEL. For a compressed database, two additional random parameters will be required — WEIGHT to give a single data weight for all samples in the record and SCALE to give the gain used to compress that record.

The regular data array is similar to an image array in that the order of axes is arbitrary. However, the convention is for the first axis to be of type COMPLEX, having a dimension of 3 for uncompressed data (real, imaginary, weight) and a dimension of 1 for compressed data. The other axes of the regular array are IF, RA, DEC, FREQ and STOKES.

F.1.1 *uv* database sizes

The number of words in each "visibility" is given by

$$(\#\ random\ parameters) + [(dimension\ of\ \texttt{COMPLEX}\ axis) \times (\#\ polns) \times (\#\ freqs) \times (\#\ IFs)]$$

The size of the database to be loaded to disk with FILLM is given by

$$(word\ per\ record) \times (\#\ vis) \times 4\,\text{bytes}$$

The number of visibilities is given (approximately) by

$$\frac{length\ of\ observation}{integration\ time} \times number\ of\ baselines$$

where the number of baselines is the usual $\frac{1}{2}n(n-1)$. This is equal to 351 for the VLA for the usual 27-antenna array.

For example, a 12-hour observation with 30 second integrations, one frequency and 2 IFs with RR, RL, LR and LL written in compressed format (DOUVCOMP = TRUE in FILLM) will occupy about 34 Mbytes on disk. In practice, the *uv* file will usually be a little larger due to the way the system allocates space on the disks. You must also remember to allow room for the extension tables — see § F.3. If this database had been written in uncompressed format, the *uv* data would have occupied around 62 Mbyte, but would have carried information about different data weights for different IFs.

Consider another example illustrated by the IMHEAD listing below:

```
Image=MULTI     (UV)         Filename=20/07/90    .L BAND.   1
Telescope=VLA                Receiver=VLA
Observer=AFTST               User #= 1364
Observ. date=20-JUL-1990     Map date=23-JUL-1990
# visibilities    105198     Sort order  TB
Rand axes: UU-L-SIN  VV-L-SIN  WW-L-SIN  BASELINE  TIME1
           SOURCE  FREQSEL  WEIGHT  SCALE
-----------------------------------------------------------------
Type    Pixels   Coord value  at Pixel    Coord incr   Rotat
COMPLEX      1   1.0000000E+00    1.00 1.0000000E+00    0.00
STOKES       4  -1.0000000E+00    1.00-1.0000000E+00    0.00
IF           2   1.0000000E+00    1.00 1.0000000E+00    0.00
FREQ         1   1.4524000E+09    1.00 2.5000000E+07    0.00
RA           1    00 00 00.000    1.00      3600.000    0.00
DEC          1    00 00 00.000    1.00      3600.000    0.00
-----------------------------------------------------------------
Maximum version number of extension files of type HI is   1
Maximum version number of extension files of type AN is   1
Maximum version number of extension files of type NX is   1
Maximum version number of extension files of type SU is   1
Maximum version number of extension files of type FQ is   1
Maximum version number of extension files of type CL is   1
Maximum version number of extension files of type SN is   2
```

This compressed (COMPLEX Pixels = 1) *uv* database contains 9 random parameters, 4 polarizations, 1 frequency, and 2 IFs. for *each* of 105198 visibilities. The size of the database file itself is, therefore,

$$\Big[\{9 + [1 \times 4 \times 1 \times 2]\} \times 105198 \times 4\Big] \text{ bytes} = 7.153\,\text{Mbytes}.$$

Note that this data set is 17/33 the size of an uncompressed data set.

F.1.2 Compressed format for *uv* data

The use of "compressed" data can make substantial savings in the amount of disk space that you require, particularly for spectral-line databases. All tasks should now be able to handle either the compressed or the uncompressed formats. Compressed data files can be identified by the dimension of 1 for the COMPLEX axis in the database header. (Uncompressed data will have a dimension of 3.) The savings can be close to a factor of three for spectral line observations.

This is achieved by converting all data weights into a single WEIGHT random parameter, by finding a single SCALE random parameter with which to scale all real and imaginary parts of the visibilities into 16-bit integers, and by packing the real and imaginary terms into one 32-bit location using magic-value blanking for flagged data. This is to be compared with the uncompressed format in which each of the real, imaginary and weight terms are each stored in a 32-bit floating-point location. The use of a single weight value masks

real differences in system temperatures between polarizations and IFs, which one should retain for the lowest possible noise in imaging.

In general, data compression is a good thing and should be used, but with a little caution. With a single frequency, single IF, and single polarization, you will not save any disk space. In all other cases, there are respectable savings to be made. However, the use of a packed data word for the real and imaginary parts of the visibility function along with magic value blanking imposes a restriction on the "spectral dynamic range" of the data set of around 32000:1. Consequently, there are some situations where compressed data should *not* be used. For example, if the spectral dynamic range in the *uv* database is likely to be greater than, say, 1000:1, you must use *uncompressed* data format to avoid loss of accuracy. This situation can arise in maser spectra, for example, in which there are maser lines of 1 Jy and > 32000 Jy; in this case, you should never use compressed data. Bandpass calibration can cause large correction factors to be applied to the edge channels of a database. In the presence of noise or interference, bad channels can become very much greater in amplitude than good channels. In such cases you must either use uncompressed format or be very careful to flag bad channels or to drop them with the BCHAN and ECHAN adverbs *as* you apply the bandpass calibration. In general, continuum data sets can be loaded with data compression since these dynamic range considerations will not normally apply. The loss of weight information may not be worth the savings in disk space, however. Compressed data takes less time to read/write due to the small number of bytes, but casts in cpu time some due to the extra computation.

If there has been on-line or later flagging that depends on polarization, IF, or spectral channel (*i.e.*, RFI excision) or differences in the intrinsic weights between polarizations, IFs, or spectral channels (*i.e.*, different system temperatures for different IFs), then data compression causes a serious loss of information related to the data weight.

F.2 Image files

Since images are regular arrays, the sizes of image files are easier to calculate. The images are stored as floating-point numbers, *i.e.*, 32 bits per pixel, so the image file size in bytes is given by

$$4 \times \prod_i length(i)$$

where $length(i)$ is the number of pixels on the i^{th} axis. For example, a 128-channel cube with each plane a 256×256 image will require around 34 Mbytes of disk storage space. This may be increased a little due to the way in which the system allocates space for files.

F.3 Extension files

Subsidiary data about *uv* database and image files are written in "extension files." These include, for example, history records (HI files), plot instructions (PL files), calibration solutions (SN files), ad nauseum. Some extension files can become large. A history file uses 1024 bytes for every 14 history records, a small amount under most circumstances. A plot file is normally small, but the output from GREYS can be as large as one plane of the image and the output of KNTR is larger by a factor of the number of panes in the plot.

The CL (calibration) table contains the total model of the interferometer at each interval. Many of these logical records are blank in the case of most interferometers but, for the VLBA, these records will contain essential information. The CL table contains a logical record for each antenna in the array at each CL time

stamp. The CL time stamps are set by the user when loading the data. The default for VLA data is every 5 minutes. For VLBI data, it is every 1 minute. Each CL logical record requires

$$\{15 + [14 \times (\#\, IFs) \times (\#\, pols)]\} \times 4 \,\text{bytes}$$

For a 12-hour observation with the VLA with 2 IFs and 4 polarization pairs, with entries every 5 minutes, the CL table will occupy about

$$\left[\{15 + [14 \times 2 \times 2]\} \times 4 \times \frac{(12 \times 60)}{5} \times 27\right] \text{bytes} = 1.05\,\text{Mbytes}$$

Since most other files are considerably smaller, their sizes can be ignored. That they exist and may require some disk, should not be forgotten. To look at the full *AIPS* disk usage on your computer in summary form:

> TASK 'DISKU' ; INP ↵	to review the inputs.
> INDISK 0 ; DETIME 0 ↵	to look at all disks and all data sets
> USERID 32000 ↵	to look at all user numbers.
> DOALL FALSE ; GO ↵	to run the task in a summary mode.

Then to look at file sizes in detail:

> INDISK n ; USERID 0 ↵	to restrict the display to your data sets and one disk.
> DOALL TRUE ; GO ↵	to run the task to list all files on disk n.

F.4 Storing data on tape

Images and *uv* databases are written to magnetic tape by FITTP for archival purposes and for transfer to other computers and sites. Three FITS-standard formats are available, controlled through the adverb FORMAT. The preferred format is 32-bit floating point (IEEE standard) format. There are no dynamic range limitations in this format and, on many modern computers, no bit manipulation is required since they use IEEE floating internally.

Of the two integer formats, there is little reason to use the 32-bit integer since it poses dynamic range, re-scaling, and other problems with no saving in space. The 16-bit integer format uses 16-bit signed 2's complement integers to represent the data. Such numbers are limited to the range -32768 to 32767. FITTP has to find the maximum and minimum in the image and then scale the data to fit in this numeric range. For images of limited dynamic range, this format is perfectly adequate. In fact, FITAB offers the option to reduce the dynamic range even further with the QUANTIZE adverb. For images written to FITS disk files, this allows for better compression before the files are transmitted over the Internet. For high-dynamic range images, the 16-bit format may not be adequate. (The integer formats are no longer allowed for *uv* data. More than one user has reduced all his "good" spectral channels to pure 0 by scaling all the *uv* data to include one really horrendously bad sample.) A less important benefit of the floating point format is that the numbers representing your data are recorded exactly on tape as they are stored on disk; there are no "quantization errors". This may be important for software development.

The preceding paragraphs do not tell the full story, however. The portion of the FITS standard used by FITTP does not allow for *uv* data on tape in a compressed format. Instead, FITTP expands the data into the uncompressed form and then writes the data on tape. In the conversion, the real and imaginary values that were stored in one packed number are expanded into three real values — one each for real, imaginary and weight terms — and the weight and scale random parameters are removed since they are no longer required. Consequently, the compressed data are expanded to

$$\frac{\{(\#\, random\ parameters - 2) + [(\#\, pol) \times (\#\, IFs) \times (\#\, frequencies) \times 3]\}}{\{(\#\, random\ parameters) + [(\#\, pol) \times (\#\, IFs) \times (\#\, frequencies)]\}}$$

the original size (where $\#\ random\ parameters$ is the original number in the compressed database).

As an example, let us consider a multi-source spectral-line database stored on disk in compressed format. The data set has seven channels each at 2 IFs with 2 polarizations. There are nine random parameters and 834031 visibilities. From § F.1, we can calculate the size of the uv file to be 123 Mbytes. (Remember, this doesn't include any of the extension files, some of which might be several Mbytes in size.) Before the file is written to tape in 32-bit floating format, it is first expanded by a factor of

$$\frac{\{(9-2)+[2\times 2\times 7\times 3]\}}{\{9+[2\times 2\times 7]\}} = 2.333\,.$$

Consequently, the data will occupy

$$123\times 2.333\,\mathrm{Mbytes} = 287\,\mathrm{Mbytes}$$

on tape. In other words, this database and all the associated extension files will not fit on a standard, 6250 bpi tape even using BLOCKING = 10, but modern tape technology solves this problem.

Note that FITTP writes history file data into the FITS header and writes table extension files as extensions after the main image or data set within the same tape file. Plot (PL) and slice (SL) files may be saved to FITS files beginning in 31DEC24 and 31DEC25. Set DOPLOT to 1. Note that the form used for slice and plot files will not be usable to any software other than *AIPS*.

Task FITAB uses "binary tables" to represent visibility data rather than the old, mildly deprecated "random groups" form of the FITS format. This has several advantages for uv data. It allows compressed data to be recorded in that form exactly (except for byte-order questions which should not concern the reader). It also allows the data and attached tables to be divided into pieces which will reduce the size of files to be copied over the Internet, making copying and tape storage somewhat more reliable. The principal disadvantage of FITAB uv data output is that only two packages can read it so far. The one "outside" package that reads the format is obit available from Bill Cotton at NRAO Charlottesville.

F.4.1 DAT and Exabyte tapes

The arrival of modern tape technologies has hastened the demise of 9-track tapes. First Exabyte (8mm) and then DAT (4mm) have provided much higher storage capacities than the 9-track tapes and have also provided faster seeks between file marks and greater data reliability. The new technologies are very much cheaper as well, in part because they have been adopted by the PC market. They are both technically quite complex internally. The DAT tape has a "system log" area at the beginning which allows for the fast seeks. It is a bit fragile, however, since it is updated when the tape is unloaded and hence can be incorrect if there is an unfortunate power failure. Both technologies are still evolving and both now offer various data encoding/compression options. Unfortunately, the data compression techniques vary considerably with tape model and manufacturer and hence should not be used to archive or transport data. The data are blocked on the tapes by means known only to the manufacturers and are not significantly under user control. It is still probably good to use a large BLOCKING, but only for I/O transfer reasons. The EOF marks can be expensive on these tape devices.

Exabytes at low density have a capacity of about 2.2 Gbytes on a 112m tape and use about 1 Mbyte (or maybe even 4 Mbytes) for each EOF mark. The large size of the EOF limits the number of files you can write rather significantly. The EOFs are also slow to process mechanically. Exabytes at high density have a capacity of 4.5 Gbytes on a 112m tape and use 48 Kbytes per EOF mark. DATs have a capacity of 2.0 Gbytes on a 90m tape, but also come in 60m and 120m sizes. The EOF mark size is not readily available, but is probably no more than 48 Kbytes. The early warning of the end-of-medium is 40 Mbytes before the actual end of tape.

F.5 Very large data sets

FITTP and FITAB cannot write multi-volume tapes. Some spectral-line and VLBA databases (and perhaps some continuum databases) may be so large that the file cannot fit on one modern tape even with BLOCKING = 10. What can you do to backup your data? The simplest solution is to use FITAB. This task can write *uv* data in compressed form and can break up a data set into "pieces." You write as many pieces as will fit on the first tape, noting what the piece number is when the end-of-tape is reached. You can then tell FITAB to begin with that piece number on the next tape (BDROP = n-1 where n is the number of the piece that encountered the end-of-tape). Unfortunately, the FITS format used by FITAB, while perfectly legitimate, is only understood by *AIPS* versions beginning with 15APR99. Furthermore, FITAB output is only understood by *AIPS* and OBIT. If the data set is so large that it will not fit on the required tape device using FITTP or the data must be taken to a system that does not understand FITAB's format, there are several approaches that you can adopt.

First, you could SPLIT out the database into *single-source* databases and back each of these up individually. Alternatively, you could subdivide the large database in several smaller databases with UVCOP by specifying a different time range for each of the smaller databases and then back these up individually. Another way to solve the problem is to realize that the calibration and flagging information that you have carefully generated during the calibration is contained in the extension tables — the raw data that you loaded is not modified until you finally SPLIT out the individual sources. Consequently, you can write create a dummy *uv* database to which all the extension tables are attached with the task TASAV, then save this "database" on tape with FITTP. The raw visibilities can be saved in the form of copies of the archive tapes.

F.6 Additional recipes

F.6.1 Banana stuffing

1. Pare and rub 4 **bananas** through a sieve into bowl.
2. Add 1/2 grated **onion**, 1 **green pepper** chopped fine, 3 tablespoons finely chopped **parsley**, 4 slices cooked **bacon** chopped fine, 1 1/4 cups **bread crumbs**, pinch of **thyme**, 1 teaspoon **salt**, and 1 **egg**.
3. Mix thoroughly, fill 1 **chicken**, and roast in the usual manner.

F.6.2 Banana nut bread

1. Cream 1 cup **sugar** and 1/2 cup **margarine** together.
2. Add 2 **eggs**, 2 cups **flour**, 1/2 teaspoon **salt**, and 1 teaspoon **baking soda** and mix thoroughly.
3. Add 1 cup chopped **nuts** (walnuts or pecans), 3/4 cup mashed **bananas**, and, lastly, 4 teaspoons **sour milk** and mix well.
4. Put in greased loaf pan.
5. Bake in 350° F oven for 1 hour.

V VLA Maintenance Tasks in *AIPS*

Most tasks in *AIPS* are designed to work for a generalized interferometer or single-dish. However, a number of tasks have been written specifically at the behest of scientists supporting the calibration and maintenance of the VLA and EVLA. These include a task to read the "telcal" text file to prepare an initial solution (SN) table, a task to solve for antenna position corrections from appropriate solutions in an SN table, a task to fit corrections to the antenna pointing model from a file produced by the on-line system in pointing mode, and several tasks to process holography data up to and including a map of the corrections to be applied to the antenna panels. This appendix is an attempt to document such tasks. Note that TLCAL is a task which may be of interest to general VLA users. The antenna location task LOCIT and the holography tasks may well be of interest to scientists maintaining interferometers other than the VLA. However, the pointing task PEEK will probably be of use only to people tasked with maintaining the VLA.

V.1 Real-time calibration tables: TLCAL

The VLA on-line system determines antenna-based complex gain and delay values for the average of every calibrator scan. These are written into a text file which is stored at the VLA site on a computer named mchammer. The actual file name is

/home/mchammer/evladata/telcal/*year*/*month*/*asdmfile*.GN

where *year* is the four-digit year, *month* is the two-digit month, and asdmfile is the long complicated name assigned to the particular scheduling block. It is possible to read such files directly from computers in Socorro and task TLCAL supports such access. The text file may also be copied by, for example, the analysts and made available to users in Socorro and elsewhere. The gains in the telcal file all assume that every source is 1.0 Jy in all spectral windows. If you have run SETJY on your data set before using TLCAL, then you can adjust the gains of the known flux calibrators using the fluxes stored in the source (SU) table. In this case, the SN table produced will be ready for use with GETJY.

To run TLCAL on a file in your data area, enter

> TASK 'TLCAL' ↵

> INDI *n* ; GETN *m* ↵ — to specify the data set.

> DOAPPLY 1 ↵ — to use the fluxes in the source table.

> ASDMFILE(1) = 'MYAREA: ↵ — two parts are concatenated.

> ASDMFILE(2) = 'TSUB0001.sb23643640.eb23666175.56454.14034020834 ↵

> INP ↵ — to check that you remembered to leave off the close quotes above.

> GO ↵ — to run the task.

Substitute your file name parameters for those used above and leave off the .GN part of the name and the close quotes (to preserve your desired case). It would be wise to check the SN table produced with, *e.g.*, SNPLT . Bad RFI may render some telcal values bogus.

The Telcal file is text file with very long lines beginning with 3 header lines. Then there are columns for MJD (to high accuracy), UTC (sexagesimal), LST (days), LST (sexagesimal), IFID (5-character string), Sky frequency (MHz), Antenna ID (*e.g.*, ea03), gain amplitude, phase(degrees), residual, delay (ns), flagged (true/false), zeroed (true/false). hour angle (radians), azimuth (radians), elevation (radians), source name, and flag reason. TLCAL uses the MJD, antenna, amplitude, phase, delay, flagged, and source columns. A combination of the IFID (peculiar to the EVLA) and the sky frequency is used to identify the IFs to which these apply.

V.2 Correcting antenna positions: LOCIT

The data taken for pointing analysis after antennas have been moved is now also used to determine corrections to the assumed antenna locations. The data are loaded into *AIPS* using BDF2AIPS in the usual way, but DOKEEP = 1 is required in order to load pointing data. Then FRING is run on one scan from a strong source with SOLINT = 5 minutes and with no rates (DPARM(9) = 1). Then CLCAL is used to apply the SN table to all times in the CL table. Then the data set is reduced in size by running SPLAT, averaging in frequency and time:

> DEFAULT SPLAT ↵	to initialize the adverbs
> INDISK *n* ; GETN *m* ↵	to specify the data set.
> OUTDISK INDISK ↵	to keep on same disk.
> APARM 3, 1, 0, 0, 1 ↵	to average data in frequency, but not over IF. The input data are taken at 1-second interval.
> DOCAL 1 ; GAINU 0 ↵	to apply the calibration.
> CHANNEL 64 ; CHINC 0 ↵	to average all spectral channels together.
> SOLINT 5/60 ↵	to average over time to 5-second intervals.
> INP ↵	to check the inputs.
> GO ↵	to make a rather smaller data set.
> INDISK *n* ; GETN *j* ↵	to specify the averaged data set.
> INEXT 'NX' ; INVERS 0 ; EXTDEST ↵	to delete the index table which has each direction of the pointing as a separate scan.
> DEFAULT 'INDXR'	to initialize everything.
> INDISK *n* ; GETN *j* ↵	to specify the averaged data set.
> CPARM 0 , 10 ↵	to set maximum scan length to 10 minutes.
> GO ↵	to make the new index table.

It might be wise to make a backup copy of the data set at this point in case later machinations go wrong. Use UVCOP.

A solution table is now needed containing the phases of every scan in the data set. Use

> DEFAULT CALIB ↵	to initialize the task and adverbs.
> INDISK *n* ; GETN *j* ↵	to specify the averaged data set.
> SOLINT 10 ↵	to average full scans.
> APARM 3, 0 ↵	to require at least 3 antennas, but do *not* average polarizations or IFs.
> SOLTYPE 'L1R' ; SOLMODE 'P' ↵	to solve for phases only with recursive L1 methods.
> DOCALIB 1 ; GAINUS 0 ↵	to apply calibration, a no-op at this stage.
> SNVER 0 ; REFANT *nn* ↵	to create a new solution table and to select a reference antenna (*nn*)
> INP ↵	to double check.
> GO ↵	to make the solution table needed next.

Finally, fit the antenna positions. It is best to do each IF separately and also try the mode that uses the phase difference between IFs.

> DEFAULT LOCIT ↵	to initialize things including DOTV false, and REFANT 0.
> INDISK *n* ; GETN *j* ↵	to specify the averaged data set.
> BIF 1 ; EIF BIF ↵	to examine one IF at a time.
> STOKES 'RR' ↵	to examine one polarization at a time.

> DPARM 0, 1, 10, 120 ↵	to use each scan by itself (rather than differencing consecutive scans), do not fit the K term, and limit the fit to elevations between 10 and 120 degrees. All hour angles and declinations are used. The total phase is used.
> BPARM 3, 0, -180, 180, 3, 4, 1, 3 ↵	to plot 3 antennas per page between -180 and 180 degrees as boxes with connecting lines, and, if doing more than 1 IF or polarization, plot them in the same plot.
> DOOUTPUT -1 ↵	to ignore any values contained in pre-existing files named OUTPRINT.*arrayname*. If this file exists before running LOCIT and DOOUTPUT > 0, it will be read and the corrections contained within it added to those found before they are written to a new OUTPRINT.*arrayname* file and also to the other output files. See HELP LOCIT ↵ for details.
> OUTPRINT 'FITS:*date*' ↵	to set the output file base *name* used for output text files.
> INP ↵	to check that all is well.
> GO ↵	to run the task.

Then do

> STOKES 'LL' ; GO ↵	to redo with the other Stokes.
> BIF 2; EIF BIF ↵	to redo with the second IF.
> STOKES 'RR' ; GO ↵	to redo with the other Stokes.

Now compare the answers and uncertainties found in each of the four runs. Also it may be good to do

> BIF 1; EIF 2 ↵	to include both IFs.
> STOKES 'RR' ↵	to do one Stokes at a time.
> BPARM(9) 1 ↵	to select the mode which uses the phase difference between the 2 IFs.
> GO ↵	to get a fifth estimate of the antenna position corrections.

Repeat for STOKES 'LL'.

Now comes the hard part. LOCIT will have created a great many plot files, an example of which is shown in Figure V.1. It also writes text files named OUTPRINT with extensions:

FIT	This file contains all solutions with their uncertainties in a form that is easy to read. It may be used to accumulate corrections depending on the setting of DOOUTPUT.
EVLA	This file contains only significant corrections in a form that can be hand edited to put in the dates when the antennas move and when the database is fixed. The edited result may be edited to a system file on the web to enable VLANT to apply the appropriate corrections.
PAR	This file, containing only significant corrections, is read into Parminator to update the database.
001	This file contains only significant corrections in a script form to read into CLCOR. Do this, to make a new CL table and then rerun CALIB to see if the resulting phases really end up near zero.

You will need to review the plots (TVPL is helpful) and study the .FIT file to decide which antennas have significant corrections that need to be made. You will almost certainly need to edit the .001 file and then apply it to your averaged data set. Repeat CALIB and then examine the resulting SN table with EDITA or SNPLT. If things worked well, then edit the .PAR and .EVLA files to match.

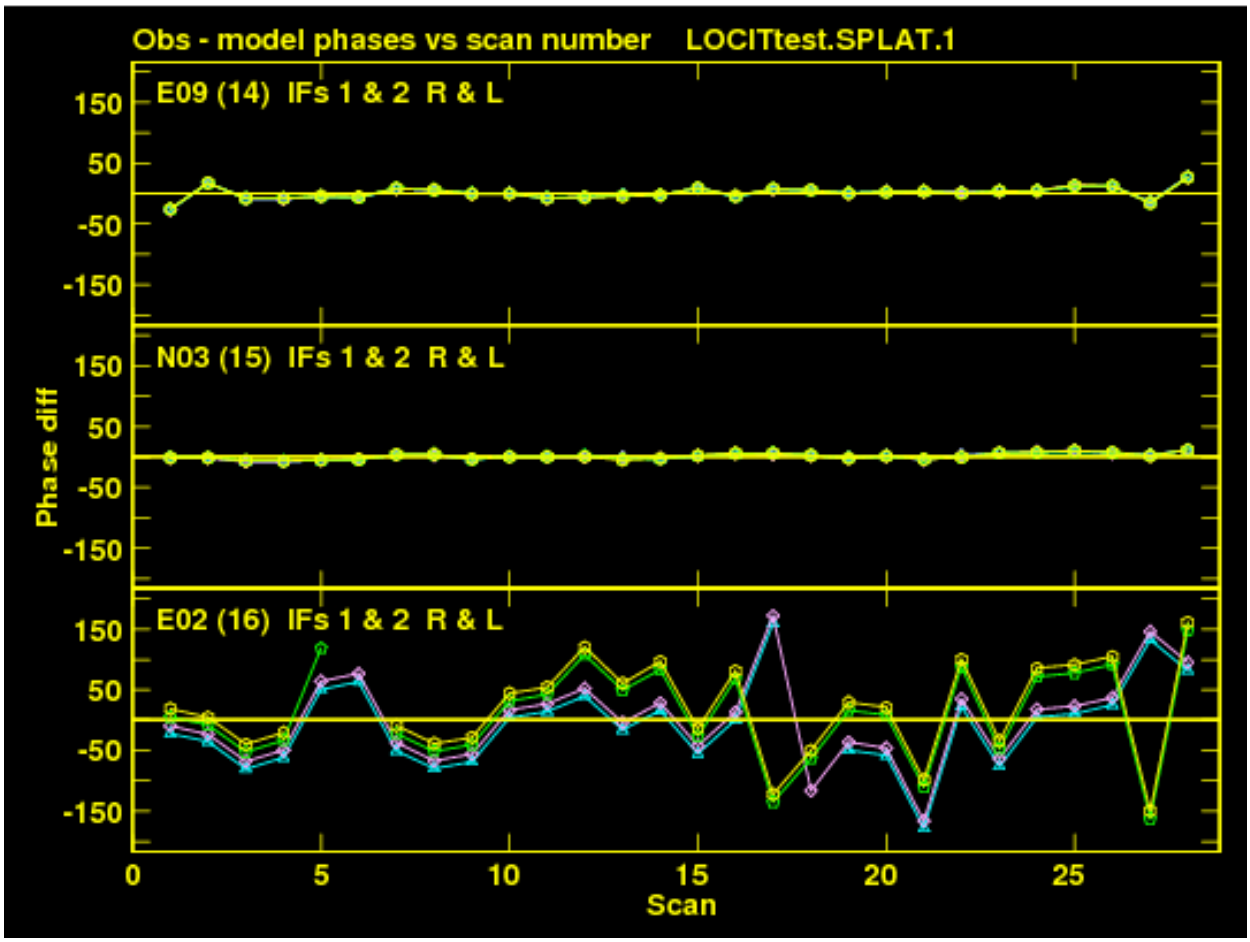


Figure V.1: Example of the residual phases after a LOCIT solution is applied to the input SN-table phases. The solutions for antennas 14 and 15 are fine, but antenna 16 did not converge to a correct solution. The data from both IFs and both polarizations were used in this run of LOCIT and are represented by different colors in the plot.

V.3 Improving antenna pointing: PEEK

When the EVLA on-line system conducts a pointing observation, be it for a long series of such observations after antenna moves or a single observation to be applied to the next target source, the software updates a text file with extension .PNT. These files may be copied from the computers at the VLA and then analyzed to determine increments to the parameters of the pointing model. This analysis was done with a program called peek available from Ken Sowinski's disk area. That program has been adapted to run as an *AIPS* task. The method of setting the input parameters was changed to be the familiar *AIPS* methods, the program was given the option to make *AIPS*-style plots in addition to the gnuplot files, and has the option to suppress some of the output text files.

An example usage would be

> DEFAULT PEEK ↵	to initialize the task name and all adverbs.
> INDISK n ; GETN m ↵	to specify an arbitrary catalog entry to which plot files are attached. It may be omitted if DOPLOT≤ 0 or DOTV = 1.
> ASDMFILE(1) = 'MYAREA:peek/ ↵	two parts are concatenated.
> ASDMFILE(2) = 'PX8262 ↵	file renamed to useful name size
> DOPLOT 1 ; DOTV 1 ↵	to make *AIPS* plots on the TV display.
> INP ↵	to check the inputs.
> GO ↵	to run the task.

Do not include the .PNT extension in ASDMFILE; the task will add it. Adverb APARM is used to set data and output selection parameters. The defaults are to include all dates, to include elevations between 7 and 122 degrees, to include all samples with amplitude > 0.01, to omit samples with wind speed > 5 m/s, to include normalized beamwidths between 0.95 and 1.15, and to write the filtered change file only when the change is > 0.1 arc minutes and > 3 times the uncertainty.

Adverb FPARM is used to determine which parameters the task determines, with the default being the usual set (Tilt, Azzero, Elzero, Colima, and Refrac). FPARM can be, but is normally not, used to attach the tilt to the antenna rather than the pad and to suppress the writing of various output text files. These text files are written to the disk area which is the "root" directory of the ASDMFILE adverb — MYAREA:peek/ in the example above. The output files are named with ASDMFILE and extensions .PTR (large overall summary), .gp (gnuplot plotting instructions), .changes (all of the adjustments found), and .filtchg (those adjustments that are deemed significant). In addition, files are written named antnn.dat that include all relevant parameters for all antennas which have been fit and which are used by gnuplot.

You can use the .gp file by typing gnuplot followed by the name of the .gp file. It will make a file named plot.ps. Adverb DOPLOT may be used to request *AIPS*-style plots with substantially better control, using DPARM, over symbol types, sizes, and colors. In both styles of plotting, six plots appear on each plot page showing measured and residual azimuth and elevation corrections plotted against azimuth, elevation, and relative time. You may plot these on the TV or make plot files, one for each antenna fit. If using the TV, button D tells the task to stop plotting but continue doing all the fits and text file outputs. A new procedure to set colors for LWPLA has been written called OKCOLORS. It sets the colors to useful values for plotting on a white background. Black backgrounds use a lot of printer ink and so are less desirable when printing many plots.

V.4 Improving antenna surfaces: holography

In "holography" data from the EVLA, reference antennas are pointed towards the nominal source position, as normal. All other antennas are pointed to an offset position controlled by the observer. The angular offset from the nominal position, for each antenna, is written to data sets in the same variables that are normally used for u, v, w. Normally, when UVCOP is used to select IFs, the values of u, v, w are scaled to the new header frequency. To prevent this, set UVCOPPRM(8) = 1 and, if any holography sources have names other than HOLORASTER, also set SRCALIAS.

In holography mode, only data on a baseline between a reference antenna and an offset (non-reference) antenna are normally used. (Data between two moving antennas may be plotted by UVHOL however.) The correlations observed on these baselines are stored in the multi-source file in the normal way. Data are calibrated as usual and then extracted with UVHOL for holographic analysis. This analysis includes determination of the shapes of the primary beams with PBEAM, Fourier analysis with HOLOG to make images of the dish surface including the deviations from the desired shape, and finally PANEL to prepare detailed prescriptions for adjusting the antenna panels. Verb TVLAYOUT plots an antenna panel layout on top of a TV holography image and has the layouts for the VLA and VLBA built in.

Holography data are taken by driving the moving antennas to a desired offset and then dwelling for a period of time, accumulating some number of samples. Then the moving antennas are driven to the next desired offset, usually horizontally or vertically, and the process is repeated. Holography samples are recorded with the u random parameter as the azimuth offset direction cosine, the v random parameter as the elevation direction cosine, and the w random parameter as a counter of the dwell interval. In the old VLA, the w had an additional 40000.0 added as a holography indicator. With the EVLA, the source name is the sole method of discriminating between regular visibility data taken for calibration and the holography data.

Beginning in late 2022 and supported only in 31DEC23 (and beyond), a new "on the fly" mode of holography observation has been introduced. In that mode, the moving antennas move continuously at the user-specified rate. This is much faster since there is not need to blank data while the telescopes start up and then settle at the new position. Of course, the data integration time at each offset position is considerably reduced. UVHOL and HOLOG have the new adverb OTFMODE to tell the tasks that the data were taken in this mode and to tell HOLOG to average the data in time if necessary. In this mode, the w parameter does not appear to have any reliable meaning, and UVHOL treats each new time as a new position. Otherwise, these tasks function as they always have.

In 31DEC23, the task HOLIS lists the groups of holography data showing which scans contain which range of coordinates. It tests for data samples out of order in OTF mode calling them "Backwards." It calls each scan at constant L or M a group in OTF mode and each dwell position a group in non-OTF mode. It can print a very detailed account of the latter or a summary. This listing should allow the user to find particular coordinates for plotting in UVHOL.

UVHOL has three main modes of operation: printing the data much like UVPRT, averaging and writing the data to text files for analysis by HOLOG, and plotting the data in *AIPS* plot files or on the TV display. UVHOL has all of the usual data selection and calibration adverbs. In addition it has a number of basic holography-specific adverbs. ANTENNAS is used to list the reference antennas and BASELINE is used to list the moving antennas. Since it takes some time for the antennas to settle down after each move, adverbs NPOINTS or APARM(1) through APARM(3) control which samples within each pointing are retained. APARM also controls whether the data are averaged over all moving antennas and/or all reference antennas and/or over time. Normally, holography data should not have the parallactic angle correction applied on DOPOL, but it can be done if desired. DPARM is used to advise the print routines on scaling of data and weights and also to invoke special plotting options (use db for amplitudes, plot phase or imaginary rather than amplitude/real). There are a number of the usual plot options, such as SYMBOL, DO3COL, LTYPE, and the like. DPARM(8)=1 has the

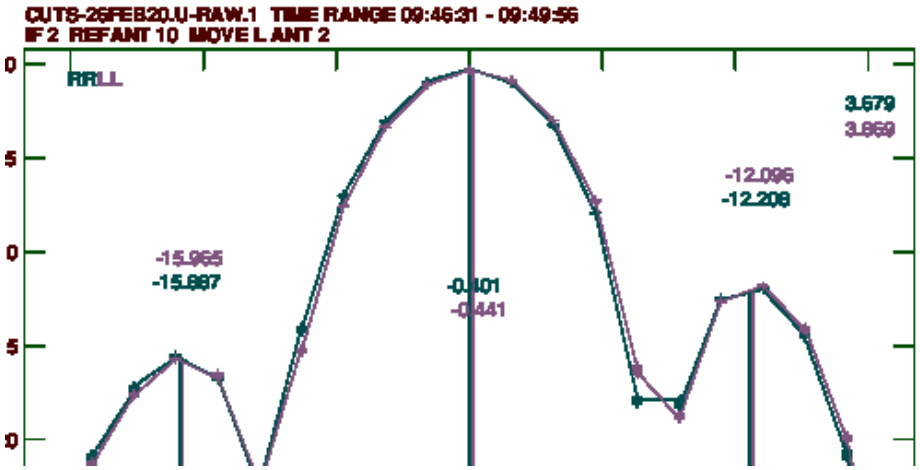


Figure V.2: Example of the amplitudes measured in a single holography scan as plotted by UVHOL. Antenna 2 moved in the L coordinate while antenna 10 remained pointed directly at the source. Both polarizations are plotted with multiple samples at each pointing (not averaged). The fit maxima are plotted along with their values in db and the sidelobe ratio in db.

task fit the main and sidelobe peaks and display the values on the plot. DOCRT controls where the print out is sent, either to the terminal or to OUTPRINT.

On OPTYPE='HOLG' mode, the holography data are written to text files, a number of files for each pair of antennas. Each line of these files, after a number of header lines, contains l, m, amplitude or real, phase or imaginary, error in amplitude or real, and error in phase or imaginary. In this mode, OUTPRINT specifies the logical directory name of the output files which are then named with an additional HOLO*nn-mmppii* where *nn* is the reference antenna (BASELINE(*j*) or 0 if multiple), *mm* is the antenna (or 0 if multiple), *pp* are 2 letters giving the polarization, and *ii* is the IF. These files are then given to HOLOG.

An example of plotting one short raster at 4.8 GHz has inputs

> DEFAULT UVHOL ↵	to initialize the task name and all adverbs.
> INDISK *n* ; GETN *m* ↵	to specify the holography data set.
> SOURCES 'HOLORASTER' ' ↵	to specify the standard holography source name. Required with EVLA data. Note the extra quote mark at the end which makes all the other values of SOURCES blank.
> STOKES 'HALF" ↵	to plot RR and LL.
> ANTENNAS 9,0 ↵	to specify one reference antenna.
> BASELINE 8,0 ↵	to specify one moving antenna.
> ICHANSEL 5, 60 ↵	to omit outer channels from the average.
> DOCALIB 1 ; DOBAND 1 ↵	to apply the calibration.

> APARM 1, 3, 2 C_R	to include all but the first 3 samples and last 2 in each pointing with no averaging.
> DPARM 0 0 0 0 0 1 0 1 C_R	to plot amplitudes in db and to find the center and sidelobe maxima and display and plot their locations, values, and ratio in db.
> DO3COLOR 1 C_R	to plot the two polarizations in different colors.
> OPTYPE 'PLOT' C_R	to make plots rather than output text.
> INP C_R	to check the inputs.
> GO C_R	to make the plot file shown in Figure V.2.

Now, to continue the holography analysis, we write out the data in appropriate text files. An example, would be

> TGET UVHOL C_R	to get the previous parameters for UVHOL.
> OPTYPE 'HOLG' C_R	to set the holography mode.
> OUTPRINT 'MYAREA:' C_R	to set the logical directory for the output text files.
> ANTENNAS 1, 5, 9, 17 C_R	to specify more of the reference antennas.
> APARM 1, 3, 2, 0, 1, 1 C_R	to include all but the first 3 samples and last 2 in each pointing averaging over time within the pointing and over all reference antennas.
> INP C_R	to double check the inputs.
> GO C_R	to write out the data for HOLOG.

This will write a file named MYAREA:HOLO00-08RR01 and one named MYAREA:HOLO00-08LL01. Note that we did not specify the one IF to be used, so IF number one was what we got.

HOLOG reads the visibility data from the text file produced by UVHOL. It can read either amplitude and phase or real and imaginary (APARM(7) in UVHOL). It then Fourier transforms the data to produce the grading in the aperture plane of the antenna. Images of the re-gridded observed amplitude and phase (A_AMP, A_PHA), of the weights in the re-gridding (WGT), of the derived amplitude and phase of the illumination in the antenna aperture (V_AMP, V_PHA), of the amplitude and phase of the point-spread function (P_AMP, P_PHA), of the phase corrections removed by the focus model (MODEL), of the surface deviations of the antenna (V_DEV) in meters, and of the interpolated antenna power pattern (A_PWR) may be produced under control of adverb DPARM. Task PANEL requires V_AMP (DPARM(4)) for the mask and V_DEV (DPARM(9) for the deviations. An example of the images produced by HOLOG is shown in Figure V.3.

Many of the control adverbs must be set to appropriate values. A sample set of inputs is

> DEFAULT HOLOG C_R	to initialize everything.
> INFILE 'MYAREA:HOLO00-28RR01 C_R	to point at the text file, 28 is the moving antenna with several reference antennas.
> OUTNAME '28-Q2-40935E' C_R	to name the output images.
> OPTYPE 'SUBR' C_R	to specify the type of antenna, sub-reflector, sub-reflector with reference pointing, or primary focus.
> FACTOR 13 C_R	to set magnification factor for sub-reflector antennas; default 13.0.
> OFFSET 0.512 C_R	to set the distance from the prime focus to the bottom of the subreflector in meters; 0 -¿ 0.512.
> APARM 0, -1, 25, 3.4, 9 C_R	to set antenna outer diameter, inner diameter, and focal length in meters.
> BPARM 30, 256, 0, 1, 1, -2, -2.5, -12, 8100, -1 C_R	to set data reduction parameters.
> DPARM 0 0 0 1 0 1 0 1 1 0 C_R	to request V_AMP, P_AMP , MODEL, and V_DEV images.

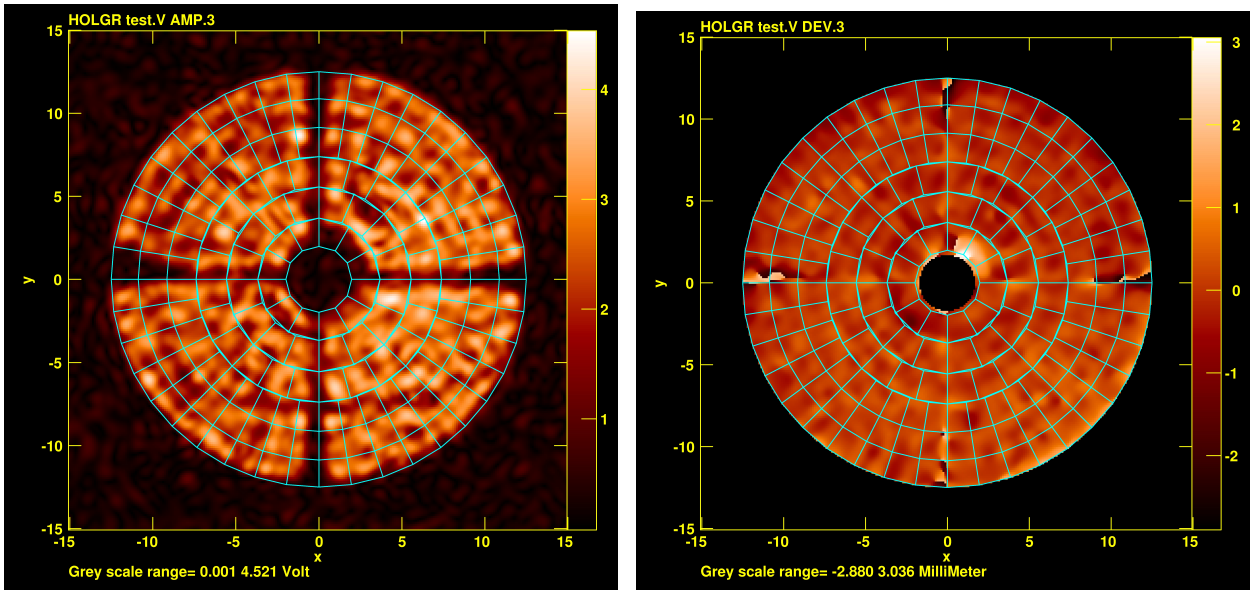


Figure V.3: Example of the mask (V_AMP) and deviation (V_DEV) images produced by HOLOG. Task GREYS can do both the pseudo-coloring with OFMFILE = 'RYPHLAME' and the panel layout with INFILE = 'VLA'.

> INP ↵ to double check everything.
> GO ↵ to make the images.

BPARM specifies the required image size in meters, the image size in pixels, minimum and maximum antenna scan angle, amplitude scaling factor, Fourier transform control (-2 says use DFT with VLA sign convention), minimum and maximum coordinate used in correcting for pointing, focus, and feed offset (negative says in the radial coordinate), a control parameter (100 inhibits local phase ambiguity correction in the V_PHA plane, 8000 turns on finding the Cassegrain offset), and type of data (-1 for VLA). The image size in meters is a function of frequency. To avoid this, CELLSIZE may be set to override BPARM(1) when analyzing multiple frequencies. CPARM all zero says to use the default spheroidal function in gridding on both L and M axes; highly recommended, but not relevant when using DFT imaging. Note that HOLOG can also generate images from a feed model using VPARM.

In 31DEC23, an experimental task HOCLN was written to do a Clean operation on the images produced by HOLOG. Normally, data sampling in holography is essentially on a regular grid at or less than the critical sampling. As a result, this complex Clean operation usually is of little value.

The last task in the process of measuring corrections to the antenna surface is PANEL

> DEFAULT PANEL ↵ to initialize everything.
> INDISK *n* ; GETN *m* ↵ to specify the V_DEV image.
> IN2DI *i* ; GET2N *j* ↵ to specify the V_AMP image used as a mask.
> OUTFILE 'MYAREA:PANEL.out ↵ to specify the text file to receive the panel adjustment instructions.
> DOPLOT 7 ↵ to request plots of the input deviations, the panel adjustments, and the residuals.
> APARM 2, 1, 0, 0, 42 ↵ To set the panel model, the clip level in the mask image, and the number of pointings. The map size and frequency should be available from the image header.
> INLIST ' ' ↵ to use the default VLA panel layout.
> INP ↵ to review the inputs.

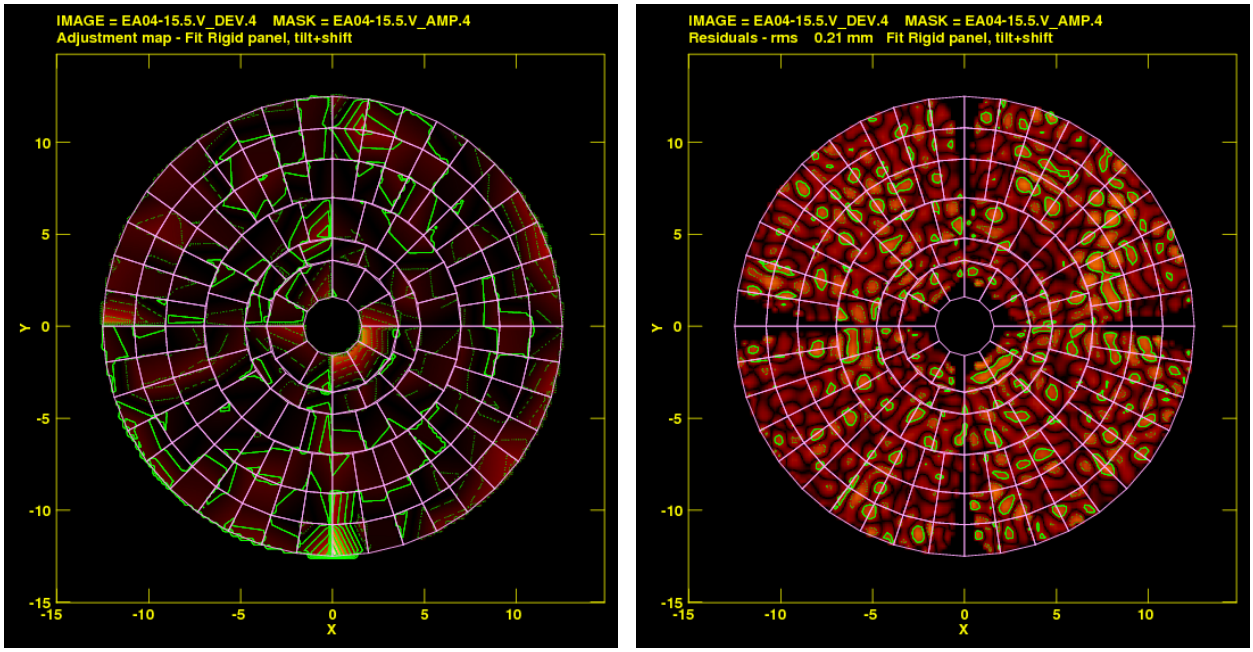


Figure V.4: Example of the panel adjustment image and the residual image computed and plotted by PANEL.

> GO C_R to fit the data, write the adjustment instructions, and make plots.

The corrections are determined by fitting all samples in each panel to a model of a rigid panel with only shift or with tilt and shift or a flexible panel. There are a large number of samples in each panel, so in 31DEC23, an option to omit samples near the edges of panels was added. This APARM(6) option allows the user to suppress effects spilling over from adjacent panels. Example plots of the panel adjustments and residual are shown in Figure V.4. Adding 8 to DOPLOT produces an image of the dish with all panels marked to show the corner definitions which rotate around the dish.

With appropriate holography data, the output of UVHOL may also be used to fit an analytic function to the primary beam. The task that does this is called PBEAM and it fits a polynomial function and makes contour plots of the input data, the fit model, the residual, and IRING-like plots comparing data and model in two colors, and then IRING-like plots of the residual image and residual image divided by the model. Plots against radius of the smooth model with the data points and of the residuals may also be generated. In 31DEC25, QBEAM can construct and plot similar images without the beam fitting which may be useful for polarization data.

Sample inputs could contain

> DEFAULT PBEAM C_R to initialize everything.

> OUTNAME 'PBEAM Cband C_R to make output images of the data, model, and residual. Required if DOTV is false and DOPLOT is > 0.

> INFILE 'MYWORK:C-6052I_1' C_R to provide the text file from UVHOL.

> OUTFILE 'MYWORK:CbandTxt C_R to save the messages including answers to a text file.

> VPARM 0 C_R to use all defaults in the fitting.

> DOPLOT 15 C_R to get all possible plots.

> OPTYPE 'ELLI' C_R to fit eccentricity and position angle.

> INP C_R to double check everything.

> GO C_R to do the fit and write out images and plots.

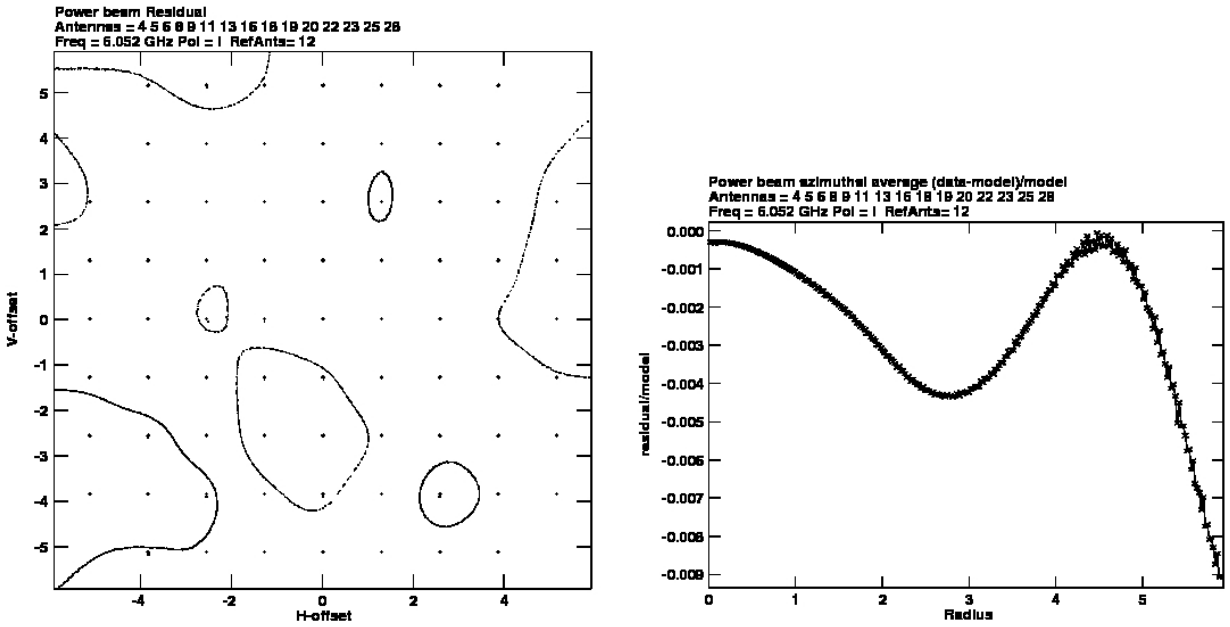


Figure V.5: Example of the plots from PBEAM: contour plot of the residual image and IRING-like plot of the residual image divided by the model.

Sample plots are shown in Figure V.5.

V.5 Antenna polarization: PDPLT

The basic calibration task PCAL will write channel-dependent polarization D-terms into a PD table (§ 4.3.16). POSSM can plot PD tables, one antenna at a time while BPLOT can plot all antennas on one plot, separated vertically. In 31DEC20 a new task PDLT was written to plot up to four PD tables on top of each other. This allows comparison of an antenna's D-term spectrum over time, *i.e.*, in multiple data sets with different antenna stations. Antennas may also be compared as in the example below.

To run DLT on one file to compare different antennas,

> DEFAULT 'PDPLT' C_R	
> INDI *n* ; GETN *m* C_R	to specify the data set.
> IN2DI *n* ; GET2N *m* C_R	to specify the same data set.
> IN3DI *n3* ; GET3N *m3* C_R	to specify a different data set.
> INVERS 3 C_R	to set the PD version to use with the first data set.
> IN2VERS 1 C_R	to set the PD version to use with the second data set.
> IN3VERS 3 C_R	to set the PD version to use with the third data set.
> ANTENNAS 1, 6 C_R	to select the antennas from the first PD table to appear in the two plots.
> BPARM 2, 3 C_R	to select the antennas from the second PD table to appear in the two plots.
> CPARM 4, 5 C_R	to select the antennas from the third PD table to appear in the two plots.

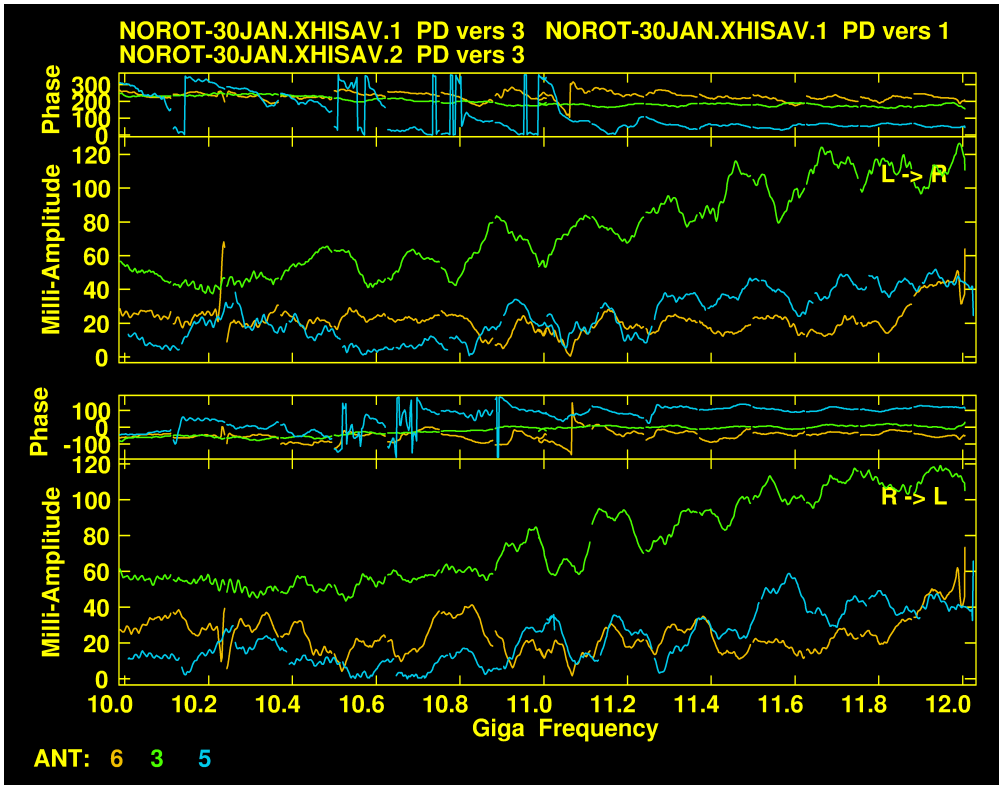


Figure V.6: The second plot from the example PDPLT.

> ECHAN 62 ↵ to omit 2 questionable channels in each IF.
> OPTYPE 'A&P' ↵ to lot both amplitude and phase.
> APARM 0 0 1 ↵ to label the x-axis with frequency.
> FACTOR -0.05 ↵ to use connected line plots only.
> DO3COLOR 1 ↵ to separate the plots with color.
> INP ↵ to check your adverb values.
> GO ↵ to run the task.

The second of the two plots made is shown in Figure V.6.

Z System-Dependent $\mathcal{AIPS}$ Tips

This appendix has not been updated in some time other than some technical corrections. The information contained herein should be indicative of conditions, but not correct in detail. This is particularly true of the Charlottesville and Green Bank information.

Although $\mathcal{AIPS}$ attempts to be system independent, some aspects of its use depend inevitably on the specific site. These vary from procedural matters (*e.g.*, assignment of workstations and location of sign-up sheets, tape drives, and workstations or other terminals) to the hardware (*e.g.*, names and numbers of workstations and tape and disk drives) to the peculiar (*e.g.*, the response of the computer to specific keys on the terminal, the presence of useful job control procedures). This appendix contains information specific to the NRAO's individual $\mathcal{AIPS}$ installations. It is intended that non-NRAO installations replace this appendix with one describing their own procedures, perhaps using this version as a template. The general description of using $\mathcal{AIPS}$ on workstations was given in Chapter 2 and will not be repeated here.

Within the NRAO, $\mathcal{AIPS}$ is installed on two main architectures — Linux PCs and Mac workstations. All our old SUN, IBM RS/6000, CONVEX C–1 and DEC VAX 11/750 and 11/780 systems have been decommissioned. Currently the fastest $\mathcal{AIPS}$ machines in the Observatory are the whichever PC was bought last.

Z.1 NRAO workstations — general information

All NRAO workstations run some version of the Unix operating system, Linux on PCs and Mac OS/X on Apple products. Unix systems are intrinsically sensitive to the difference between upper and lower case. Be sure to use the case indicated in the comments and advice given in the following notes. $\mathcal{AIPS}$ itself is case-insensitive, however; conversion of lower-case characters to upper-case occurs automatically. (Unix systems have a variety of characters for the prompt at monitor (job-control) level, and allow users to set their own as well. We will use $ as the prompt in the text below.)

Z.1.1 The "midnight" jobs

The versions of $\mathcal{AIPS}$ on all NRAO PC systems are kept up to date continually with the master versions on the Socorro Linux PC called `dave`. This is achieved by automated jobs that start running at very antisocial hours of the early morning. Any changes formally made to the TST version of $\mathcal{AIPS}$ are copied to the relevant computers and recompiled/relinked. Midnight jobs run in Charlottesville, Socorro, Green Bank, and many other sites around the world.

Z.1.2 Generating color hard copy

Z.1.2.1 Color printers

Color printers are, these days, simply printers that understand the color extensions to the PostScript language used to describe plots. The NRAO owns several Tektronix color printers, two public ones in Charlottesville (`ps1color` in the $\mathcal{AIPS}$ Caige and `ps3color` in the Library) and two at the AOC in Socorro (`nm-aoc212` on the ground floor and `nm-aoc324` upstairs in the former library). You may display your PostScript file on the printer in Socorro simply by typing

`$ lpr -Pnm-aoc324` *filename* ↵ where *filename* is the name of your file.

The paper size is 8.5×11 inches, which is the default for *AIPS* tasks TVCPS and LWPLA. To have the file printed on transparency paper use queue `aoc324c/trans`. Full control over this complex printer is available with the `multiprint` command; type `multiprint --help` ↵ for information. If you do not wish to save the plot as a disk file, you may also print it directly from within *AIPS*. The color printer is one of the printer choices when you start up AIPS, but you probably want to select a regular PostScript printer as your default printer. You can change your printer selection with the verb PRINTER; use PRINTER 999 ↵ to see what your choices are and then PRINTER *n* ↵ to choose the printer numbered *n*. *AIPS* print routines will re-direct PostScript files that actually contain color commands to the first `PS-CMYK` printer in the list, but will not re-direct ordinary print jobs to some printer other than a color printer.

Z.1.2.2 Software to copy your screen

To obtain a color hard-copy of what is on your screen, there are several software options you can choose. These include TVCPS, `xv`, and `import`. Having created a PostScript (or other format) file, you can print it on color printers at the NRAO or copy the file via e-mail, `scp`, or `ftp` to some other site.

The TVCPS task in *AIPS* will create a color Encapsulated PostScript file from whatever is displayed on the *AIPS* TV server (XAS). If you use the OUTFILE adverb, this file is saved with whatever name you specify (see § 3.10.1). If you specify a black-and-white output to TVCPS, then the output can be sent to any PostScript printer. Color PostScript must be sent to a color printer. You can, of course, edit the saved file (if you are a PostScript wizard or use HELP POSTSCRIPT) and can insert the file (since it is encapsulated) in another document.

The `xv` program is a Unix utility program available on most systems at the NRAO. It is mainly intended for image display of GIF, JPG, TIFF, and other format files. When you start `xv`, click the right button mouse anywhere in the `xv` window to bring up the control window. One of its features is a screen grab which is controlled by the "Grab" button in the lower right corner of the control window. *Before* you press this, arrange your windows and icons so that you can see exactly what it is you want to grab (*e.g.*, the XAS server). Now press the "grab" button. A window with instructions will appear. Move the cursor to the top left of the area you want to grab. Then press and hold down the center mouse button, and drag the mouse cursor until it is at the bottom right of the area you want to grab. As you do this, you will see a box pattern on the screen outlining the area selected. Once you are done selecting the area, release the mouse cursor. When `xv` has finished grabbing the screen, whatever you grabbed will appear in the main `xv` window. You can now use the "save" button of the control window to save this as any format you want. One nice feature of this is the "save as Postscript" option. It allows you to scale, rotate, and position the image in relation to the page. Its user interface is better than most image utilities.

Finally, the `import` program provides similar functionality to the "grab" feature of `xv`, with many options about output formats and much more. Enter import -help ↵ for a summary of the options. For example, enter import -quality 100 outfile.jpg ↵. The cursor will change to a plus sign. Position it at the top left corner of the area you wish to grab, hold down a mouse button, and drag the cursor to the bottom right of the area you wish to capture. When you let up on the button, the file `outfile.jpg` will be written in `jpg` format. Alternatively, position the cursor inside the window you wish to capture and click the left mouse button. The entire window will be captured. The file extension determines the format, so `.eps` will produce encapsulated PostScript.

Z.1.3 Gripe, gripe, gripe, . . .

The so-called "designated AIP" is now Eric Greisen at all times (except when he is on vacation). He assists local and remote users with their AIPS problems, providing quick advice or simple fixes to bugs, More complex problems may require some time and may even require receipt of the user's data in order to debug the problem. Contact the designated AIP (and all members of the group) at the e-mail address `daip@nrao.edu`. The "my.nrao" web portal lets you into the "helpdesk" which has an AIPS department. Management prefers if you use this approach to request assistance.

Suggestions and complaints entered on all computers with the `GRIPE` verb (see § 11.1) are sent immediately by e-mail to `daip` and thereby to all members of the group. All traffic via daip from 2000 to the present is archived at `https://listmgr.nrao.edu/pipermail/daip` and is well-known to Google. You might try Googling your AIPS error message to see if you get something useful. We stand willing, and are now able, to respond to user problems and requests on a timely basis.

The Gripes database described in AIPS Memo No. 88 used to be maintained, but appears to have disappeared.

Z.1.4 Solving problems at the NRAO

Below are details specific to NRAO systems for handling some of the problems which may arise in AIPS.

Z.1.4.1 Booting the workstations

Modern workstations, especially the powerful PCs and Macs, are complex Unix systems which may have remote users within the NRAO and guests from elsewhere on the Internet. Users should *never* attempt to boot the system on their own. If the machine appears to be dead, find or call one of the people listed on the bulletin boards in the AIPS Caige for this purpose.

Z.1.4.2 Printout fails to appear

Check the AIPS output messages that appeared shortly after you submitted your print job, whether it be from `PRTMSG` or `LWPLA`, or some other task. You should see the output of the Unix command to show the printer queue status. If anything went wrong with the print submission, an error message should be obvious. If not, check the output of the `lpq` command, see what print queue was involved, and check it again from the Unix command level (not from inside `AIPS`).

`AIPS` will delete spooled files about 5 minutes after they are submitted. If the print queue is stalled (due, say, to a jammed printer) or backed up with a lot of jobs, it is possible that the file was deleted before it was gobbled up by the print spooler. This time delay has been made a locally-controlled parameter, so it is possible to set it to values higher than 5 minutes.

Finally, check to see if the printout was (a) diverted to the "big" printer (`psnet` in room 213 at the AOC or `ps3dup` in the Charlottesville library) because it was too long for the smaller printers, (b) you forgot which printer you had selected on `aips` startup, or, at the AOC, (c) someone has taken the output and filed it in the "today" file bin (at the AOC this is on the left side of the post directly behind the `psnet` printer).

Z.1.4.3 Stopping excess printout

To find out what jobs are in the spooling queue for the relevant printer, type, at the monitor level:

`$ lpq` ↵	to list default print queue
`$ lpstat` ↵	to list default print queue under Solaris

or to display a specific queue

`$ lpq -P`*ppp* ↵	to show printer *ppp*

where *ppp might* be `psnet` at the AOC or `ps3dup` in Charlottesville. If the file is still in the queue as job number *nn*, you can type simply

`$ lprm -P`*ppp nn* ↵	to remove the job

`lprm` and `cancel` will announce the names of any files that they remove and are silent if there are no jobs in the queue which match the request.

Unfortunately, it is now very difficult to stop long print jobs. The large memories of modern printers mean that more than one print job can already be resident in the printer while your long unwanted job is being printed. Therefore, turning off the printer is not an option. Try to be more careful and not generate excess printout in the first place (save a tree). *AIPS* tasks now check the size of print jobs that will go directly to a printer and ask if you really mean to print them.

A nice option available for most *AIPS* print tasks or verbs is adverb `OUTPRINT` which allows you to divert the output to a text file. Then you can use an editor like `emacs` to examine the file in detail before printing. The Unix command `wc -l file` will count the number of lines in a text file called `file` for you; note that `-l` is the letter ell, not the number one. *AIPS* provides a "filter" program to convert plain (or Fortran) text files to PostScript for printing on PostScript printers. The command

`$ F2PS -`*nn* `<` *file* `| lpr -P`*ppp*

will print text file *file* on PostScript printer *ppp*. The parameter *nn* is the number of lines per page used inside *AIPS*; it is likely to be 97 if direct printing comes out in "portrait" form or 61 if the direct print outs come out in "landscape" form.

Z.1.4.4 `CTRL Z` problems

The last process placed in the background via `CTRL Z` can be brought back to the foreground by typing `fg` ↵ in response to the monitor level `%` or `$` (or whatever) prompt Alternatively, the user can type `jobs` ↵, which displays all background processes associated with the current login and can bring a specific process to the foreground by typing `fg %`*m* ↵, where *m* is the job number as displayed by the `jobs` command as `[`*m*`]`. For example, if a user initiated his `AIPS`*n* by typing `aips new pr=4` ↵ and:

`^Z`	`CTRL Z` typed by accident (or intentionally).
`Stopped`	`aips new` is put in the background as "stopped" and user is returned to the Unix level.
`$ jobs` ↵	to display status of background jobs.
`[1] + Stopped  aips new`	info from Unix, where `[1]` means job 1, "Stopped" is job 1's state and "aips new" is the command used to start up job 1.
`$ fg` *m* ↵	to return job *m* to the foreground.
`aips new`	appears on the screen just to tell the user to which job he is talking (*i.e.*, it does *not* re-execute `aips new`). You should now be talking to your `AIPS`*n* again.
↵	to get `AIPS`*n* > prompt.

Z.1.4.5 "File system is full" message

The message `write failed, file system is full` will appear when the search for scratch space encounters a disk or disks without enough space. This is only a problem when none of the disks available for scratch files has enough space, at which point the task will shut down. Use the BADDISK adverb to avoid disks with little available space.

Z.1.4.6 I can't use my data disk!

If at some point during your work you find you are prevented from reading or writing files on a data disk, it could be that your AIPS number does not have access to that area. If you encounter the message:

```
AIPS 2:  CATOPN: ACCESS DENIED TO DISK 8 FOR USER 1783
```

it means that user 1783 has not been given access to write (or read) on disk 8. This can be seen, in the AIPS session, by typing FREESPAC to list the mounted disks. If you see a data disk listed with an access of `Not you`, it means your AIPS number has not been enabled for that disk. If you feel that you should have access to that particular disk, see the data analysts (at the AOC) or an AIPS Manager about enabling your user number.

Z.2 AIPS at the NRAO AOC in Socorro

A small visitor facility for AIPS users is available in both Charlottesville and Green Bank. However, most non-NRAO people wishing to reduce data at an NRAO facility with AIPS should consider coming to Socorro. Not only is the green chili good, but the user facilities are quite substantial.

Public workstations at the AOC are Linux computers, many being two processor, 8 core machines with 24 Gbyte of RAM and a lot (4 Terabytes) of disk. A complete list of the public-use workstations can be found on the web at `https://www.aoc.nrao.edu/computing/workstations.shtml`.

Of the several printers available at the AOC, most AIPS users will want to use the high-volume PostScript printer `aoc213` and the Tektronix color printer `aoc213c`, both of which are located in Room 213.

You may also use the cluster computers and Lustre disk farm at the AOC remotely. Information about reserving such usage may be found at `httpss://info.nrao.edu/computing/guide/cluster-processing`. You may need to use your `my.nrao` login to submit a helpdesk ticket requesting this usage.

Z.2.1 Reserving public-use workstations at the AOC

If you are visiting the AOC for observing or to reduce data then you should fill out a visitor reservation form at least two weeks before you expect to arrive. These forms can be found on the web at `https://www.aoc.nrao.edu/`, select "Official Visitors", then "regular reservation form", log in with your `my.nrao` account, and follow instructions from there. If you check "OBSERVING AND/OR DATA REDUCTION" as your purpose of visit at the end of Section A of the reservation form, then you will automatically be assigned an observer login account and a workstation. If you have any special data reduction needs, you should make a note of them in the comments section of the form.

If you are unable to fill out this form using the web or if you have any questions about reservations, you should contact AOC reservations at 575–835–7357 or `nmreserv nrao.edu`. Note that reservations for VLBA

projects will not be accepted unless the observer has been notified that correlation is complete and that the data have been scrutinized. If you have any special data reduction needs then you should make a note of them in the Section C of the form.

If you are an AOC staff member and wish to sign up for a public-use workstation then you should contact `helpdesk@nrao.edu` or visit real people in Room 262. You are expected to use your own login account on the public workstations. In general, visitor's reservations take priority over reservations for AOC staff.

Z.2.2 Using Linux workstations at the AOC — introduction

In order to start *AIPS* you need to log in to your assigned workstation and open a terminal window. At the AOC, we generally use the windowing environments supplied by the operating system vendors (RedHat Linux) rather than imposing a lowest-common-denominator "standard" environment. This means that the procedures for logging on and manipulating windows depend on the current version of Linux, which will progress with time.

On sitting down at a workstation you will be presented with a log-in window that has a space for you to type your user name. Enter your assigned username and press return and then, when prompted, enter your password and press return again.

You will be presented with a rather vanilla X-Windows screen. Figure out how to start a terminal window of some sort, `xterm`s are preferred.

Z.2.3 Starting *AIPS*

Once you have logged in to your workstation and have a terminal emulator window open, you may start *AIPS*İf you wish to run *AIPS* on the machine at which you are sitting we recommend that you start *AIPS* with the following command.

```
aips pr=3
```

This will bypass the printer menu and select the high-volume PostScript printer `aoc213/dup` in Room 213 as the default *AIPS* printer.

Note that *AIPS* is configured to start a separate message window at the AOC. If you do not want the separate message window to appear, issue one of the following commands before starting *AIPS*.

```
export AIPS_MSG_EMULATOR=none
```

If you are using the KornShell or bash or.

```
setenv AIPS_MSG_EMULATOR none
```

if you are using csh or tcsh. You can move the cursor into the message window and hit `CTRL-Z` to cause it to disappear instead.

You may replace `xterm` with the name of another terminal emulator (e.g. `dtterm`) if you wish and you can also turn off the Tektronix emulator using one of the following commands.

```
export AIPS_TEK_EMULATOR=none
```

If you are using the KornShell or bash,

```
setenv AIPS_TEK_EMULATOR none
```

if you are using csh or tcsh. This emulator is easily ignored.

If the *AIPS* TV is already running, you may use the following to avoid the messages that are produced as *AIPS* figures out whether the TV is already running.

```
aips tst pr=4 tvok
```

Z.2.3.1 Starting AIPS on another machine

If you are going to run AIPS on a remote machine and use your local workstation as a display, we recommend that you start AIPS on your local workstation first and allow it to start the TV. You can then iconize the terminal that you are using to run AIPS and open a new terminal window. You should then use the `slogin` command to log in to the remote machine. You should then start AIPS using the following command.

```
aips tst pr=4 tv=mydisplay tvok
```

You should replace `mydisplay` with the name of the workstation at which you are sitting.

Z.2.3.2 On-line FILLM

On-line FILLM applied to the old VLA and does not apply now.

Z.3 AIPS at the NRAO in Charlottesville and Green Bank

AIPS is installed in Charlottesville based on the Linux computer named `valen`. Similarly, it is installed in Green Bank on a computer named `bratac`. The installations are current, with Midnight Jobs updating the development version each night. However, there are no current members of the AIPS group at either site, so the actual state of AIPS usage, particularly for visitors, is hard to determine from Socorro. Both sites have visitor procedures and welcome outside visitors. However, Charlottesville is concerned with ALMA users wishing to use `CASA` and Green Bank is concerned with GBT users needing to use `GBT-IDL`. Visitor information for both sites can be found from the main NRAO web site, including forms to register a visit and much information. Visitors to Charlottesville and Green Bank should inquire locally for information concerning public workstations supporting AIPS.

Z.4 Additional recipes

Z.4.1 Going bananas with bananas

1. Garnish a baked ham or ham steak with bananas.
2. Make a quick, rich desert with bananas and cream.
3. Bananas are perfect for lunch boxes. They come in their own wrapper, are easy to eat and mess-less.
4. Slice a banana in half lengthwise, brush with melted butter and bake it until tender; serve it as a "vegetable" with roasted meats or fish. Very Caribbean.
5. Don't forget old favorites like bananas sliced over cereal, diced in pancake batter, or buried midst the ice cream in a banana split.
6. Slice and stir-fry bananas with carrots, tomatoes and ground beef for a super-quick main dish.

Z.4.2 Banana-chocolate tea bread

1. Cream 1/2 cup softened **butter**, gradually add 1 cup **sugar**, beating until light and fluffy. Add 2 **eggs**, one at a time, beating well after each addition.
2. Combine 1 1/2 cups all-purpose **flour**, 2 tablespoons **cocoa**, 1 teaspoon **baking soda**, 1 teaspoon **salt**, and 1/2 teaspoon `cinnamon`; sift together.
3. Stir flour mixture into egg mixture, blending well.
4. Add 1 teaspoon **vanilla extract**; stir in 1 cup mashed **banana**, 1/2 cup **sour cream**, 1/2 cup chopped **walnuts**, and 1/3 cup miniature **semi-sweet chocolate chips**.
5. Spoon batter into two greased and floured 7-1/2 x 3 x 2-inch loaf pans. Bake at 350° F for 55 minutes or until a wooden pick inserted in center comes out clean. Cool in pans 10 minutes, remove from pans and cool completely on a wire rack.

Thanks to Tim D. Culey, Baton Rouge, La. (`tsculey@bigfoot.com`).

Z.4.3 Banana caramel pie

1. Mix 1 cup **dark brown sugar**, 1/4 cup **all-purpose flour**, 1/4 teaspoon **salt** in a saucepan. Stir in 1/4 cup **cold water** and 2 **egg yolks**. Beat until smooth.
2. Gradually stir in 1 cup **boiling water**. Then cook, stirring constantly, about 3 minutes until smoothly thickened.
3. Stir in 1 tablespoon **butter**, 1/2 teaspoon **vanilla**, and 1/4 cup **evaporated milk**. Cool slightly.
4. Pour into pre-baked, cooled 8 or 9 inch **pastry pie shell**.
5. Slightly before serving, slice 4 ripe **bananas** and arrange over filling. Top with whipped cream or with a meringue made out of the 2 left-over egg whites.

Z.4.4 Curried shrimp

1. Cook 2 1/2 pounds **shrimp** for 3 minutes. Peel and devein.
2. Heat 1/3 pound **butter** or margarine in large saucepan. Saute 4 chopped **scallions** and 2 cups chopped, peeled **apples** until tender. Stir in 2 tablespoons **curry powder**, 1 tablespoon **ground ginger**, and 1/3 cup **flour**. Stir for 2 minutes. Remove from heat and blend in 3 cups **chicken broth**. Return to heat, cook stirring until mixture boils and thickens.
3. Add 1 pound roasted **cashews**, 1 pound **Turkish apricots**, and, if desired, 2 ounces diced **crystallized ginger** and **raisins**. Cook over low heat for 15 minutes.
4. Add shrimp and mix in.
5. Cut 3 **bananas** into thick slices and add to mixture. Serve over cooked white or curried rice.

Thanks to Chiquita Bananas.

G Glossary

adverb — See *POPS symbols.*

AIPS monitor — a computer terminal (perhaps lacking a keyboard) whose CRT screen is used in AIPS solely for the display of information related to the progress of the execution of the AIPS tasks. (Except, at those AIPS sites without a terminal dedicated to this use, the AIPS user's interactive terminal is used for dual purposes—i.e., to serve as the AIPS monitor as well.) Many of the messages which the AIPS tasks write to the monitor also are recorded in the *message file* (*q.v.*).

aliased response — in a radio interferometer map, a spurious feature due to a source—or to a sidelobe—that lies outside of the field of view. Consider the sampling of a visibility function V at the lattice points of a rectangular grid as multiplication of V by the comb-like distribution $R(u,v) = \sum_k \sum_l \delta(u - k\Delta u, v - l\Delta v)$. The Fourier transform $\widehat{RV}$ of RV is given by the convolution $\hat{R} * \hat{V}$. Since $\hat{R}$ is again a comb-like distribution, with peaks, or teeth, separated by $\frac{1}{\Delta u}$ in one direction and by $\frac{1}{\Delta v}$ in the perpendicular direction, $\widehat{RV}$ is periodic, and, about the position of each tooth in the comb, it looks like an infinite summation of rectangular pieces of $\hat{V}$, each of size $\frac{1}{\Delta u} \times \frac{1}{\Delta v}$, taken from all over the plane. Aliased responses can be suppressed very effectively, by judicious choice of the *gridding convolution function* (*q.v.*).

For a more complete discussion, see Dick Sramek and Fred Schwab's Lecture No. 6 in the *Third NRAO Synthesis Imaging Summer School*. Also see VLA Scientific Memoranda Nos. 129 and 131.

aliasing — in spectral analysis, error which is due to undersampling: one may wish to sample a signal that is known to be bandlimited, but whose bandwidth may not be known a priori. The Fourier transform of *Shannon's series* is periodic; aliasing error is of the form of an overlapping, or superposition, of these "replicated" spectra. See *Nyquist sampling rate* and *aliased response*.

ALU — (Arithmetic Logic Unit) an (optional) microcomputer CPU unit within the I^{2}S TV display device which allows simple arithmetic operations, such as sums, products, and convolutions, to be performed on the data recorded in the I^{2}S image planes. At present, AIPS makes little use of the ALU, since many of its features are unique to the I^{2}S display unit. See *I^{2}S*.

antenna file — in AIPS, an *extension file*, associated with a *u-v data file*, in which a list of the interferometer antenna positions is stored.

antenna/i.f. gain — Many of the systematic errors affecting radio interferometer measurements are multiplicative in the visibility amplitude and additive in the visibility phase, and are ascribable to individual antenna elements and their associated i.f./l.o. chains. For each antenna/i.f. these sources of error may be lumped together into a complex-valued function of time, $g(t)$, called the *antenna/i.f. gain*. Then, the visibility measurement obtained on the i–j baseline at time t is given by $\tilde{V}(u_{ij}(t), v_{ij}(t)) = g_i(t)\overline{g}_j(t)V(u_{ij}(t), v_{ij}(t)) + \epsilon_{ij}(t)$, where V is the true source visibility and where the spatial frequency coordinates (u,v) have been parametrized by time. $g_i\overline{g}_j$ is the systematic "calibration error", and ϵ_{ij}, an additive error component, is assumed to be random and well-behaved. (Another type of systematic error, the *instrumental polarization* (*q.v.*), is not included in the g_k, and always must be corrected, by proper calibration, in order to interpret polarization data.)

Some of the most serious sources of error—including atmospheric attenuation, error arising from variations in the atmospheric path length, clock error, and error in the baseline determination—conform fairly well to this multiplicative model. This model relation is exploited heavily by the *self-calibration algorithm* (*q.v.*). Compare *antenna/i.f. phase*, and see *isoplanaticity assumption* and *correlator offset*.

antenna/i.f. phase — The *antenna/i.f. phase* for antenna k of an interferometer array is given by the argument (or phase) of the *antenna/i.f. gain* g_k: $\psi_k(t) = \arg g_k(t)$. Often in *self-calibration* one assumes that no amplitude errors are present and solves only for the ψ_k.

antenna residual delay — See *residual delay* and *global fringe fitting algorithm.*

antenna residual fringe rate — See *residual fringe rate* and *global fringe fitting algorithm*.

AP — See *array processor*.

AP–120B array processor — an *array processor* manufactured by Floating Point Systems, Inc., and used at a number of AIPS sites. Its floating-point word length is 38-bits. Typically it is equipped with a main data memory of 32–64 kilowords and a program source memory of 2048 words. With both a pipeline multiplier and a pipeline adder, and a memory cycle time of 167 ns., when programmed at top efficiency it can perform at an arithmetic rate of 12 million floating-point operations per second.

The AP–120B is no longer in production; this product has been superseded by the 5000 series product line. Though the AP–5000's are used at some AIPS sites, their advanced features are not used by AIPS—only those features which are shared by the older model are fully exploited by AIPS tasks.

array processor — a computer peripheral attachment which is capable of performing certain floating-point computations, especially vector and matrix operations, at high speed, and independently of the *host computer* central processing unit. Usually the high-speed performance is achieved by a technique known as pipelining. The basic arithmetic operations of addition and multiplication are performed in stages, by a so-called pipeline adder and a pipeline multiplier. These units operate just like an assembly line in a manufacturing plant. Some array processors (AP's) are constructed with multiple pipelines. Address computations are performed concurrently with the arithmetic operations, by a unit which is separate from the pipelines. The algorithms best-suited to an array processor implementation are those which can be structured so as to keep the pipelines filled a fair fraction of the time. Most AP's have their own high-speed data memory, but some are parasitic on the memory of the host computer. Portions of many AIPS tasks have been programmed for the Floating Systems, Inc. model *AP-120B array processor*, (*q.v.*). Also see *array processor microcode*, *Q-routine*, and *pseudo-array processor*.

array processor microcode — program source code written in the assembly language of an *array processor*, (*q.v.*). Array processor (AP) manufacturers usually provide an extensive library of utility subroutines that may be called from a

high-level programming language, such as Fortran; however, some computationally-intensive algorithms cannot be easily or efficiently implemented using only these libraries. Portions of these algorithms must be written in microcode—a painstaking process. The assembly languages of different models of AP's differ considerably (as do the subroutine libraries, too, in fact) because of differences in the hardware architectures. Thus the AIPS programming group tries to avoid writing microcode. But portions of the AIPS *tasks* for map making, deconvolution, and self-calibration are written in AP microcode. Also see *Q-routine*.

associated file — In AIPS, any two or more files among a collection consisting of a *primary data file* and all of its *extension files* are termed *associated*.

auto re-boot — a boot initiated by the computer itself, of its own volition. See *boot*.

back-up — The act of copying the contents of a computer file to some permanent storage medium such as magnetic tape or punched cards, for the purpose of protecting against accidental loss or in order to liberate storage space (e.g., disk space), is termed *backing-up*. The new copy of the file is termed a *back-up* copy, or simply a *back-up*. See *scratch*.

bandwidth smearing — in a radio interferometer map, space-variance of the *point spread function* which is attributable to non-monochromaticity, or finite bandwidth. The point spread function—at a particular point in a map—taking into account bandwidth smearing, but ignoring other instrumental effects, is termed a *delay beam*. Bandwidth smearing is a radial effect: the delay beams become more elongated, in the radial direction from the interferometer *phase tracking center*, as their distance from the phase tracking center increases. The delay beams are easily calculable when all of the receivers in an array have identical, and known, i.f. passbands. E.g., with rectangular passbands of width $\Delta\nu$, and observations centered at a frequency ν_0, the measured visibility amplitude of a point source is proportional to $\frac{\sin\gamma}{\gamma}$ —where $\gamma \equiv \pi(ux + vy + wz)\frac{\Delta\nu}{\nu_0}$, (u, v, w) denotes the spatial frequency coordinates, measured in wavelengths at ν_0, and (x, y, z) denotes the direction cosines of the location of the point source, with respect to the phase tracking center. For more details, see Alan Bridle and Fred Schwab's Lecture No. 13 and Bill Cotton's Lecture No. 12 in the *Third NRAO Synthesis Imaging Summer School* and see VLA Scientific Memo. No. 137.

Bandwidth smearing can, in principle, be eliminated (assuming that the bandpasses are known) by applying an image reconstruction algorithm which has a knowledge of the smearing mechanism; that is, by an algorithm which is more general than the usual deconvolution algorithms—see *image reconstruction*. The most common method for reducing bandwidth smearing is the technique of *bandwidth synthesis*, (*q.v.*).

bandwidth synthesis — a technique of radio interferometry which is intended to diminish the effect of *bandwidth smearing*. Bandwidth synthesis observing is very similar to spectral-line mode observing: the i.f. bandpasses are split up into a number of pieces, or channels, and the data in each channel are treated separately up until the mapping/deconvolution stage of processing. At that stage, the problem can be formulated as a system of simultaneous convolution equations: one has the system $g_1 = b_1 * f + \epsilon_1, \ldots, g_n = b_n * f + \epsilon_n$, where n is the number of frequency channels, g_k is the *dirty map* for channel k, b_k the *dirty beam* for that channel, f the unknown radio source brightness distribution (here assuming that f is not a function of frequency), and ϵ_k is noise (were it not for the noise, and for the fact that each deconvolution problem is ill-posed—in its own right—, there would be no reason to treat the equations simultaneously, or even to consider more than a single one of them). (For a description of a refinement to the bandwidth synthesis technique, for sources with spatially-varying spectral indices, see *broadband mapping technique*.) Note that all the b_k are identical, apart from a dilation factor; i.e., as the *u-v coverage* "shrinks", toward the low end of the observing band, the b_k dilate by the reciprocal of the *u-v* shrinkage factor.

The present state of software development does not allow solving the problem in quite the way it is formulated above. Rather, some mapping/deconvolution algorithm is applied separately to each of the channels, and the resulting maps are averaged.

baseline-time order — An ordered set of visibility measurements $\{ V_{ij}(t_k) \mid 1 \le i < j \le n,\ k = 1, \ldots, l \}$ recorded with an n element interferometer at times t_k is said to be in *baseline-time order* if the ordering is such that all of the data for the 1–2 baseline, sorted by time, occur first, followed by the data for the 1–3 baseline, again sorted by time, etc., etc. (This canonical ordering by baseline is the order $V_{12}, V_{13}, \ldots, V_{1n}, V_{23}, \ldots, \ldots, V_{n-1,n}$.) Compare *time-baseline order*.

Baseline-time ordering of a *u-v data file* is convenient for purposes of data display.

batch editor — a *text editor* within the AIPS program which allows the user to prepare *batch jobs* (*q.v.*), to be run non-interactively.

batch job — AIPS may be run either interactively—allowing the user to make 'split-second' decisions—or in batch mode. In batch mode, the user first decides on a set strategy for reducing the data, and then, using the special AIPS *batch editor*, the user prepares a *text file*, containing those AIPS commands which are appropriate to the anticipated data reduction needs. The batch job is placed in a *batch queue*, and the job steps are executed by the *batch processor*, in a non-interactive mode.

batch processor — the server, or scheduler, for *batch jobs* (*q.v.*). The AIPS batch processor follows certain rules in scheduling: batch jobs requiring the use of an array processor (AP) often are scheduled to run only during nighttime hours; the processor serving one of the *batch queues* might refuse service, altogether, to a job requiring an AP; and batch jobs may be given lower priority than those AIPS tasks which are run interactively.

batch queue — a waiting line for *batch jobs*. The AIPS batch queue is a single-server queue—i.e., the server (the *batch processor*) initiates the execution of the jobs one after the other, rather than in parallel. However, AIPS can be configured with more than one batch queue, each with its own batch processor; this number varies according to site.

"battery-powered" Clean algorithm — a modified version of the Clark Clean algorithm, devised by Fred Schwab and Bill Cotton. At each major cycle of the algorithm, or perhaps less frequently, the residual map is computed not by convolving the current iterate with the dirty beam map, but rather by computing the visibility residuals, and then re-gridding and re-mapping. By this means, the edge effects are compensated, and hence one can search the full dirty map field of view for Clean components. Simultaneously, instrumental effects (finite bandwidth and finite integration time) and sky curvature (the wz term) can be compensated for (i.e., the algorithm solves a more general equation than a convolution equation). See *Clark Clean algorithm*.

A "mosaicing" version of this algorithm is implemented in the AIPS task MX. The deconvolved image is defined over some number $1 \le n \le 16$ of rectangular patches. Within each patch, the data are corrected for sky curvature, by the correction appropriate to the center of the patch. Instrumental corrections are not included, at present.

beam — 1. in radio interferometry, the inverse Fourier transform (FT^{-1}) of the *u-v sampling distribution*, or FT^{-1} of a weighted u-v sampling distribution, possibly convolved with a gridding convolution function—the idealized response to a point, or unresolved, radio source. 2. a numerical approximation to 1. 3. a digitized version of 2, sampled on a regular grid (usually regarded as a map or image). 4. ≈ *point spread function*, q.v. 5. (occasionally) as above, but taking into account instrumental effects, so that the beam depends on position in the sky. See *dirty map*.

Occasionally, any one of the above, other than 5, is termed the *synthesized beam*.

beam patch — in the Clark Clean algorithm, that portion of the central part of the beam which is used in the inner iterations, or the minor cycles. In the AIPS implementation, the beam patch size typically is set at 101 pixels × 101 pixels. See *Clark Clean algorithm*.

beam squint — In radio interferometry, direction dependent, or space-variant *instrumental polarization*, which is difficult to calibrate, can arise from *beam squint*. The beam squint effect, for the usual case of a pair of (nominally) orthogonally polarized feeds on each array element, is due to differences in their power patterns—in particular, to differences in the directions of their peak response.

blanked pixel — in a digital image, a pixel whose value is undefined. In computer storage of quantized digital images, some special numeric value is assigned to the blanked pixels, so that they may be recognized as undefined and given whatever special treatment is required. See *pixel*.

BLC — *bottom left corner* (of an image). See $m \times n$ *map*.

blink — See *TV blink*.

boot — A computer is restarted by means of a *boot*strapping procedure, whereby the operating system and the data management facilities are re-initialized in a succession of steps. This ritual, through which the computer gathers it wits, is termed the *boot*. A boot (≈ *re-boot*) is required after any system crash (e.g., after a power failure). Usually the sequence of steps required to accomplish the boot is posted in a notice located close to the system operating console, or on the CPU panel. On modern computers, such as the Vax, the boot procedure is highly automated. In fact, there may be an abbreviated boot procedure, termed a *quick boot*, to follow after a "soft" system crash. (On such systems, a quick boot should be attempted before resorting to a full boot.) Indeed, some systems (the Vax included) re-boot on their own initiative following a soft system crash—this is termed an *auto re-boot*.

BOT marker — (Beginning-Of-Tape marker) a short strip of metal foil attached near the front, or beginning, end of a computer magnetic tape. The tape drive uses the BOT marker in order to position the tape at its starting position.

bpi — (bits per inch) the basic unit of measurement used to specify the density at which information is recorded on a computer magnetic tape: the effective number of bits per inch per track. The standard recording densities are 800, 1600, and 6250 bpi. Modern computer tapes are nine-track tapes: eight recording tracks are used for the data, and the ninth track is used to record "parity bits" for error-checking. See *tape blocking efficiency*.

broadband mapping technique — a refinement of the radio interferometric method of *bandwidth synthesis* (*q.v.*), in which one solves simultaneously for the radio brightness distribution $f_{\nu_r}(x,y)$ at some reference frequency ν_r, and for the (spatially varying) spectral index $\alpha(x,y)$ across the observing band. Assuming that the observing band is split into frequency channels centered at $\nu_1, \ldots, \nu_n$, one solves the simultaneous system of convolution equations $g_1 = b_1 * f_1, \ldots, g_n = b_n * f_n$, where g_k is the *dirty map* from channel k, b_k the *dirty beam* from that channel, and where f_k is given by

$$f_k(x,y) = \left(\frac{\nu_k}{\nu_r}\right)^{\alpha(x,y)} f_{\nu_r}(x,y)\,.$$

All of the b_k are identical, apart from a dilation factor. Assuming that the frequency channels are narrow enough, one can expand the *u-v coverage* considerably, with immunity to the *bandwidth smearing* effect. Fractional bandwidths as large as 20–30% can be used, depending on the linearity of the spectral index variations.

This mapping technique is described by Tim Cornwell [Broadband mapping of sources with spatially varying spectral index, VLB Array Memo. No. 324, Feb. 1984]. Extensive modification of one of the standard deconvolution algorithms is required. The requisite modification of the *Högbom Clean algorithm* is in progress.

b-t order — See *baseline-time order*.

bug — an actual or a perceived programming error or program deficiency. The bug may be in the eye of the beholder since the program user may fancy an application similar to, but differing from, the one for which the program is intended. In AIPS there is a formal mechanism for reporting program bugs; see *gripe file* for a description.

byte — a unit of eight bits of computer storage.

carriage-return key — One of the most used keys on any computer terminal keyboard is the carriage-return key (C_R). This is the button which ordinarily must be depressed when one has finished typing a command to the computer, in order for the computer to accept or acknowledge the command.

catalog entry — an entry within an AIPS *catalog file* ("CA" file) pertaining to a particular *primary data file*.

catalog file — In AIPS, each user has, for each disk on which he has data stored, his own *catalog file*, or "CA" file—a directory of all of his primary data files which reside on that disk. The AIPS *verb* CATALOG (as do its variants MCAT and UCAT) allows the user to see a summary listing of the contents of his catalog files. See *header record*.

catalog slot — in AIPS, a numbered space reserved in a *catalog file* for the insertion of a *catalog entry*.

cell-averaging — in radio interferometer mapping, gridding convolution which is achieved simply by averaging the visibility data which lie in each u-v grid cell. This is equivalent to use of a *gridding convolution function* equal to the *characteristic function* of the rectangle $\{|u| < \Delta u/2,\ |v| < \Delta v/2\}$, where Δu and Δv denote the grid spacing—i.e., it is equivalent to the use of a so-called *pillbox* function. The Fourier transform of the pillbox gridding convolution function is proportional to a separable product of two $\frac{\sin x}{x}$ functions; this function does not decay rapidly enough to yield very effective *aliasing* suppression. The zero-order *spheroidal functions* offer much better aliasing suppression, at somewhat increased computational expense (equivalent to averaging the data over a region 36 times larger, in the case of the default gridding convolution function used by the AIPS mapping tasks).

cellsize — in radio interferometer mapping, the size $\Delta u \times \Delta v$ of the u-v grid cells. Ordinarily, the visibility data are smoothed by an appropriate *gridding convolution function* and this convolution then is sampled at the coordinate locations of the centers of the grid cells. After appropriate weighting, the *discrete Fourier transform* yields the *dirty map*. Δu and Δv are chosen according to *Shannon's sampling theorem*: if the size of the dirty map is x radians by y radians, then $\Delta u = \frac{1}{x}$ wavelengths and $\Delta v = \frac{1}{y}$ wavelengths.

cereal bowl map defect — same as *negative bowl artifact*. See *zero-spacing flux*.

characteristic function — The characteristic function χ_A of a set $A \subset X$ is defined for all $x \in X$ by the formula

$$\chi_A(x) = \begin{cases} 1\,, & \text{if } x \in A\,, \\ 0\,, & \text{if } x \notin A \end{cases}$$

(χ_A is also called the *indicator function* of A, and the notations c_A and 1_A commonly are used in lieu of χ_A.) Note that this usage of the term, which is standard in mathematical analysis, differs from its usage in probability and statistics, where it refers to the Fourier transform of a probability measure (i.e., to the FT of the distribution function of a random variable).

chromaticity — in visual perception, essentially the dominant wavelength and the purity of the spectral distribution of light, as perceived. *Hue* and *saturation* determine the chromaticity, which is independent of *intensity*. See *C.I.E. chromaticity diagram*.

C.I.E. chromaticity diagram — a two-dimensional diagram devised in 1931 by the Commission Internationale de l'Eclairage (International Commission on Illumination) to show the range of perceivable colors as a function of normalized chromaticity coordinates (x, y), under standardized viewing conditions. The color, for an additive mixture of monochromatic red, green, and blue (R, G, B denoting the intensities at 650, 520, and 380 nm. wavelengths) as perceived by a 'standard observer', is displayed in this diagram as a function of the normalized *chromaticity coordinates* $x = R/(R + G + B)$ and $y = G/(R + G + B)$.

Other chromaticity diagrams can be drawn for different choices of primary hues, for mixtures of nonmonochromatic light, or for 'nonstandard observers'. In digital imagery, such a diagram may be tailored to a particular color image display unit. See [G. S. Shostak, Color basics—a tutorial. In R. Albrecht and M. Capaccioli, I.A.U. Astronomical Image Processing Circular No. 9, Space Telescope Science Institute, Jan. 1983] and [G. Wyszecki and W. S. Stiles, *Color Science*, Wiley, New York, 1967], a comprehensive textbook on colorimetry.

Clark Clean algorithm — a modified version of the Högbom Clean algorithm, devised by Barry Clark in order to accomplish an efficient *array processor* implementation of Clean (see [B. G. Clark, An efficient implementation of the algorithm Clean, *Astron. Astrophys.*, **89** (1980) 377–378]). To operate on, say, an $n \times n$ map, the original Clean algorithm requires on the order of n^2 arithmetic operations at each iteration, and typically there may be hundreds or thousands of iterations. The Clark algorithm proceeds by operating not on the full residual map, but rather by picking out only the largest residual points, iterating on these for a while (during its *minor cycles* or inner iterations) and only occasionally (at the *major cycles*) computing the full $n \times n$ residual map, by means of the FFT algorithm. After each major cycle, it again picks out the largest residuals and goes into more minor cycles. And, for further economy, during these inner iterations the dirty beam is assumed to be identically zero outside of a relatively small box (termed the *beam patch*) which is centered about the origin. See *Högbom Clean algorithm*.

Clean — See *Högbom Clean algorithm*.

Clean beam — in the Högbom Clean algorithm, an elliptical Gaussian function h with which the final iterate is convolved, in order to diminish any spurious high spatial frequency features—also termed *restoring beam*. h is specified by its major axis (usually the FWHM), its minor axis, and the position angle on the plane of the sky of its major axis. Usually these parameters are set by fitting to the central lobe of the dirty beam. See *Högbom Clean algorithm* and *super-resolution*.

Clean box — a rectangular subregion of a *Clean window* (*q.v.*).

Clean component — in the Högbom Clean algorithm, a δ-function component which is added to the $(n-1)$st iterate in order to obtain the nth iterate. Its location is the location of the peak residual after the $(n-1)$st iteration, and its amplitude is a fraction μ (the *loop gain*) of the largest residual. See *Högbom Clean algorithm*.

The AIPS task implementing the (Clark) Clean algorithm stores a list of the Clean components in an extension file which is termed a *components file*.

Clean map — an approximate deconvolution of the *dirty beam* from the *dirty map*, derived by an application of the Högbom Clean algorithm or one of its derivatives. See *Högbom Clean algorithm*.

Clean speed-up factor — in the *Clark Clean algorithm*, a number α in the range $[-1, 1]$ used in determining when to end a major cycle. Smaller α causes a larger number of major cycles to occur (at greater computational expense) but yields a result closer to that of the classical *Högbom Clean algorithm*.

Clean window — in the Högbom Clean algorithm, the region A of the residual map which is searched in order to locate the *Clean components* comprising the successive approximates to the radio source brightness distribution. In the AIPS implementation, A is a union of rectangles, called *Clean boxes*, which may be specified by the user. When A is not explicitly specified, the algorithm searches over the central rectangular one-quarter area of the residual map. See *window Clean* and *Högbom Clean algorithm*.

clipping — the discarding (i.e., the *flagging*) of visibility data whose amplitudes exceed some threshold value, or the discarding of visibility data whose differences from some tentative source model are too large in amplitude. The AIPS task CLIP is used for clipping. See *u-v data flag*.

closure amplitude — Assume that the visibility observation on the i–j baseline ($i < j$) is given by $\tilde{V}_{ij} = g_i \overline{g}_j V_{ij}$, where V_{ij} is the true visibility and where g_i and g_j are the *antenna/i.f. gains* (ignore any additive error). Then, for certain combinations of (at least four) baselines, one may form ratios of observed visibilities (and their conjugates)—including each visibility only once—in such a manner that the g's cancel one another. For example, if $i < j < k < l$, then

$$\frac{\tilde{V}_{ij}\tilde{V}_{kl}}{\tilde{V}_{il}\overline{\tilde{V}}_{jk}} = \frac{V_{ij}V_{kl}}{V_{il}\overline{V}_{jk}}\,.$$

The modulus of such a ratio is termed a *closure amplitude* (and its argument, a *closure phase*).

Closure amplitude is called a "good observable", since, under the above assumptions, it is not sensitive to measurement error. The closure amplitude and closure phase relations are exploited in the *hybrid mapping algorithm* (*q.v.*). Also see *self-calibration algorithm*.

closure phase — Assume that the visibility observation on the i–j baseline ($i < j$) is given by $\tilde{V}_{ij} = g_i \overline{g}_j V_{ij}$, where V_{ij} is the true visibility and where g_i and g_j are the *antenna/i.f. gains* (ignore any additive error). Then, for a combination of any three or more baselines forming a closed loop, one may sum the visibility phases in such a manner that the *antenna/i.f. phases* ψ_k drop out. For example, if $i < j < k$, then $\arg\tilde{V}_{ij} + \arg\tilde{V}_{jk} - \arg\tilde{V}_{ik} = \arg V_{ij} + \psi_i - \psi_j + \arg V_{jk} + \psi_j - \psi_k - \arg V_{ik} - \psi_i + \psi_k$. Such a linear combination of observed visibility phases is termed a *closure phase*.

Closure phase is called a "good observable", since, under the above assumptions, it is not sensitive to measurement error. The closure phase relations are exploited in the *hybrid mapping algorithm* (*q.v.*). Also see *closure amplitude* and *self-calibration algorithm*.

color contour display — a color digital image display of a real-valued function f of two real variables (x, y), in which the color assignment (the *hue*) is a coarsely quantized function of $f(x, y)$. The visual effect of this type of *pseudo-color display*, in the case when f is continuous, is similar to the traditional sort of contour display. One sees curves along which f is constant, separated by swathes of constant hue—each hue corresponding to a distinct quantization level.

color triangle — Any three non-collinear points plotted on a chromaticity diagram determine a color triangle. Since the points are non-collinear, they correspond to basic, or *primary* hues. All of those colors on the chromaticity diagram which fall within the triangle determined by the three points may be produced by addition of the three hues. See *C.I.E. chromaticity diagram*.

compact support — See *support*.

components file — in AIPS, an extension file, associated with an image file containing a *Clean map*, whose content is a list of the positions and amplitudes of the *Clean components* included in that Clean map, as determined by the Clean algorithm. The source model specified by this list of components often is used in *self-calibration*.

conjugate symmetry — that property which characterizes a *Hermitian function* (*q.v.*). Generally an assumption of conjugate symmetry is implicit whenever one speaks of the *u-v coverage* corresponding to some radio interferometric observation.

Conrac monitor — the CRT unit of the I^2S TV display device, in use at a number of AIPS installations. See *I^2S*.

convolution theorem — This theorem is well-known, but seldom is quoted in its distributional form: for two distributions, f and g, the Fourier transform of the convolution of f and g is given by $\widehat{f * g} = \hat{f}\hat{g}$, whenever one distribution is of *compact support* and the other is a "tempered" distribution. (Loosely speaking, a tempered distribution is one which does not increase too rapidly at infinity.) See [Y. Choquet-Bruhat, C. Dewitt-Morette, and M. Dillard-Bleick, *Analysis, Manifolds, and Physics*, North–Holland, New York, 1977, ch. VI].

One ought to be aware of this form of the theorem, since often one must deal with convolution of functions that are not of compact support—*dirty beams, principal solutions, invisible distributions,* etc.—whose Fourier transforms do not exist as ordinary functions, but only as distributions or generalized functions.

Convolution of distributions, itself, is defined, in general, whenever the support of either distribution is compact, or (in one dimension) when the supports of both distributions are limited on the same side. For distributions which are absolutely integrable ordinary functions, and whose Fourier transforms possess the same property, the compact support assumption is not required here, or above. Related fact: convolution is not always associative (i.e., $f * (g * h) \neq (f * g) * h$), in general), but it is associative provided that all the distributions, with the possible exception of one, are of compact support. See the above-cited reference.

convolving function — See *gridding convolution function*.

coordinate reference pixel — in an AIPS *image file*, a *"pixel"* whose coordinates are recorded in the image *header* together with the coordinate increments (i.e., the pixel coordinate separations) that allow the physical coordinates of all other pixels in the image to be computed. This "coordinate reference pixel" may not actually be present in the image: all that matters are its physical coordinates and its pixel coordinates (which too are recorded in the header—and which may, in fact, be fractional).

Often, in a radio map (and by default, when the standard AIPS map making tasks are executed), the position of the coordinate reference pixel coincides with the map center and with the *visibility phase tracking center*. See *$m \times n$ map* and *pixel coordinates*.

correlator offset — One of the basic assumptions of much of the VLA calibration software (e.g., the *self-calibration algorithm*) is that the systematic errors in the visibility measurements are multiplicative errors that are ascribable to individual array elements and their associated i.f./l.o. chains, and that—at a given instant—each such antenna-based error has an identical effect on each visibility observation involving that antenna/i.f. combination. Systematic measurement errors which do not conform to this model are called *correlator offsets* or *non-closing errors*. See *antenna/i.f. gain*.

Correlator offsets can be the limiting factor in obtaining high dynamic range VLA maps. Some observers have reported fairly large multiplicative correlator offsets which vary slowly with time and which do not appear to vary with the *phase tracking center* or with source structure. From observations of an external calibrator, one may estimate, and compensate for, such offsets. This mechanism is provided in the AIPS tasks BCAL1 and BCAL2. See [R. C. Walker, Non-closing offsets on the VLA, VLA Scientific Memo. No. 152].

crash — the abrupt failure of a computer system or program. More specifically, a *system crash* is the abrupt failure of a computer—or of a computer's operating system—causing the computer to halt the execution of programs; and a *program crash* is the abrupt failure of a computer program resulting either from a flaw in the logic of the program itself, or from some peculiar interaction with the operating system, the storage management facility, another program, or the user—or from an act of God. A *hardware crash* (e.g., a *disk crash*) is a crash which results from the failure of the computer electronics or electro-mechanics, and a *software crash* is one which results from a flaw or an inadequacy in program logic, or in operating system program logic. A *soft crash* is a crash from which it is easy to recover—i.e., easy to restart the computer and resume work—, and a *hard crash* is the opposite.

crosshair — 1. a marker on the *TEK screen*, or *green screen*, which may be moved about through the use of thumbwheel knobs which are located on the terminal keyboard panel. The position of the crosshair may be sensed by the computer program, and thus the user may point out to the program features that are of interest in the graphical display on the CRT screen. 2. a marker with the same function as just described, but on a TV display device, and more likely controlled by a *trackball* than by thumbwheels. Same as *TV cursor*; and see *trackball*.

cube — See *data cube*.

cursor — 1. a marker on an interactive computer terminal indicating the position on the CRT screen where the next character is to be typed. 2. *TV cursor*—on a TV display device, a marker whose manually controlled position may be sensed by the computer. See *crosshair*.

data cube — 1. in VLA spectral line data analysis, a three-dimensional map or "image" representing a function of three real variables—two spatial variables representative of position in the sky, and one variable related to frequency or velocity. 2. any n-dimensional *image*, $n \geq 3$.

Computer access of a multi-dimensional data array, residing in any standard type of storage medium such as disk or magnetic tape, is sequential, as if the data were one-dimensional. Spectral

line data cubes are stored plane-by-plane, row-by-row, column-by-column. Permutation of the correspondence between plane, row, and column, and the coordinate axis numbering, is referred to as *transposition* of the data cube.

database — a computer filing system, or file structure system. For example, the AIPS database consists not only of the data themselves, but also of the directories and the cross-reference lists of all the AIPS data files (including extension files), the data format definitions, etc., as well as the rules and principles governing the use thereof.

data file — on a computer storage medium, such as disk or magnetic tape, the concrete, or physically present representation of a logically distinct grouping of data in a manner permitting repeated access by computer programs.

data flag — See *u-v data flag*.

deconvolution — the numerical inversion of a convolution equation, either continuous or discrete, in one or several variables; i.e., the numerical solution (for f) of an equation of the form $f * g = h +$ noise, given g and given the right-hand side of the equation. Except in trivial cases, deconvolution is an ill-posed problem: In the absence of constraints or extra side-conditions, and in the case of noiseless data—assuming that some solution exists—there usually will exist many solutions. In the case of noisy data, there usually will exist no exact solution, but a multitude of approximate solutions. In the latter case, if one is not careful in the choice of a numerical method, the computed approximate solution is likely not to have a continuous dependence on the given data. The so-called *regularization method* (*q.v.*) (of which the *maximum entropy method* is a special case) is an effective tool for the deconvolution problem.

Discrete two-dimensional deconvolution is an everyday problem in radio interferometry, owing to the fact that—under certain simplifying assumptions—the so-called *dirty map* is the convolution of the *dirty beam* with the true celestial radio image. In addition to the maximum entropy method, the *Högbom Clean algorithm* is commonly applied to this problem. See Tim Cornwell and Robert Braun's Lecture No. 8 in the *Third NRAO Synthesis Imaging Summer School*.

delay — See *residual delay*.

delay beam — in radio interferometry, the *point spread function* or *beam*, taking into account *bandwidth smearing*, but ignoring other instrumental effects. See *bandwidth smearing*.

DFT — an abbreviation for *discrete Fourier transform* and *direct Fourier transform* (*q.v.*). When used in disciplines other than radio astronomy, it usually signifies the former.

Dicomed Image Recorder (Model D47) — a computer-controlled image display device intended for photographic reproduction of digital images. The film is exposed by a cathode ray tube. The device is capable of 4096 pixel × 4096 pixel resolution and of both black-and-white and color reproduction. The digital exposure control and eight-bit pixel input allow 256 discrete exposure levels. The CRT has a single electron gun and a screen with a white phosphor; color reproduction is accomplished by means of multiple exposures, with the insertion of red, green, and blue filters. There is a Dicomed recorder at the NRAO in Charlottesville, and another at the VLA.

direct Fourier transform — a term used imprecisely in radio astronomy to mean either: 1) a finite trigonometric sum, of the form

$$\sum_{j=0}^{n-1} a_j e^{2\pi i u_j x} ,$$

with a_j complex, where the (real) u_j are irregularly-spaced; 2) the brute-force *evaluation* of such a sum; or 3), the naïve, or brute-force evaluation (using $O(n^2)$ arithmetic operations) of the (n-point) *discrete Fourier transform*.

The direct Fourier transform, in senses 1) and 2) of the definition, arises in synthesis mapping applications because of the irregular distribution of the visibility measurements. Common practice is to use a *gridding convolution function* to interpolate the data onto a regularly-spaced lattice, so that, for computational economy, the *fast Fourier transform algorithm* may be used.

dirty beam — in radio interferometry, simply a *beam,* but computed with precisely the same operations as those used to compute some companion *dirty map* (i.e., with the same u-v coverage, the same manner of gridding convolution, the same u-v weight function and taper, etc.). In Cleaning a dirty map, only the companion dirty beam should be used.

dirty map — 1. ignoring instrumental effects, the inverse Fourier transform (FT^{-1}) of the product of the visibility function V of the radio source and the (possibly *weighted* and/or *tapered*) u-v *sampling distribution* S; i.e., FT^{-1} of the u-v *measurement distribution.* 2. a discrete approximation to 1; in this case, the product SV is convolved with some function C, of *compact support,* and an inverse discrete Fourier transform of samples of $C * (SV)$ taken over a regular grid yields the *dirty map.* 3. as in 2, but corrected for the taper ($\check{C}$, the FT^{-1} of C) induced by the convolution. 4. any of the above, but now taking into account various instrumental effects (receiver noise, non-monochromaticity or finite bandwidth, finite integration time, sky curvature, etc.).

If it is assumed that $V \equiv 1$, then the map, or point source response, so obtained is termed the *beam* (*q.v.*). Also see *gridding convolution function, u-v taper function, u-v weight function, dirty beam,* and *principal solution.*

discrete Fourier transform — The (one-dimensional) discrete Fourier transform (DFT) $y_0, \ldots, y_{n-1}$ of a sequence of complex numbers $x_0, \ldots, x_{n-1}$ is given by the summation

$$y_k = \sum_{j=0}^{n-1} x_j e^{2\pi i j k/n} .$$

(The multi-dimensional generalization is straightforward). The x_j are given by the *inverse DFT* of the y_k:

$$x_j = \frac{1}{n} \sum_{k=0}^{n-1} y_k e^{-2\pi i j k/n} .$$

(Frequently the forward and inverse transforms are defined in the manner opposite to that given here, and the $\frac{1}{n}$ normalization factor sometimes is moved about.) The DFT arises most naturally in numerically approximating the Fourier coefficients $c_m = \frac{1}{2\pi} \int_0^{2\pi} f(x) e^{-imx}\, dx$ of a 2π-periodic function f which is representable by the trigonometric series $\sum_{m=-\infty}^{\infty} c_m e^{imx}$. The *fast Fourier transform algorithm* (*q.v.*) can be used for efficient numerical evaluation of the DFT.

disk hog — a derogatory term, used to connote a computer user whose disk data files are excessively voluminous or numerous, therefore putting other computer users at a relative disadvantage. Unneeded data files should be *scratched*, or destroyed, in order to free up disk space. Large disk files which will not be needed for a time should be *backed-up* on magnetic tape and then deleted from disk.

dynamic range — a summary measure of image quality indicative of the ability to discern dim features when relatively

stronger features are present—i.e., a measure of the ability to distinguish the dim features from artifacts of the *image reconstruction* procedure (in a radio map, from remnants of the sidelobes of stronger features) and from noise. The dynamic range achievable in a radio interferometer map is determined primarily by the uniformity of the *u-v coverage*, the density and extent of the coverage, the sensitivity of the array, and the quality of the calibration.

If the true radio source brightness distribution f is known, one can define the dynamic range of a reconstruction $\tilde{f}$ as, say, the ratio of the maximum value of $|f|$ to the r.m.s. difference between f and $\tilde{f}$. When f is unknown, as is usually the case, an empirical measure of the dynamic range is used—perhaps the ratio of the maximum value of $|\tilde{f}|$ to the r.m.s. level in an apparently empty region of the map, or the ratio of the strongest feature to the weakest "believable" feature—, but there is no widely-accepted definition.

What one might wish to call the "true" dynamic range of a radio map is a spatially-variant quantity. The ability to discern a dim feature depends on its proximity to brighter features, because there are relatively stronger sidelobe remnants near the bright features. The quality of a map (and perhaps the dynamic range—depending on how it is defined) deteriorates away from the *phase tracking center*, because of the inability of the image reconstruction algorithms to compensate for various instrumental effects (e.g., bad pointing, *bandwidth smearing*, etc.).

EDT — a sophisticated text editor (a *screen editor*) used on the Vaxes. It makes use of the "keypad" feature of the fancier terminals. EDT can be run only on certain model terminals: on the DEC (Digital Equipment Corp.) Models VT–52 and VT–100, and on terminals such as the Visual–50's and the Visual–100's which are capable of emulating the DEC terminals. See *text editor*.

EMACS — a sophisticated text editor used on the Vaxes, as well as on many computers which run under the UNIX operating system. (There is also a version for the IBM-PC.) EMACS is a *screen editor*, and the one which is favored by most among those in the AIPS programming group. On terminals with the "keypad" feature, the keypad keys can be programmed by the user to perform many useful editing tasks; however, EMACS can be run from other models of terminals, as well. EMACS provides two powerful and convenient features which most other text editors do not offer: the ability to temporarily exit from the editor and "return to monitor level," and the ability to initiate an interactive "job control session," or initiate *sub-tasks*, in an EMACS buffer. See *text editor*.

explain file — in AIPS, a *text file* containing a detailed explanation of a particular AIPS *task* or *verb*, often including hints, suggested applications, algorithmic details, and bibliographical references. Issuing the AIPS verb EXPLAIN causes the contents of an explain file to be printed on the terminal screen or on a line printer. Compare *help file*.

EXPORT format — a visibility data magnetic tape format for transport of VLA data from the DEC–10 computer or the on-line computer at the VLA.

EXPORT tape — a magnetic tape containing data recorded in the *EXPORT format*.

exp $\times$ sinc function — a useful gridding convolution function: same as the *Gaussian-tapered sinc function* (*q.v.*), except that the exponent of the argument to the exponential function may be other than two.

extension file — in AIPS, a data file containing data supplemental to those contained in a *primary data file* (either a *u-v data file* or an *image file*). Whenever a primary data file is deleted by the standard mechanism within AIPS for file destruction, all extension files associated with that primary data file also are destroyed. Extension files, however, may be deleted without deleting the the associated primary data file.

Extension files are grouped into categories of named types. Examples: *plot files*, *history files*, *slice files*, *gain files*, etc.

When an AIPS task creates a new primary data file from an old one, generally it attaches, to the new file, clones of any extension files associated with the old file that remain relevant to the new one.

false color display — In digital imagery, a *false color display* is one which is generated by using a number $n > 1$ of real-valued functions $f_1(x, y), \ldots, f_n(x, y)$ to control the proportions, at each *pixel* coordinate (x, y), of an additive mixture of three primary hues. In practical terms, the user of a digital display system supplies $f_1, \ldots, f_n$, and twists knobs that control the mapping $\mathbf{R}^n \rightarrow \mathbf{R}^3$ that sends the n pixel values at each (x, y) into the proper image *chromaticity* and *intensity*. Compare *pseudo-color display*.

A so-called *true color display* is obtained with $n = 3$ and with *transfer functions* chosen such that the color assignment corresponds in an approximate way to the actual coloration of a scene (as in a color photograph).

fast Fourier transform algorithm — a fast algorithm for the computation of the *discrete Fourier transform* (DFT) $y_0, \ldots, y_{n-1}$ of a sequence of n complex numbers $x_0, \ldots, x_{n-1}$,

$$y_k = \sum_{j=0}^{n-1} x_j e^{2\pi i j k/n} ,$$

typically requiring only $O(n \log n)$ arithmetic operations — or a multi-dimensional generalization thereof. By contrast, straightforward, or naïve evaluation of the DFT requires $O(n^2)$ operations. The fast Fourier transform algorithms (FFT's) which currently are the most popular are the Cooley–Tukey (1965) algorithms, for the case of n highly composite. For n a power of two, the (radix-2) Cooley–Tukey FFT requires about $2n \log_2 n$ real multiplications and $3n \log_2 n$ real additions. More generally, the Cooley–Tukey algorithms require a few times $n\sigma(n)$ complex arithmetic operations, where $\sigma(n)$ is the sum of the prime factors of n, counting their multiplicities. S. Winograd has produced FFT algorithms which are more efficient than those of Cooley and Tukey, typically requiring about the same number of additions, but only about 20% the number of multiplications. (Computation of the required complex exponentials—or sines and cosines—is not counted, since these generally are either pre-computed and stored in compact tables, or generated recursively.)

A further advantage of the FFT algorithms is their avoidance of round-off error, which can build up severely when the DFT is evaluated by brute-force. There are related, fast algorithms for the convolution of sequences of real numbers, for the discrete cosine transform, etc. Algorithmic details may be found in [H. J. Nussbaumer, *Fast Fourier Transform and Convolution Algorithms*, Springer–Verlag, Berlin, 1982]. The computational complexity of the DFT is discussed by L. Auslander and R. Tolimieri [Is computing with the finite Fourier transform pure or applied mathematics, *Bull. (New Series) Amer. Math. Soc.*, **1** (1979) 847–897].

AIPS programs which use the FFT make use of the Cooley–Tukey algorithm. When an *array processor* is used to compute the large two-dimensional DFT's of data which reside on disk, as typically is required in synthesis mapping, the input/output time greatly exceeds the actual computation time.

FFT — See *fast Fourier transform algorithm.*

FITS format — (Flexible Image Transport System) a magnetic tape data format well-tailored for the transport of image

data among observatories. The FITS format is recommended for bringing data into and out of AIPS. See [D. C. Wells, E. W. Greisen, and R. H. Harten, FITS: A flexible image transport system, *Astron. Astrophys. Suppl. Ser.*, **44** (1981) 363–370]. Also see *u-v FITS format* and *FITS tape*.

FITS tape — a magnetic tape containing data recorded in the *FITS format*. FITS format data blocks are 2880 bytes in length. The resultant *tape blocking efficiency* is 83%, 75%, and 61% at recording densities of 800, 1600, and 6250 bpi, respectively.

flagging — in AIPS, the act of discarding one or more visibility data points by setting a *u-v data flag* (*q.v.*). Compare *clipping*.

fringe rotator — in a correlating-type radio interferometer, a mechanism to introduce a time-varying phase shift into the local oscillator signal of a receiver, in order to reduce the frequency of the oscillations of the correlator output. Fringe rotation allows the correlator output (whose amplitude is proportional to visibility amplitude) to be sampled at a lower rate. The natural fringe frequency can be as high as 200 Hz on the VLA. The fringe rotation is chosen so that the fringe frequency for a point source located at the so-called *fringe stopping center* would be reduced to zero, or at least close to zero. Usually the fringe stopping center and the *delay tracking center* coincide; both then are called the *visibility phase tracking center*. For further details, see A. R. Thompson's Lecture No. 2 and L. R. D'Addario's Lecture No. 4 in the *Third NRAO Synthesis Imaging Summer School*, and see R. M. Hjellming and J. Basart's Ch. 2 of the *Green Book*.

full-synthesis map — in earth-rotation aperture synthesis, with stationary interferometer elements, a *map* derived from an observation which is of such lengthy duration that the fullest possible *u-v coverage* is obtained (i.e., from an observation extending from "horizon to horizon"). Compare *snapshot*.

gain file — in AIPS, an *extension file*, associated with a *u-v data file*, in which a table of approximate *antenna/i.f. gains* (typically obtained by *self-calibration*) is stored.

Gaussian-tapered sinc function — A useful *gridding convolution function* (*q.v.*), of *support width* equal to the width $m\Delta u$ of m u-v grid cells, is given by the separable product of two Gaussian-tapered sinc functions, each of the form

$$C(u) = \begin{cases} \left(\frac{\pi u}{b\Delta u}\right)^{-1} e^{-\left(\frac{\pi u}{a\Delta u}\right)^2} \sin\frac{\pi u}{b\Delta u}\,, & |u| < \frac{m\Delta u}{2}\,, \\ 0\,, & \text{otherwise}\,. \end{cases}$$

The choice $m = 6$, $a \simeq 2.52$, and $b \simeq 1.55$, yields what is, in a certain natural sense, an optimal gridding convolution function of this particular parametric form (see [F. R. Schwab, Optimal gridding, VLA Scientific Memo. No. 132]). Also see *spheroidal function*.

Gerchberg–Saxton algorithm — a simple iterative algorithm which, in the field of signal processing, is used for the extrapolation of band-limited signals—and, in image processing, for deconvolution. Assume that the Fourier transform $\hat{f}$ of an image f has been measured over a region B, and that f is known to be confined to a region A. Let χ_A denote the *characteristic function* of A and χ_B that of B. Denote the measured data by $\hat{g}_{\rm approx}$—i.e., $\hat{g}_{\rm approx} = \chi_B\hat{f} + \text{error}$. From the initial approximant f_0 ($f_0 \equiv 0$ may be used) a sequence f_n of successive approximants to f is obtained, via the formula

$$f_{n+1} = f_n + \mu\chi_A \cdot (\hat{g}_{\rm approx} - \chi_B\hat{f}_n)\check{}\,.$$

Here, $\check{}$ denotes inverse Fourier transform, and μ is a fixed scalar, analogous to the *loop gain* parameter of the Högbom Clean algorithm.

To apply the algorithm in radio interferometry, one may identify χ_B with the *u-v sampling distribution* and think of A to be analogous to a *Clean window*. Denoting the *dirty map* by g and the *dirty beam* by b, the iteration can be written as

$$f_{n+1} = f_n + \mu\chi_A \cdot (\hat{g} - \hat{b}\hat{f}_n)\check{}\quad (= f_n + \mu\chi_A \cdot (g - b * f_n))\,.$$

The Gerchberg–Saxton algorithm has been implemented by Tim Cornwell in an AIPS program named APGS. APGS includes an *ad hoc* non-negativity constraint—at each iteration, any pixel value which would be driven negative is modified to become non-negative. Convergence usually is sluggish.

Some algorithms which are very similar to the Gerchberg–Saxton algorithm are the Lent–Tuy algorithm, which is used in medical imaging, the Papoulis, or Papoulis–Youla algorithm, used in signal processing, and the so-called method of alternating orthogonal projections, used in image reconstruction. See [J. L. C. Sanz and T. S. Huang, Unified Hilbert space approach to iterative least-squares linear signal restoration, *J. Opt. Soc. Am.*, **73** (1983) 1455–1465] and references cited therein.

Gibbs' phenomenon — in the neighborhood of a discontinuity of a periodic function f, the overshoot and oscillation (or ringing) of the partial sums S_n of the Fourier series for f. In the vicinity of a simple jump discontinuity, S_n always overshoots the mark by about 9%, regardless how large n. See [H. S. Carslaw, *Introduction to the Theory of Fourier's Series and Integrals*, Dover, New York, 1930, ch. IX].

In harmonic analysis, often the Fourier coefficients are multiplied by a weight function tending smoothly to zero at the boundaries of its *support*, in order to smooth out the discontinuities and thereby reduce the ringing in the synthesized spectrum. (This degrades the spectral resolution, however.) See *Hanning smoothing*. For a discussion of Gibbs' phenomenon in the context of VLA cross correlation analysis, see Larry D'Addario's Lecture No. 4 in the *Third NRAO Synthesis Imaging Summer School*.

GIPSY — (Groningen Image Processing System) a data reduction system, similar in scope to AIPS, used in the Netherlands for analysis of Westerbork Synthesis Radio Telescope (WSRT) data.

global fringe fitting algorithm — an antenna-based algorithm (in the spirit of the *self-calibration algorithm*) for VLBI fringe search. For an n element array, the classical VLBI fringe fitting technique, a correlator-based method, requires the estimation of $n^2 - n$ parameters. The global fringe fitting method reduces this number to $3n - 3$. Expressing the *antenna/i.f. gain* for antenna k of the array as $g_k(t,\nu) = a_k e^{i\psi_k(t,\nu)}$ (here we include a frequency dependence) one has that the observed visibility on the i–j baseline, to first-order, is given by

$$\begin{aligned} V_{ij}(t,\nu) \quad &= \quad a_i a_j V_{ij}(t_0,\nu_0) \times \\ &\qquad e^{\sqrt{-1}((\psi_i-\psi_j)(t_0,\nu_0)+(r_i-r_j)(t-t_0)+(\tau_i-\tau_j)\nu-\nu_0))}\,, \end{aligned}$$

where V_{ij} is the true visibility, and where the r_k are the *antenna residual fringe rates* and the τ_k the *antenna residual delays*.

Given a source model, one may solve for the $\psi_k(t_0,\nu_0)$, the r_k, and the τ_k, using either a least-squares method or a Fourier transform method. Because of the overdeterminacy provided by a simultaneous solution for the parameters, this method allows proper delay and fringe rate compensation of data on baselines of too low signal-to-noise for the correlator-based method to work effectively. A full description of the method is given in [F. R. Schwab and W. D. Cotton, Global fringe search techniques for VLBI, *Astron. J.*, **88** (1983) 688–694]. This algorithm is implemented in the AIPS program CALIB.

graphics overlay plane — same as *graphics plane.*

graphics plane — a storage area within a TV display device, such as the I^{2}S, in which a full screen load of one-bit graphics information (labeling, plotting, axis lines, etc.) is stored. A typical I^{2}S unit is equipped with four graphics planes, each 512 pixels × 512 pixels in area. Compare *image plane*.

gray-scale display — a black-and-white display of a digitized *image*—typically either a photographic or a video display.

gray-scale memory plane — same as *image plane*.

Green Book — *An Introduction to the NRAO Very Large Array*, edited by R. M. Hjellming, NRAO, Socorro, NM—a useful reference on many of the technical aspects of the VLA.

green screen — same as *TEK screen*.

gridding convolution function — in radio interferometer map making, a function C—usually supported on a square the width of, say, six u-v grid cells—with which the u-v measurement distribution is convolved. The purpose is twofold: 1) to interpolate and smooth the data, so that samples may be taken over the lattice points of a rectangular grid (in order that the fast Fourier transform algorithm may be applied) and 2) to reduce aliasing (the convolution in the u-v plane induces a taper in the map plane). See *aliased response, gridding correction function, cell-averaging, dirty map,* and *uniform weighting*.

With judicious choice of C, a high degree of aliasing suppression is possible. A high degree of suppression is desirable, even when there are no "confusing" radio sources very near the field of interest, because the effect is not only to reduce the spurious responses due to sources lying outside of the field of view, but also to reduce the response to sidelobes of the source of interest, which too are aliased into the map from outside the field of view. See *spheroidal function.*

gridding correction function — in radio interferometry, the reciprocal $1/\hat{C}$ of the Fourier transform (FT) of the *gridding convolution function* C. Since the map plane taper induced by the gridding convolution usually is very severe, the dirty map normally is corrected by point-wise division by the FT of the convolution function. Obviously C should be chosen such that $\hat{C}$ has no zeros within the region that is mapped. See *dirty map*.

gripe — in AIPS, an entry in the *gripe file* (*q.v.*).

gripe file — in AIPS, a disk file repository for formal reports of program *bugs*, and for formal complaints and suggestions of a more general nature. A mechanism by which the user may enter gripes into the gripe file is activated by the issuance of the AIPS verb *GRIPE*. The AIPS group provides prompt, written responses to all *gripes*.

Hanning smoothing function — in the analysis of power spectra, a weight function w by which the measured correlation function is multiplied, in order to reduce that oscillation (*Gibbs' phenomenon*) in the computed spectrum which is due to having sampled at only a finite number of lags. w, as a function of lag, is given by

$$w(\tau) = \begin{cases} \frac{1}{2}\left(1 + \cos\frac{\pi\tau}{\tau_{\max}}\right), & |\tau| < \tau_{\max}, \\ 0, & \text{otherwise}. \end{cases}$$

This is equivalent to convolving the discrete spectrum with the sequence $\{\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\}$.

Hanning smoothing sometimes is applied to the cross correlation measurements obtained in VLA spectral line observing, in order to reduce the effect of sharp bandpass filter cutoffs. It also is used frequently in radio astronomical autocorrelation spectroscopy. See *Gibbs' phenomenon*, and for more on smoothing see [R. B. Blackman and J. W. Tukey, *The Measurement of Power Spectra*, Dover, New York, 1958].

hard copy — computer output printed on paper (rather than, say, written on magnetic tape); e.g., a printed contour plot or gray scale display, or a listing of a catalog file.

hardware mount — the combined acts of installing a computer external storage module, such as a disk pack or a reel of magnetic tape, in some electro-mechanical unit (e.g., a disk drive or a tape drive) that provides computer access to this data storage medium, and placing that unit in readiness to be operated under computer control (e.g., positioning a magnetic tape at the *BOT marker*). Compare *software mount.*

header record — a distinguished record within a *data file*—generally the first record—which serves to define the contents of the other records in the file by supplying relevant parameters, units of measurement, etc.; also termed simply *header*.

In AIPS, however, the header record of each *primary data file* is stored apart from that file, in a file which is termed a "CB" file. And a directory, termed a *catalog file* (*q.v.*), or "CA" file, of all of each user's primary data files on a given disk is stored on that disk. AIPS *extension file* headers are stored within the extension files themselves.

help file — in AIPS, a *text file*, whose contents may be displayed on the terminal screen of the interactive user, giving a brief explanation of a particular AIPS verb, adverb, pseudo-verb, task, or miscellaneous general feature. Compare *explain file*.

Hermitian function — a complex-valued function, of one or more real variables, whose real part is an even function and whose imaginary part is odd. The Fourier transform (FT) of a real-valued function is Hermitian, and the inverse FT of a Hermitian function is real.

Since each of the radio brightness distributions $I(x,y)$, $Q(x,y)$, $U(x,y)$, and $V(x,y)$ representing *Stokes' parameters* is real-valued, *Stokes' visibility functions* have the property of *conjugate symmetry*: $V_I(-u,-v) = \overline{V}_I(u,v)$, $V_Q(-u,-v) = \overline{V}_Q(u,v)$, $V_U(-u,-v) = \overline{V}_U(u,v)$, and $V_V(-u,-v) = \overline{V}_V(u,v)$. (Here, $V_I = \hat{I}$, $V_Q = \hat{Q}$, etc., where $\hat{}$ denotes FT.)

history file — in AIPS, an *extension file* containing a summary of all, or most of the processing, by AIPS tasks, of the data recorded in all associated files.

Hogbom Clean algorithm — a deconvolution algorithm devised by Jan Högbom for use in radio interferometry [J. A. Högbom, Aperture synthesis with a non-regular distribution of interferometer baselines, *Astron. Astrophys. Suppl. Ser.*, **15** (1974) 417–426]. Denote (the discrete representations of) the dirty map by g and the dirty beam by b. The algorithm iteratively constructs discrete approximants f_n to a solution f of the equation $b * f = g$, starting with an initial approximant $f_0 \equiv 0$. At the nth iteration, one searches for the peak in the residual map $g - b * f_{n-1}$. A δ-function component, centered at the location of the largest residual, and of amplitude μ (the *loop gain*) times the largest residual, is added to f_{n-1} to yield f_n. The search over the residual map is restricted to a region A termed the *Clean window.* The iteration terminates with an approximate solution f_N either when N equals some iteration limit $N_{\max}$, or when the peak residual (in absolute value) or the r.m.s. residual decreases to some given level.

To diminish any spurious high spatial frequency features in the solution, f_N is convolved with a narrow elliptical Gaussian function h, termed the *Clean beam.* Generally h is chosen by fitting to the central lobe of the dirty beam. Also, one generally adds the final residual map $g - b * f_N$ to the approximate solution $f_N * h$, in order to produce a final result, termed the *Clean map,* with a realistic-appearing level of noise. See *super-resolution.*

host computer — In the parasitic relationship of a computer program or program package, such as AIPS, to the computer on

which it runs, the latter is termed the *host computer.* Also, in the master–slave relationship of a computer to one of its peripheral devices, such as an array processor, the master may be termed the *host*.

hue — one of the three basic parameters (*hue*, *intensity*, and *saturation*) which may be used to describe the physical perception of the light that reaches one's eye. Hue, which is also termed *tint*, or simply *color*, refers to the dominant wavelength of the coloration, at a given location in an image or scene. The term also may be used to describe a multi-modal color spectrum—e.g., one speaks of a purple hue. Different spectral distributions of light, of identical intensity and saturation, are capable of producing identical retinal responses; these unique responses comprise the set of perceptible hues.

Color matching tests have established that there are three basic types of human retinal receptors, whose peak responses are to red, green, and blue light. These are the three *primary hues* used in additive color mixing—e.g., in digital image display. They may be used to produce all, or virtually all, of the perceptible hues.

See *C.I.E. chromaticity diagram*.

hybrid mapping algorithm — an algorithm for calibration of radio interferometer data which is essentially equivalent to the *self-calibration algorithm* (*q.v.*) (used in VLA data reduction), except in that it makes explicit use of the *closure phase* and *closure amplitude* relations, rather than explicit use of the relation $\tilde{V}_{ij} = g_i \overline{g}_j V_{ij}$ relating observed visibility to the product of the true visibility and a pair of *antenna/i.f. gains*. Hybrid mapping, which is used extensively in VLBI data reduction, is described in [A. C. S. Readhead *et al.*, Mapping radio sources with uncalibrated visibility data, *Nature*, **285** (1980) 137–140].

Either algorithm (assuming that one cares to make some distinction) can be applied to data obtained with connected- (e.g., the VLA) and non-connected-element interferometers (e.g., VLBI arrays). Any differences in the results produced by the two algorithms would be attributable primarily to differences in the effective weighting of the data (in particular, early implementations of both algorithms discarded data which could have been used to obtain over-determined solutions for the calibration parameters).

IIS — See *I^2S*.

image — in the context of AIPS, any finite-volume, linear, rectangular, or hyper-rectangular array of pixels; e.g., a digitized photograph, or a radio map. The term also is used (less technically) to refer to the *display* of data—e.g., a television picture of a radio map.

image catalog — in AIPS, a disk file containing data records describing the data stored on the TV display device *image planes.* These records are essentially identical in structure to the *header records* stored in the *catalog file.* The data in the image catalog furnish the information that is required for proper axis labeling, pixel value retrieval, etc.

image file — in AIPS, a *primary data file* whose content is an *image.*

image plane — a storage area within a TV display device, such as the *I^2S*, in which a full screen load of single word pixels is stored. A typical I^2S unit is equipped with four image planes, each 512 pixels × 512 pixels in area (each pixel is represented by eight bits). Often several image planes are used at one time—either for black-and-white or *pseudo-color* display of a large image, sections of which may occupy different image planes—or for *false color* or *true color* display of a smaller image, now using, say, three image planes—one to control each of the three electron guns (for red, green, or blue phosphor) in the TV display. Compare *graphics plane*.

image reconstruction — the attempted recovery of an *image* after it has undergone the distorting effects, the blurring, etc., produced by some physical measurement and recording device, such as a camera, a radio interferometer, or a tomography machine. The operation of many measurement devices can be adequately modeled by a linear Fredholm integral equation of the first kind. In the two-dimensional case, e.g., one assumes that the measurement $g(x, y)$ is related to the undistorted image $f(x, y)$ by the equation

$$g(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} K(x,y,x',y') f(x',y')\, dx'\, dy' + \epsilon(x,y)\,.$$

(Often it is convenient to use the more compact, operator notation, $g = \mathbf{K}f + \epsilon$.) The kernel K of the equation is called the *point spread function*, (*q.v.*). Measurement error and the error arising from any simplifying assumptions are lumped together into the $\epsilon(x, y)$ term. Some particularly well-behaved measurement systems can be adequately modeled by a simple convolution equation, in which case K is given by $K(x, y, x', y') = h(x - x', y - y')$. This is the case, e.g., when the VLA is used to observe a small 'unconfused' radio source; then g may be identified with the *dirty map* and h with the *dirty beam*. Or when K, considered as a function of (x, y), is given at each (x', y') by the *delay beam* for that position, the equation models the *bandwidth smearing effect* (*q.v.*); as the bandwidth $\to 0$, the convolution model again becomes valid.

Except in trivial cases, solution of the Fredholm equation always is an ill-posed problem. Mild conditions on K and f (the classical 'Picard conditions'—see F. Smithies [*Integral Equations*, Cambridge Univ. Pr., London, 1958]) ensure the existence of (non-unique) solutions when $\epsilon \equiv 0$. But, because of the effect of measurement noise, one usually does not seek an exact solution, but rather an approximate solution—one which fits the data to within the measurement errors. Uniqueness and regularity of the computed approximate solution are obtained by imposing such constraints as known *support*, non-negativity, and smoothness conditions. See *regularization method*. Also see H. C. Andrews and B. R. Hunt [*Digital Image Restoration*, Prentice–Hall, Englewood Cliffs, NJ, 1977] and *phaseless reconstruction*.

inputs file — in AIPS, a *text file*, whose contents may be displayed on the terminal screen of the interactive user, giving a summary of the *adverbs* relevant to a given *verb* or a given AIPS *task*.

instrumental polarization — any contamination of a polarization measurement by an instrument's response to an undesired polarization state. In radio interferometry, the instrumental polarization arises mainly from feed imperfections and from plumbing leaks between the feeds and the receiver front-ends. One tries to remove the instrumental polarization by applying corrections derived from observations of calibration sources whose polarization properties are known. Within AIPS, there is, at present, no facility for polarization calibration. The polarization calibration of VLA data normally takes place on the DEC–10 computer at the VLA. For more details, see Carl Bignell's Lecture No. 4 in the *1985 Summer School Proceedings*. See *beam squint*.

intensity — one of the three basic parameters (*hue*, *intensity*, and *saturation*) which may be used to describe the physical perception of color. Intensity is a measure of the energy of the spectral distribution, at a given point in an image or scene, weighted by the spectral response of the visual system. *Luminance* is the energy of the physical spectrum, but not weighted by the visual response. *Brightness* sometimes is used synonymously with either term.

See *C.I.E. chromaticity diagram*.

invisible distribution — in the context of radio interferometry, a function f (or a generalized function—or distribution) whose Fourier transform $\hat{f}$ vanishes everywhere that the interferometer pairs have sampled. This term was introduced by R. N. Bracewell and J. A. Roberts [Aerial smoothing in radio astronomy, *Austr. J. Phys.*, **7** (1954) 615–640]. Also see *principal solution*.

For an actual interferometer, there exist fewer physically plausible invisible distributions than for an idealized interferometer. This is because each visibility sample is not a point sample of $\hat{f}$, but rather some kind of local average. By the *Paley–Wiener theorem*, if $\hat{f}$ is nontrivial and vanishes in some open neighborhood, then f cannot be of *compact support*, and hence it may be considered implausible.

IPL — (Initial Program Load) same as *boot*.

isoplanaticity assumption — in the context of radio interferometry (the term is used too in optics), the assumption that over each element of an array all wavefronts arriving from different parts of the sky to which the interferometer pairs are sensitive are subject to identical atmospheric phase perturbations. A patch of sky over which the assumption is valid is referred to as an *isoplanatic patch*.

Approximate validity of the isoplanaticity assumption is a necessary condition for the success of calibration (*self-calibration*, in particular) of radio interferometer data (from an earth-based array) if one is to rely on a model incorporating time-varying *antenna/i.f. gains*, one per antenna, whose arguments (or phases) are to include the atmospheric phase corruption. However, see F. R. Schwab [Relaxing the isoplanatism assumption in self-calibration; applications to low-frequency radio interferometry, *Astron. J.*, **89** (1984) 1076–1081].

I^2S — (International Imaging Systems Models 70 and 75) a TV display device, capable of both black-and-white and color display, manufactured by the Stanford Technology Corporation. At an AIPS site typically it is equipped with four 512 pixel $\times$ 512 pixel eight-bit *image planes*, four one-bit *graphics planes*, a *trackball*, and sometimes an *ALU*. The eight-bit pixel representation (in the image planes) allows the intensity of each of the three electron gun beams to be set at any of 256 discrete levels. (Actually, 1024 levels can be used, because of an extra two bits of capability provided in the *transfer function* tables and the internal arithmetic unit.) An I^2S is attached to three of the NRAO's computers on which the AIPS system runs (the VLA and Charlottesville Vaxes).

line editor — a *text editor* (*q.v.*) which allows the modification of single lines or records within a text file, but one which does not allow the simultaneous modification of more than one line. *SOS* and *SEDIT* are both line editors. *Screen editors* (*q.v.*) are more versatile than line editors.

lobe rotator — same as *fringe rotator*, (*q.v.*).

loop gain — in the Högbom Clean algorithm, the fraction μ of the largest residual which is used in determining the amplitude, or flux, of a *Clean component*. Convergence can be achieved for μ in the range $(0, 2)$, but generally a small value, say $\mu = \frac{1}{10}$, is recommended, especially in dealing with extended sources. See *Högbom Clean algorithm*.

luminance — See *intensity*.

l_1 solution algorithm — See *self-calibration gain solution algorithm.*

l_2 solution algorithm — See *self-calibration gain solution algorithm.*

major cycle — In the *Clark Clean algorithm* (*q.v.*), a number of minor cycles, or inner iterations, followed by the computation by the FFT algorithm of the full residual map, comprise a major cycle.

map — an *image*, one or more of whose coordinate axes represents some spatial coordinate.

maximum entropy method — a *regularization method* (*q.v.*) for the numerical solution of ill-posed problems, given noisy data, in which the regularizing (or smoothing) term—which measures the roughness of the computed approximate solution $\tilde{f}$—is given by the negative of the Shannon entropy of $\tilde{f}$, $-H(\tilde{f})$: in the continuous case, letting A denote the domain of definition of $\tilde{f}$, $H(\tilde{f}) \equiv -\int_A \tilde{f}(x) \log \tilde{f}(x)\, dx$, where $\tilde{f}$ has been normalized so that $\int_A \tilde{f}(x)\, dx = 1$ (and $0 \log 0 \equiv 0$); and in the discrete case, $H(\tilde{f}) \equiv -\sum \tilde{f}(x_i) \log \tilde{f}(x_i)$, where $\tilde{f}$ has been normalized so that $\sum \tilde{f}(x_i) = 1$. The underlying philosophy of the method, espoused early on by Jaynes ("Jaynes' method of prior estimation", [E. T. Jaynes, Prior probabilities, *IEEE Trans. Syst. Sci. Cyb.*, **SSC-4** (1968) 227–241]) and by J. P. Burg at a 1967 meeting of the Society of Exploration Geophysicists, is that one is being "maximally noncommittal" in regard to the insufficiency of the data if one maximizes the entropy, and thus minimizes the "information content", of $\tilde{f}$, subject to the constraint that $\tilde{f}$ should agree with the given data.

For one-dimensional discrete convolution equations, with noiseless, regularly-spaced data, there exists a closed-form solution—for other cases, iterative methods are used, as with other forms of the regularization method.

Use of the method in radio astronomy was encouraged by J. G. Ables in 1972 in public lectures, and it now is in common use in radio interferometry (cf. [S. F. Gull and G. J. Daniell, Image reconstruction from incomplete and noisy data, *Nature*, **272** (1978) 686–690]). Non-negativity of the computed solution is a natural by-product of the method. For reconstruction of polarized brightness distributions in interferometry (Stokes' Q, U, and V), which, unlike the total intensity, may assume negative values, Ponsonby has derived an appropriate generalization of the method [J. E. B. Ponsonby, An entropy measure for partially polarized radiation…, *Mon. Not. R. Astr. Soc.*, **163** (1973) 369–380]. See *Variational Method*.

memory page — See *virtual memory page*.

memory paging — same as *virtual memory page swapping*.

memory thrashing — an excessive amount of *virtual memory page swapping* (*q.v.*) on a computer (such as the Vax) with a virtual memory operating system. A condition of memory thrashing is likely to occur whenever too many programs with large memory requirements are active (a single program with excessive memory requirements also can cause memory thrashing).

message file — in AIPS, a *text file* containing progress report messages generated during the execution of AIPS *tasks* and also containing a chronicle of the user's interaction (via *verb* commands) with AIPS. Each AIPS user is assigned a message file, the contents of which may be printed out, typed upon a terminal display screen, or emptied—at will—by invoking the appropriate verb command. See *AIPS monitor*.

message terminal — same as *AIPS monitor*.

minor cycle — in the *Clark Clean algorithm* (*q.v.*), an inner iteration, in which the peak residual over a sub-region (the *Clean window*) of the full residual map is found and is used to obtain the next successive iterate. Compare *major cycle.*

microcode — See *array processor microcode*.

monitor — See *AIPS monitor* or *Conrac monitor.*

$m \times n$ map — The convention adopted for AIPS is opposite the standard matrix algebra terminology: whereas an $m \times n$

matrix is comprised of m rows and n columns, an $m \times n$ *map* or *image* in AIPS has, in the usual display format, m pixels along the horizontal axis (usually termed the x-axis) and n pixels along the vertical axis (usually termed the y-axis). Moreover, pixels of a two-dimensional map in the usual display format are numbered from the bottom left-hand corner: the pixel location specified by the ordered pair (i, j) is in column number i and row number j, counting from the bottom left. In other than two-dimensional "images", the $(1, \ldots, 1)$ pixel is also said to be at the "bottom left corner" (BLC), just as in the two-dimensional case. See *data cube*, *pixel coordinates*, and *coordinate reference pixel*.

MX — See *"battery-powered" Clean algorithm*.

natural weighting — See *uniform weighting*.

negative bowl artifact — See *zero-spacing flux*.

non-closing offset — See *correlator offset*.

Nyquist sampling rate — the slowest rate of sampling which, according to the *Shannon sampling theorem* (*q.v.*), would allow a band-limited function $f(t)$ to be recovered via the *Shannon series*. If the smallest symmetric interval which contains the *support* of the Fourier transform of f is the interval $[-a, a]$, then the Nyquist sampling rate for f is $2a$; i.e., the interval between samples (the *sampling period*) must be less than the *reciprocal bandwidth* $1/2a$. The terms *oversampling* and *undersampling* refer to sampling at rates faster or slower than the Nyquist rate. The difference between f and the Shannon series formed from too coarsely spaced samples is called *aliasing*.

operating system —

page — See *virtual memory page* and *terminal page*.

page swapping — See *virtual memory page swapping*.

Paley–Wiener theorem — The classical Paley–Wiener theorem says that a square-integrable complex-valued function $\hat{f}$, defined over the real line, can be extended off the real line as an entire function of exponential type $\leq 2\pi a$ if and only if $f(x) \equiv 0$ for $|x| > a$ —i.e., iff $\hat{f}$ is band-limited to $[-a, a]$ (here $\hat{}$ denotes Fourier transform). (An everywhere-analytic function $g(z)$ is said to be of exponential type $\leq A$ if $\exists c$ such that, for all z, $|g(z)| \leq ce^{A|z|}$.) For a derivation, see H. Dym and H. P. McKean [*Fourier Series and Integrals*, Academic Press, 1972]. The *Shannon series* is a means of extending $\hat{f}$ to $\mathbf{C}$. The extension of the Paley–Wiener theorem to the case of generalized functions (to tempered distributions) is called the Paley–Wiener–Schwartz theorem.

The Fourier transform $\hat{f} : \mathbf{R}^n \rightarrow \mathbf{C}$ of a function f with support in a given n-dimensional convex compact set K can be analytically extended to all of $\mathbf{C}^n$. Growth properties on $\hat{f}$ which are sufficient in order for the converse to hold are given by K. T. Smith, D. C. Solomon, and S. L. Wagner [Practical and mathematical aspects of the problem of reconstructing objects from radiographs, *Bull. Amer. Math. Soc.*, **83** (1977) 1227–1270] (in addition to the classical version of the multi-dimensional Paley–Wiener theorem, for rectangular K, they give versions with tighter growth bounds, and for arbitrary convex K). Smith *et al.* use the Paley–Wiener theorems to establish indeterminacy theorems for tomographic reconstruction. Their results are also relevant to Fourier synthesis, because of the connection between the two-dimensional Fourier transform and the one-dimensional Radon transform. The Paley–Wiener theorems have also been used in establishing results on the problem of *phaseless reconstruction* (*q.v.*) and in proving the convergence of constrained *Gerchberg–Saxton*-type algorithms (see A. Lent and H. Tuy [An iterative method for the extrapolation of band-limited functions, J. Math. Anal. Appl., **83** (1981) 554–565]).

phaseless reconstruction — the reconstruction of an image f (see *image reconstruction*) from knowledge of (only) the magnitude $|\hat{f}|$ of the Fourier transform of f (and usually from only partial knowledge of $|\hat{f}|$). Phaseless reconstruction has been considered for the NRAO's proposed millimeter wave interferometer array [T. J. Cornwell, Imaging of weak sources with compact arrays, NRAO Millimeter Array Memo. No. 12]. Recent results on phaseless reconstruction appear in the JOSA Feature Issue on Signal Recovery [*J. Opt. Soc. Am.*, **73** No. 11 (Nov. 1983)]. Also see the papers by J. R. Fienup and by R. H. T. Bates *et al.* in the *1983 Sydney Conference Proceedings*.

phase tracking center — same as *visibility phase tracking center*, (*q.v.*).

physical memory — core or semiconductor memory within a computer (as opposed to slower memory—virtual memory, disk storage, magnetic tape footage, etc.). A typical Vax is equipped with a physical memory 3–4 megabytes in size.

pillbox — See *cell-averaging*.

pixel — (*pic*ture *el*ement) an element of a digitized image (or of a map). A pixel is characterized by its position in the image and by its numerical value. See $m \times n$ *map*, *coordinate reference pixel*, and *pixel coordinates*.

pixel coordinates — in an AIPS *image file*, the *pixels* are numbered consecutively, beginning with $(1, \ldots, 1)$ at the bottom left corner (*BLC*) of the image. See *coordinate reference pixel* and $m \times n$ *map*.

plot file — an AIPS extension file containing plotting information, in the form of the commands which are necessary in order for a line drawing peripheral device, such as a Calcomp or other pen plotter, a green screen, or an electrostatic printer/plotter, to generate a plot.

point source response — same as *point spread function*.

points per beam — in a digitized radio map, the characteristic width, somehow defined, of the major lobe of the *beam* pattern, or *point spread function*, divided by the *pixel* separation. Ordinarily the number of points per beam is calculated by measuring the narrowest diameter of the 50% contour level of the major lobe of the beam. To avoid excessively severe discretization error, deconvolution algorithms such as the *Högbom Clean algorithm* and the *maximum entropy method* require, as a rule-of-thumb, at least three (and preferably 4–5) points per beam.

point spread function — (PSF) 1. the response of a system or an instrument to an impulsive, or point source, input. 2. in radio interferometry, the response of the instrument to a point, or unresolved, radio source—a fancy term for *beam*. Ignoring instrumental effects, such as finite bandwidth and finite integration time, the response does not depend upon the displacement of the source away from the *visibility phase tracking center*—hence the term *space-invariant PSF (SIVPSF)*, and the contrary term *space-variant PSF (SVPSF)*.

A so-called linear space invariant measurement system (i.e., a linear system with an SIVPSF) is equivalently described as a system which can be modeled by a convolution equation; a linear space-variant measurement system is modeled by a more general linear Fredholm integral equation of the first kind. See *image reconstruction*.

POPS — (People-Oriented Parsing System) the parser, or command interpreter, embedded within the AIPS program; that part of the AIPS program which attempts to interpret the user's commands (*POPS symbols*) and then initiate the appropriate reaction. POPS is used in other astronomical data reduction programs at the NRAO: in Condare, TPOWER/SPOWER, and the Tucson 12 m single-dish packages.

POPS procedure — See *POPS symbols.*

POPS symbols — The AIPS user's primary means of communicating his wishes to AIPS is by typing commands, termed *POPS symbols,* at the keyboard of a computer terminal. There are four classes of POPS symbols: *adverb, verb, pseudo-verb,* and *procedure.* An *adverb* is a *symbol* representing the storage area for a datum or for data that are used to control the action of verbs, tasks, and procedures; that is to say that the adverb symbols are used to set *control parameters.* A *verb* is a symbol which causes POPS (or AIPS) to initiate some action after POPS has finished interpreting, or compiling, the command line typed at the computer terminal. A *pseudo-verb* is a symbol which suspends, temporarily, the normal parsing of an input line and which causes some action to take place while the line is being compiled, and, possibly, after compilation. A *procedure* is a symbol representing a pre-compiled sequence of POPS symbols. Also see *task.*

primary beam correction — in radio interferometry, the multiplicative correction of a radio *map* by the reciprocal of an average of the power patterns of the array elements. Measurements of the primary beam parameters of the 25 m VLA elements are given by Peter Napier and Arnold Rots in the memorandum [VLA primary beam parameters, VLA Test Memo. No. 134, Feb. 1982]. There an average power pattern and its reciprocal are approximated by radial functions, polynomials in the distance from the pointing position. The AIPS task PBCOR is used to apply this correction to VLA maps. The appropriate correction at large distances from the pointing position is not well-determined, thus PBCOR "blanks" the map pixel values beyond a certain radius (see *blanked pixel*).

primary data file — in AIPS, either a *u-v data file*, containing measurements of the visibility function of a radio source, or an *image file*, containing a digitized image or a radio map. Compare *extension file*.

principal solution — in the context of radio interferometry, the inverse Fourier transform of the *u-v measurement distribution*; i.e., the *dirty map* (*q.v.*) in sense 1 of the definition. This term was introduced by R. N. Bracewell and J. A. Roberts [Aerial smoothing in radio astronomy, *Austr. J. Phys.*, **7** (1954) 615–640]. Except in the trivial case, the principal solution to the mapping problem in interferometry is a physically implausible solution, because the principal solution has not the property of *compact support*.

An *invisible distribution* (*q.v.*) added to the principal solution yields another solution—i.e., another brightness distribution which is consistent with the observations.

procedure — See *POPS symbols*.

prolate spheroidal wave function — an eigenfunction of the finite, or truncated, Fourier transform—more precisely, for given c, one of the countably many solutions of the integral equation

$$\nu f(\eta) = \int_{-1}^{1} e^{ic\eta t} f(t)\, dt\,;$$

equivalently, a solution of the differential equation $(1-\eta^2)f'' - 2\eta f' + (b - c^2\eta^2)f = 0$; or, equivalently, a solution of the wave equation in a system of prolate spheroidal coordinates. The eigenfunction of the above equation associated with the largest eigenvalue ν is termed the 0-order solution.

If we want a gridding convolution function C, of *support width* equal to the width of m grid cells, that is optimal in the sense that its Fourier transform $\hat{C}$ has the property that the concentration ratio

$$\frac{\iint_{\rm map} |\hat{C}(x,y)|^2\, dx\, dy}{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} |\hat{C}(x,y)|^2\, dx\, dy}$$

is maximized, then C is the separable product of two 0-order prolate spheroidal wave functions, with $c = \pi m/2$. See *gridding convolution function* and *spheroidal function*.

prompt character — a character (often the dollar sign "$" or the greater-than sign ">") which the computer program or the operating system prints on the terminal screen of the interactive user in order to prompt, or invite, a typed response from the user. The AIPS program's standard prompt character is the greater-than sign, and on the Vaxes at the NRAO the operating system's prompt character is the dollar sign. On most UNIX systems, the prompt character is the percent sign. Thus, most commands (or *POPS symbols*) peculiar to AIPS must be typed on a line beginning with the >-character, and any command to the operating system, such as the command to mount a tape, must be typed on a line beginning with the $- or %-character.

When operating in some lesser-used, special modes, AIPS employs other prompt characters: ":" for procedure building, ";" for procedure editing, "!" for entry of gripes, "<" for batch file preparation, and "#" for parameter reading.

Prussian helmet Clean algorithm — a modified version of the Högbom Clean algorithm, devised by Tim Cornwell. The idea is to drive the Clean algorithm toward an approximate solution f of minimal Euclidean norm—i.e., to find an f consistent with the data, confined to the Clean window, comprised of a small number of point components, and such that $\iint_{\substack{\rm Clean\\ \rm window}} [f(x,y)]^2\, dx\, dy$ is minimized. This is accomplished by adding a δ-function of amplitude ω, centered at the origin, to the dirty beam, and then just proceeding as normal with the Clean algorithm. Proper choice of ω depends on the distribution of measurement errors. See [T. J. Cornwell, A method of stabilizing the Clean algorithm, *Astron. Astrophys.*, **121** (1983) 281–285]. A provision for this modification is incorporated in the AIPS tasks APCLN and MX. See *regularization method*.

pseudo-AP — See *pseudo-array processor*.

pseudo-array processor — in AIPS, the term which is applied to a collection of Fortran subroutines which may be used to emulate the operation of an FPS Model AP–120B array processor. At those AIPS sites which do not have an array processor, the AIPS tasks which normally would make use of an array processor use the pseudo-array processor subroutines instead. See *array processor*.

pseudo-color display — In digital imagery, a *pseudo-color* display is one which is derived from a single real-valued function $f(x,y)$ and a mapping $\mathbf{R}^1 \rightarrow \mathbf{R}^3$ that controls the *hue*, *intensity*, and *saturation*—or, equivalently, the proportions in an additive mixture of three primary hues—of the coloration at each *pixel* coordinate (x,y) of the display, according to the value of $f(x,y)$. A pseudo-color display might be used, for example, to represent measurements of the intensity of the radio continuum flux density of a source.

Compare *false color display* and see *color contour display*.

pseudo-continuum u-v data file — in VLA spectral line data reduction, a *u-v data file* containing the visibility measurements from a small number of spectral line channels, recorded in the same format as continuum visibility data. The purpose is to enable the use, for spectral line data analysis, of programs originally intended only to handle continuum data reduction.

pseudo-verb — See *POPS symbols.*

PSF — See *point spread function*.

Q-routine — in AIPS, a primitive level subroutine designed to function on a particular manufacturer's production model of an *array processor*. A goal of the AIPS project is to construct

libraries of Q-routines—one library appropriate to each model of array processor which might be used in conjunction with AIPS—with identical names, argument lists, and functionality. Existing Q-routines emulate the standard library of Floating Point Systems, Inc.'s, model *AP–120B array processor*.

quick boot — an abbreviated boot procedure. See *boot*.

RANCID — (Real (or Radio) Astronomical Numerical Computation and Imaging Device) the name by which the AIPS data reduction system formerly was known.

re-boot — Having booted once already, one *re-boots*. See *boot*.

regularization method — in the numerical solution of ill-posed problems, given noisy data, a method in which the original problem is converted into a well-posed problem by requiring of the solution to the modified problem (which now is an approximate solution to the original problem) that it satisfy some smoothness constraint. The prototypical ill-posed problem has the form $Kf = g + \epsilon$, where K is a known linear integral operator (e.g., a convolution operator), where $g + \epsilon$, which is given, represents some noisy measurement, and where f is unknown. In the context of radio interferometry, one may take $g+\epsilon$ to be the *dirty map* and K to be the operator which convolves the "true" radio source brightness distribution f with the *dirty beam*. Now, denoting our approximate solution to the ill-posed problem by $\tilde{f}$, $\tilde{f}$ is found by minimizing the expression

$$(1-\lambda)\|g - K\tilde{f}\|^2 + \lambda S(\tilde{f})\,,$$

for some given choice of the *regularization parameter* λ, $0 < \lambda < 1$. $\|g - K\tilde{f}\|^2$ is the mean squared residual (occasionally some other measurement of the error is used), and $S(\tilde{f})$ is a measure of the roughness of the computed solution—say, some power of a norm or seminorm of $\tilde{f}$, or a similar quantity, such as the negative of the (Shannon) entropy of $\tilde{f}$.

Proper choice of λ must be based on statistical considerations which depend on the distribution of measurement errors; often, one chooses λ in order achieve an *a priori* reasonable value of the mean squared residual. The *maximum entropy method*, *Tikhonov regularization*, and the *Prussian helmet Clean algorithm* are special cases of the regularization method. Appropriate choice of S is discussed by J. Cullum [The effective choice of the smoothing norm in regularization, *Math. Comp.*, **33** (1979) 149–170], and the choice of S and λ, by a statistical method known as "cross validation", is described by G. Wahba [Practical approximate solutions to linear operator equations when the data are noisy, *SIAM J. Numer. Anal.*, **14** (1977) 651–677]. Often, some Sobolev norm is chosen for S.

Usually, in addition to the smoothness constraint, f is assumed to be of known, *compact support*. Other constraints, such as non-negativity, may be included as well. In the case in which the data are exact—i.e., when $\epsilon = 0$, so that $g = Kf$—one may obtain the regularized solution corresponding to $\lambda = 0$ as the limit of regularized solutions $\tilde{f}_\lambda$ as $\lambda \to 0$. See *Variational Method*. Also see D. M. Titterington [General structure of regularization procedures in image reconstruction, *Astron. Astrophys.*, **144** (1985) 381–387].

regularization parameter — in the *regularization method* (*q.v.*) for the solution of ill-posed problems, a smoothing parameter λ, $0 < \lambda < 1$, which controls the trade-off between an error term, measuring agreement of the computed solution $\tilde{f}$ with the given data, and a term $S(\tilde{f})$, which measures the roughness of $\tilde{f}$. I.e., λ controls the amount of "regularization". See *super-resolution*.

re-IPL — same as *re-boot*.

residual delay — Expressing the *antenna/i.f. phase*, ψ_k, for antenna k of a VLBI array as a function of frequency as well as of time, the residual delay on the i–j baseline at (t_0, ν_0) is given by $\tau_{ij} \equiv \left.\frac{\partial(\psi_i - \psi_j + \phi_{ij})}{\partial\nu}\right|_{(t_0,\nu_0)}$, where ϕ_{ij} denotes the visibility phase on the i–j baseline. (The partial w.r.t. t is called the *residual fringe rate*.) Usually the major contributor to residual delay is the difference in the station clock errors. The residual delay is a group delay, rather than a phase delay. It is termed residual because it is assumed that geometric effects have already been compensated for.

The "antenna components" of τ_{ij}, namely $\tau_k \equiv \left.\frac{\partial\psi_k}{\partial\nu}\right|_{(t_0,\nu_0)}$, are called the *antenna residual delays*. They are among the solution parameters of the global fringe fitting algorithm for VLBI. See *residual fringe rate* and *global fringe fitting algorithm*.

residual fringe rate — Expressing the *antenna/i.f. phase*, ψ_k, for antenna k of a VLBI array as a function of frequency as well as of time, the residual fringe rate on the i–j baseline at (t_0, ν_0) is given by $r_{ij} \equiv \left.\frac{\partial(\psi_i - \psi_j + \phi_{ij})}{\partial t}\right|_{(t_0,\nu_0)}$, where ϕ_{ij} denotes the visibility phase on the i–j baseline. (The partial w.r.t. ν is called the *residual delay*.) Usually the major contributor to residual fringe rate is the drift of the station clocks.

The "antenna components" of r_{ij}, namely $r_k \equiv \left.\frac{\partial\psi_k}{\partial t}\right|_{(t_0,\nu_0)}$, are called the *antenna residual fringe rates*. They are among the solution parameters of the global fringe fitting algorithm for VLBI. See *residual delay* and *global fringe fitting algorithm*.

resolution — See *spatial resolution*.

restoring beam — same as *Clean beam*.

roam — See *TV roam*.

run file — in AIPS, a *text file* written by an AIPS user and containing a sequence of AIPS commands (*POPS symbols*). Run files are useful for the storage of strings of commands which one might wish to execute repeatedly (in particular, for the storage of lengthy *procedures*). The run files for all users at a particular AIPS installation are stored in a common area. These files ordinarily are created through use of one of the standard *text editors* of AIPS' host computer.

sampling theorem — See *Shannon sampling theorem*.

saturation — one of the three basic parameters (*hue*, *intensity*, and *saturation*) which may be used to describe the physical perception of color. Saturation is a measure of the (perceived) narrowness of the color spectrum, or the difference of the hue from a gray of the same intensity. Neutral gray—or a "white" spectrum—is termed 0% saturated, and a monochromatic spectrum is termed 100% saturated.

See *C.I.E. chromaticity diagram*.

scratch — 1. The act of deleting a data file—i.e., surrendering the storage medium space which that file occupies—is termed *scratching* the data file. Use of the term *delete* may be preferable, but *scratch* is more common among AIPS users. One who is about to delete a data file may wish first to create a back-up copy. See *back-up*. 2. an adjective meaning *temporary*, as in *scratch file*.

In AIPS a primary data file and all of its associated extension files can be deleted by means of the verb *ZAP*.

scratch file — a data file intended for temporary storage (esp., of data which represent intermediate results—i.e., *scratch*work). Many of the AIPS tasks use scratch files; the necessary scratch files are created and destroyed automatically by the tasks. However, when an AIPS task *crashes*, sometimes a scratch file remains.

screen editor — a *text editor* (*q.v.*) which, unlike a *line editor*, allows the simultaneous modification of more than one line or record within a text file. For example, a mechanism to facilitate alignment of margins often is incorporated by a screen editor. *EDT*, *EVE*, *vi* and *EMACS* are screen editors.

scroll — See *terminal scroll* and *TV scroll*.

self-calibration algorithm — Many of the systematic errors affecting interferometer visibility measurements may be assumed to be multiplicative and ascribable to individual array elements. That is, in an n element array, the observations on the $n(n-1)/2$ baselines are afflicted by n sources of systematic error, the so-called *antenna/i.f. gains* $g_k(t)$. Given a rough estimate of the true source visibility, a model obtained, say, by mapping and Cleaning roughly calibrated data, one may solve for the unknown gains—and it is not unreasonable to do so, because there are $(n-1)/2$ times more observations than antenna gains. The number of degrees of freedom can be held further in check by assuming that the $g_k(t)$ are slowly-varying or that they are of unit modulus (i.e., that no amplitude errors are present), or by designing an array with redundant spacings.

Having once solved for the unknown g_k, one may correct the data, make another map, and repeat the process. This iterative scheme, which yields successive approximations to the true radio source brightness distribution, is known as *self-calibration*. Self-calibration is essentially identical to the technique of *hybrid mapping*, which is widely used in VLBI. See *self-calibration gain solution algorithm*; also see Tim Cornwell and Ed Fomalont's Lecture No. 9 in the *Third NRAO Synthesis Imaging Summer School* and the review paper by T. J. Pearson and A. C. S. Readhead [Image formation by self-calibration in radio astronomy, *Ann. Rev. Astron. Astrophys.*, **22** (1984) 97–130].

self-calibration gain solution algorithm — In self-calibration, the unknown *antenna/i.f. gains* $g_k(t)$ may be approximated by minimizing a functional $S(g_1,\ldots,g_n)$ given by a weighted discrete l^p norm of the residuals:

$$S(\mathbf{g}) = \left(\sum_{1\le i<j\le n} w_{ij} \left| \tilde{V}_{ij} - g_i \overline{g}_j V_{ij} \right|^p \right)^{1/p} .$$

Here $\tilde{V}_{ij}$ is the visibility measurement obtained on the i–j baseline (at a given instant), V_{ij} is the corresponding *model* visibility, and w_{ij} is a suitably chosen weight. Usually the g_k may be assumed not to vary too rapidly with time, so that one may minimize, instead, the functional

$$S(\mathbf{g}) = \left(\sum_{1\le i<j\le n} w_{ij} \left| \left\langle \tilde{V}_{ij}/V_{ij} \right\rangle - g_i \overline{g}_j \right|^p \right)^{1/p} ,$$

where $\left\langle \tilde{V}_{ij}/V_{ij} \right\rangle$ is the time-average of the ratio of observed visibility to model visibility, over a time period during which the g_k may be assumed constant.

The AIPS implementation allows the choices $p=1$ and $p=2$. Choosing $p=2$ yields the least-squares solution for $\mathbf{g}$. When one chooses $p=1$, so that a weighted sum of the moduli of the residuals is minimized, the computed gain solutions are less influenced by wild data points, but there is some loss of statistical efficiency—i.e., the least-squares solutions are superior when the distribution of measurement errors is well-behaved. (Probably the choice $p \simeq 1.2$ would offer a better compromise between efficiency and robustness). See [F. R. Schwab, Robust solution for antenna gains, VLA Scientific Memo. No. 136] for further details.

One may wish to solve only for the *antenna/i.f. phases* $\psi_k(t)$ rather than for the g_k if, for example, atmospheric phase corruption is believed to be the dominant source of systematic error. In this case, one minimizes

$$S(\Psi) = \left(\sum_{1\le j<k\le n} w_{jk} \left| \tilde{V}_{jk} - e^{i(\psi_j-\psi_k)} V_{jk} \right|^p \right)^{1/p} ,$$

or the version thereof incorporating time-averages.

Cornwell and Wilkinson [A new method for making maps with unstable radio interferometers, *Mon. Not. R. Astr. Soc.*, **196** (1981) 1067–1086] suggest adding to S terms which arise by assuming prior distributions for the g_k; these "penalty terms" would be chosen so as to increase in magnitude as the solution parameter deviates from a prior mean which one might take, say, as the running mean of previous gain solutions. The widths of the prior distributions could be based on empirical knowledge of the behavior of the array elements. Such a modification can be useful when the array is composed of antenna elements of differing collecting area. This modification is used in order to constrain the moduli of the computed gains in one version of the AIPS task for self-calibration which is used primarily for VLBI data reduction (VSCAL).

Shannon sampling theorem — Suppose the complex-valued function f of the real variable t to be square-integrable, and assume that f is band-limited; i.e., that its Fourier transform $\hat{f}(x) = \int_{-\infty}^{\infty} f(t) e^{2\pi i x t}\, dt \equiv 0$ for $|x| > a$. Then f is completely determined by its values at the discrete set of sampling points $n/2a$, $n = 0, 1, 2, \ldots$, and f can be recovered via the *Shannon series* (also called the cardinal series)

$$f(t) = \sum_{n=-\infty}^{\infty} f\left(\frac{n}{2a}\right) \frac{\sin 2\pi a(t-n/2a)}{2\pi a(t-n/2a)} .$$

The series converges both uniformly and in the mean-square sense.

The Shannon series can be derived by expanding $\hat{f}$ in a Fourier series, and then applying Fourier inversion—or it can be derived from the classical Poisson summation formula. It is sometimes referred to as Whittaker's cardinal interpolation formula or the Whittaker–Shannon sampling series, having first been studied in detail by E. T. Whittaker in 1915 and later introduced into the literature of communications engineering by Shannon in 1949. By the *Paley–Wiener theorem*, since f is band-limited, it can be analytically extended from the real line to the full complex plane, as an entire function of slow growth. The Shannon series, which converges for complex as well as real t, is one means of doing so. Whittaker referred to the series as "a function of royal blood in the family of entire functions, whose distinguished properties separate it from its bourgeois brethren."

Suppose that $f(t)$ is "small" for $|t| > b$ (no nontrivial signal is both band-limited and time-limited). Then, assuming that b is integral, the number of terms in the Shannon series that really matter is $4ab$. This suggests that the space of "essentially band-limited" and "essentially time-limited" signals has dimension equal to the *time-bandwidth product* $4ab$. The precise sense in which this is so, together with a discussion of the *prolate spheroidal wave functions* (*q.v.*), which are relevant to the problem, is described by H. Dym and H. P. McKean [*Fourier Series and Integrals*, Academic Press, New York, 1972] and by David Slepian [Some comments on Fourier analysis, uncertainty and modeling, *SIAM Rev.*, **25** (1983) 379–393].

The multi-dimensional extension of the sampling theorem to rectangles implies that if an "unconfused" radio source $f(x,y)$ is confined to a small region of sky $|x| < x_0$, $|y| < y_0$ (radians), then it can be reconstructed unambiguously from a discrete set

of visibility samples $\hat{f}(m\Delta u, n\Delta v)$, $m, n = 0, 1, 2, \ldots$, with $\Delta u = 1/2x_0$ and $\Delta v = 1/2y_0$ wavelengths. See *cellsize* and *Nyquist sampling rate*. Other useful extensions of the sampling theorem—for example, to various multi-dimensional sampling configurations (e.g., 2–D hexagonal sampling lattices), to the case of stochastically jittered sampling, to derivative sampling (e.g., in 1–D, f can be recovered from samples of f and its derivatives through order r taken at intervals $(r+1)\frac{n}{2a}$), etc.—and sampling theorems for functions whose transforms of other than Fourier type are of *compact support*—are described in survey articles by A. J. Jerri [The Shannon sampling theorem—its various extensions and applications: a tutorial review, *Proc. IEEE*, **65** (1977) 1565–1596] and J. R. Higgins [Five short stories about the cardinal series, *Bull. (New Ser.) Amer. Math. Soc.*, **12** (1985) 45–89].

Shannon series — See *Shannon sampling theorem*.

shed — See *sub-task*.

SIVPSF — See *point spread function*.

slice — a one-dimensional cut across an *image*. E.g., the slice of a two-dimensional image f which passes through (x_0, y_0) and has orientation angle ϕ is the *sub-image* h given by $h(t) = f(x_0 + t\cos\phi, y_0 + t\sin\phi)$. In AIPS, a slice may be excised from an image by issuing the *verb* command SLICE. Since AIPS deals only with digitized images, the program must interpolate to obtain data along the cut, except when the slice is taken along a row or column of the image.

slice file — in AIPS, an *extension file*, associated with an *image file*, in which a digitized *slice* (*q.v.*), or one-dimensional sub-image, of the primary image is stored. In order to display a slice, one may issue the *verb* command SL2PL, which causes AIPS to read the contents of a slice file and generate a *plot file*.

snapshot — in earth-rotation aperture synthesis interferometry, an observation which is of such short duration that Earth's motion does not significantly enhance the *u-v coverage*, or a *map* derived from such a brief observation. Compare *full-synthesis map*.

For a thorough discussion of the use of the VLA in snapshot mode, see §5 of A. H. Bridle's Lecture No. 16 in the *1985 Summer School Proceedings*.

software mount — a computer's reaction to the issuing of a command to it informing it that the *hardware mount* of some external storage module, such as a disk pack or a reel of magnetic tape, has occurred, and that the computer should open the channel of access to this module. See *hardware mount*.

sort order — the ordering of visibility measurements within a *u-v data file*. *Time-baseline order* is convenient for purposes of calibration, *baseline-time order* for data display, and so-called *x-y order* for gridding and subsequent mapping.

source editor — same as *text editor*. (Formerly, computers were used mainly for numerical computations and text editors primarily for the editing of program source code—hence the name *source editor*).

spatial resolution — In digital image analysis, this term refers rather imprecisely to the minimum size of details which can be discerned. The spatial resolution is determined by three factors: the inherent indeterminacy of whatever *image reconstruction* problem underlies the method by which the image was produced (and the properties of the image reconstruction *algorithm* which produced the image); the measurement noise; and the *pixel* size—i.e., the size of the squares or the rectangles comprising the reconstruction matrix.

In radio interferometry, the inherent spatial resolution goes roughly in inverse proportion to the physical size scale D of the array (measured in wavelengths). For observations at a wavelength λ, the inherent spatial resolution, with a filled aperture, is essentially λ/D radians. However, with a synthesis array with large gaps in the *u-v coverage*, the effective resolution is somewhat coarser. Often, some measure of the spread of the central lobe of the *dirty beam* (say, the FWHM) is quoted as the spatial resolution. However, some reconstruction methods (e.g., the *regularization methods*) produce images in which the resolution of bright features may be much finer than that of dim features. This property of regularization methods may be viewed as either good or bad: S/N dependent spatial resolution complicates the interpretation of an image, but, on the other hand, one may gain additional contrast resolution—i.e., low surface-brightness features may become more readily discernible. An honest statement concerning the spatial resolution of an image must be based upon empirical knowledge of the reconstruction method that was used. See *super-resolution*.

spawn — See *sub-task*.

spheroidal function — an eigenfunction $\psi_{\alpha n}$ of a finite, weighted-kernel Fourier transform—more precisely, for given c and given $\alpha > -1$, one of the countably many solutions of the integral equation

$$\nu f(\eta) = \int_{-1}^{1} e^{ic\eta t}(1-t^2)^{\alpha} f(t)\, dt\,;$$

equivalently, a solution of the differential equation $(1-\eta^2)f'' - 2(\alpha+1)\eta f' + (b - c^2\eta^2)f = 0$. The eigenfunction $\psi_{\alpha 0}$ of the equation above associated with the largest eigenvalue ν is termed the 0-order solution. The choice $\alpha = 0$ of weighting exponent yields the family $\{\, \psi_{0n} \mid n = 0, 1, 2, \ldots \,\}$ of prolate spheroidal wave functions.

Weighted 0-order spheroidal functions $(1-\eta^2)^{\alpha}\psi_{\alpha 0}$ are optimal gridding convolution functions in the same sense that the *prolate spheroidal wave functions* (*q.v.*) are optimal, except that now the weighted concentration ratio

$$\frac{\iint_{\rm map} |\hat{C}(x,y)|^2 (1-(2x\Delta u)^2)^{\alpha}(1-(2y\Delta v)^2)^{\alpha}\, dx\, dy}{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} |\hat{C}(x,y)|^2 |1-(2x\Delta u)^2|^{\alpha} |1-(2y\Delta v)^2|^{\alpha}\, dx\, dy}$$

is maximized (see the paper by F. R. Schwab in the *1983 Sydney Conference Proceedings*). The weighting exponent α is used to trade off the effectiveness of the aliasing suppression at the edge of the field of view, against that in the central region of the map. The choice $\alpha = 1$, with a *support width* of six u-v grid cells, yields an effective gridding convolution function, emphasizing aliasing suppression in the central region of the map; this function, ψ_{10}, with $c = 3\pi$, is the default function used in the AIPS mapping program. See *gridding convolution function*.

Stokes' parameters — the four coordinates relative to a particular basis for the representation of the polarization state of an electromagnetic wave propagating through space. Consider a wave propagating along the z-direction in a right-handed (x, y, z) Cartesian coordinate system. At a fixed point in space, let the instantaneous components of the electric field vector, in the x- and y-directions, be denoted by $E_x(t)$ and $E_y(t)$, respectively; and assume them to be stationary (in the weak sense, and square-integrable) stochastic processes. Form the matrix

$$S = \begin{pmatrix} \widehat{\langle E_x(t)\overline{E}_x(t+\tau)\rangle} & \widehat{\langle E_x(t)\overline{E}_y(t+\tau)\rangle} \\ \widehat{\langle E_y(t)\overline{E}_x(t+\tau)\rangle} & \widehat{\langle E_y(t)\overline{E}_y(t+\tau)\rangle} \end{pmatrix}.$$

Here, the bracketed expressions are expectation values, or correlation functions, in the lag variable τ, and $\widehat{\ }$ denotes Fourier transform with respect to τ. Thus each element of S is a function of frequency ν. S is Hermitian (conjugate symmetric), owing to the stochasticity assumptions. The three Pauli spin matrices, together with the 2×2 identity matrix, form a basis for the algebra of 2×2 Hermitian matrices; i.e., each such matrix S can be represented in the form

$$S(\nu) = \sigma_1(\nu) \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \sigma_2(\nu) \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} + \sigma_3(\nu) \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} + \sigma_4(\nu) \begin{pmatrix} 0 & i \\ -i & 0 \end{pmatrix} .$$

The four (real) coefficients, $\sigma_1, \ldots, \sigma_4$, of the representation of S in this basis are called Stokes' parameters. They commonly are denoted by $I(\nu)$, $Q(\nu)$, $U(\nu)$, and $V(\nu)$, respectively. In other words,

$$S(\nu) = \begin{pmatrix} I(\nu) + Q(\nu) & U(\nu) + iV(\nu) \\ U(\nu) - iV(\nu) & I(\nu) - Q(\nu) \end{pmatrix} ,$$

with I, Q, U, and V real.

Stokes' parameter I measures the total intensity of the radiation field, Q and U the linearly polarized intensity, and V the circularly polarized intensity. I always is non-negative. For a totally unpolarized wave, $Q = U = V = 0$; for a partially polarized wave, the ratio $\sqrt{Q^2 + U^2 + V^2}/I$ measures the total degree of polarization, $\sqrt{Q^2 + U^2}/I$ the degree of linear polarization, and $\frac{1}{2} \arctan \frac{U}{Q}$ the orientation angle of the linearly polarized component. $Q + iU$ is called the complex linear polarization. The IAU and IEEE orientation/sign conventions have the z-axis directed toward the observer, the x-axis directed north, and a $+i$ in the argument of the exponential kernel of the FT. Positive V corresponds to right circular polarization, and conversely. The polarization response of an interferometer can be described by forming the so-called cross-spectral density matrix, which is like the S above but is formed from measurements of the electric field taken at two points in space. For further details, including a description of the polarization response of an interferometer, for various feed configurations, see Carl Bignell's Lecture No. 6 in the *1982 Summer Workshop Proceedings*.

Stokes' visibility functions — Stokes' visibility functions, V_I, V_Q, V_U, and V_V, are the Fourier transforms (FT's) of the radio brightness (spatial) distributions of *Stokes' parameters*, $I(x, y)$, $Q(x, y)$, $U(x, y)$, and $V(x, y)$. (Here, $V_I = \hat{I}$, $V_Q = \hat{Q}$, etc., where $\hat{\ }$ denotes FT.)

For a radio interferometer with ideal circularly polarized feeds, the relations between Stokes' visibility functions and the visibilities, V_{RR}, V_{LL}, V_{RL}, and V_{LR}, obtained by correlating right circular response with right, left with left, etc., are $V_I = \frac{1}{2}(V_{RR} + V_{LL})$, $V_Q = \frac{1}{2}(V_{LR} + V_{RL})$, $V_U = \frac{i}{2}(V_{LR} - V_{RL})$, $V_V = \frac{1}{2}(V_{RR} - V_{LL})$. Note that each of Stokes' visibility functions is *Hermitian*. On the assumption that circular polarization is absent (i.e., that $V(x, y) \equiv 0$), V_{RR} is equal to V_{LL}, and both are Hermitian.

Components of the systematic errors affecting visibility measurements are i.f.-dependent; hence VLA u-v data files usually do not contain Stokes' visibilities, but rather V_{RR}, V_{LL}, V_{RL}, and V_{LR}—as these are what is required for calibration purposes. Stokes' visibility functions generally are constructed only within the mapping programs. (But the AIPS visibility data format is designed to accommodate either type of visibility function, and the map making tasks are able to recognize the form of their input data and deal with them appropriately.)

sub-image — in AIPS parlance, any linear, rectangular, or hyper-rectangular section of an *image*.

sub-task — a task, or computer program, whose execution is initiated by the action of another program. The act of initiating the execution of the sub-task is called *task shedding* or *task spawning*. See *task*.

super-resolution — The problem of image reconstruction in radio interferometry is one of finding an approximation to an unknown function f (generally assumed to be of *compact support*) from *partial knowledge* of its Fourier transform $\hat{f}$ — i.e., from a finite number of measurements of the visibility. Any of the techniques which are applied to the problem—the *Högbom Clean algorithm*, the *regularization method*, etc.—may be thought of as methods of smoothing, interpolating, and extrapolating the noisy measurements. *Super-resolution* is a term which refers to the extrapolation aspect: Cautious extrapolation yields an image whose *spatial resolution* is $\approx \lambda/D$, where D is the diameter of the largest centered region in the u-v plane which has been reasonably well sampled. Less cautious extrapolation yields super-resolution; spurious detail appears as caution is abandoned.

Super-resolution in a *Clean map* is effected by choosing an artificially narrow *Clean beam*. With regularization methods (in image reconstruction, and more generally), super-resolution comes about by choosing a small value of the *regularization parameter*. The spatial resolution achieved by a regularization method may be signal-to-noise dependent—bright features may be super-resolved, and dim ones not.

support — The closure of that subset of the domain of definition of a function f (or of a generalized function, or distribution) on which the function assumes a nonzero value is called the *support* of the function, and is denoted by $\mathrm{supp}(f)$. I.e., $\mathrm{supp}(f) = \overline{\{\, x \mid f(x) \neq 0 \,\}}$.

For example, the support of the function $f(x) = x$ is the whole real line, even though $f(0) = 0$. And the support of

$$f(x, y) = \begin{cases} 1 , & x^2 + y^2 < 1 , \\ 0 , & \text{otherwise} , \end{cases}$$

is the *closed* unit disk, $\left\{ (x, y) \;\middle|\; x^2 + y^2 \leq 1 \right\}$.

In Euclidean space, a function f whose support is bounded—i.e., such that $f \equiv 0$ "far-out"—is said to be of *compact support*. The Fourier transform of a nontrivial function of compact support (such as a u-v *measurement distribution* or a *gridding convolution function*) cannot itself be of compact support; i.e., it has "sidelobes" extending to infinity.

support width — of a function whose *support* is a rectangle or a hyper-rectangle (e.g., the Fourier transform of a band-limited function), the linear measure of one of the edges of its *support*.

SVPSF — See *point spread function*.

Synthesis Imaging in Radio Astronomy — A collection of lectures from the 1988 (Third) NRAO Synthesis Imaging Summer School edited by R. A. Perley, F. R. Schwab and A. H. Bridle. (Astronomical Society of the Pacific Conference Series, Volume 6 (1989)). A very useful reference book for the reduction of radio interferometric data. This volume supersedes the proceedings from the earlier workshops.

synthesized beam — in radio interferometry, the *beam*—but always ignoring instrumental effects. Hence, the synthesized beam is fully determined by the *u-v sampling distribution*, the *u-v*

weight function, the *u-v taper function*, and the *gridding convolution function*. See *beam*.

tape blocking efficiency — Data are stored on magnetic tape in units of *blocks*. An *inter-record gap*—essentially wasted space—separates one block from the next. The *tape blocking efficiency*, or the fraction of unwasted space, is the ratio

$$\frac{\frac{\text{block length}}{\text{recording density}}}{\frac{\text{block length}}{\text{recording density}} + \text{length of an inter-record gap}}.$$

The length of an inter-record gap is about $\frac{3}{4}$, $\frac{3}{5}$, and $\frac{3}{10}$ inch at recording densities of 800, 1600, and 6250 bpi, respectively.

taper — See *u-v taper function*.

task — used in two senses: 1) the execution of a computer program and 2) the program itself. Thus, if two computer users are (independently) running the same program at the same time, it may be said either that two tasks are running, or that two incarnations of the same task are in existence. A *sub-task* (*q.v.*) is a task whose execution is initiated by the action of another program. Many of the more complicated and the more specialized functions of AIPS are accomplished by the action of sub-tasks shed by the AIPS program. (Simpler functions are invoked by the issuance of *verb* commands—see *POPS symbols*.)

t-b order — See *time-baseline order*.

TEK screen — a cathode ray tube (CRT) terminal and display device appropriate for pictorial display of data, in the form of contour plots, graphs, etc., as well as for display of textual data. The Tektronix company's Model 4012 terminal (with a green P4 phosphor, hence the synonymous term *green screen*) is the canonical device of this type. The "make copy" button on this device can be used to produce a copy, on paper, of the image shown on the CRT screen. Each of the NRAO's AIPS data reduction computers is outfitted with a *TEK screen*.

TEK4012 — same as *TEK screen*.

Telex 6250 tape drive — a model of tape drive used on the VLA Vaxes, capable of operation at 1600 and 6250 bpi.

terminal page — Many modern computer terminals contain a semiconductor memory with a capacity of several CRT screen loads ($\approx$ 24 lines) of character data. A *terminal page* is a unit of one screen load of such data. Certain terminal keys allow one to cause data which previously appeared on the CRT screen to reappear—this feature is called *terminal scroll* (*q.v.*). A typical terminal at the NRAO has three terminal pages of memory.

terminal scroll — that feature present on certain models of computer terminals which allows data which previously appeared on the CRT screen to be made to reappear. Often, depressing one key on the terminal will cause earlier information to reappear line-by-line (this is termed *line scroll*), while the action of another key will cause a whole earlier screen load to reappear (this is termed *page scroll*).

text editor — a computer program designed for the creation, manipulation, and modification of computer files containing textual data such as reports, documentation, alphanumeric command lines, and program source code. Generally, one or more text editors are supplied by the computer manufacturer. Three text editors are in widespread use on the Vax—*SOS*, *EMACS* and *EDT*. *vi*, *edt* and *emacs* are used on NRAO's Convex computers. See *line editor* and *screen editor*.

text file — a computer data file containing only textual data, as might be written by a *text editor* (*q.v.*). Programs such as the AIPS tasks sometimes write messages, especially progress report messages, into a text file—see *message file.*

Third NRAO Synthesis Imaging Summer School — The 1988 Summer School on Synthesis Imaging which was held in Socorro, New Mexico in June 1988. The lectures were formally published in *Synthesis Imaging in Radio Astronomy*.

thrashing — See *memory thrashing*.

time-baseline order — An ordered set of visibility measurements $\{ V_{ij}(t_k) \mid \ 1 \le i < j \le n,\ k = 1, \ldots, l \}$ recorded with an n element interferometer at times $t_1 < t_2 < \ldots < t_l$ is said to be in *time-baseline order* if the ordering is such that all of the data obtained at time t_1, sorted into the canonical ordering by baseline, occur first, followed by the data obtained at time t_2, again ordered canonically, etc., etc. (The canonical ordering by baseline is the order $V_{12}, V_{13}, \ldots, V_{1n}, V_{23}, \ldots, \ldots, V_{n-1,n}$.) Compare *baseline-time order*.

Time-baseline ordering of a *u-v data file* is convenient for calibration purposes. The AIPS task for self-calibration requires that its input *u-v* data file be time-baseline ordered.

time smearing — in a radio interferometer map, the space-variant broadening of the *point spread function* (or *beam*) which is due to time averaging of the data. When, for example, the visibility data along a *u-v* track are averaged, with equal weight, over time intervals of width Δt sec., the visibility amplitude of a point source is reduced by a factor $\approx \frac{\sin\gamma}{\gamma}$ —where $\gamma \equiv \pi(u'x + v'y + w'z)\Delta t$, where the primes denote the time rate of change of the spatial frequency coordinates (u, v, w) along the track (wavelengths/sec.), and where (x, y, z) denotes the direction cosines of the location of the point source with respect to the *phase tracking center*. For further details, see A. R. Thompson's Lecture No. 2 and Alan Bridle and Fred Schwab's Lecture No. 13 in the *Third NRAO Synthesis Imaging Summer School*. Compare *bandwidth smearing*.

trackball — a spherical ball mechanism, about the size (10 cm., or so, in diameter) of a tennis ball, which may be oriented manually by the interactive user of a television display device such as the I^{2}S. The ball can be rotated about any axis, and its orientation, which is sensed by the computer, typically is used to control the enhancement or the coloration of the displayed data (i.e., to control the TV *transfer function(s)*), or to position the *TV cursor*, in order to point out to a program features in the displayed image which are of particular interest.

trackball button — On the unit which houses the trackball for the I^{2}S Model 70 TV display device are the four *trackball buttons*, labeled A, B, C, and D. These are switches that are used, in conjunction with the display routines, to exert additional control over the TV display. Occasionally these buttons are put to other use in AIPS, such as stopping the Clean deconvolution program.

transfer function — a transform which can be used to describe the output of a device (say, an electrical transducer) as a function of the input to the device. See *TV look-up table*.

TRC — *top right corner*, the corner of an image diagonally opposite the BLC. See *m* × *n map*.

true color display — a type of *false color display*, (*q.v.*).

TU77 tape drive — a model of tape drive used on the NRAO's Vaxes, capable of operation at 800 and 1600 bpi.

TU78 tape drive — a model of tape drive used on the VLA Vaxes, capable of operation at 1600 and 6250 bpi.

TV blink — a feature of a computer-controlled TV display device, such as the I^2S, intended to facilitate the comparison of a pair of images stored on two different *image planes*. The TV display is made to alternate between the two images. The AIPS implementation of blinking allows the user, by manipulating the *trackball*, to control the rate of alternation and the fraction of time that each image is displayed.

TV cursor — See *crosshair*.

TV image catalog — See *image catalog*.

TV look-up table — a memory within the control unit of a TV display device which is used for storage of the *transfer functions* controlling the intensity of the display, as a function of pixel value. Within AIPS, the transfer functions may be altered through the use of interactive verbs and manipulation of the *trackball*.

TV roam — a feature of a computer-controlled TV display device such as the I^2S which allows contiguous parts of a single large image, stored on more than one *image plane*, to be displayed as if the image were stored on a single, larger image plane. On the I^2S unit, the portion of the image to be displayed on the TV screen is selected by manipulation of the *trackball*. See *image plane*.

TV scroll — a feature of a computer-controlled TV display device such as the I^2S which allows the display of an image stored on a single *image plane* to be moved about the display screen. This feature, which also is called panning, commonly is used in combination with the *TV zoom* capability. On the I^2S unit, the scroll ordinarily is controlled by manipulation of the *trackball*. Compare *TV roam*.

TV zoom — a magnification feature of a computer-controlled TV display device such as the I^2S. On the I^2S, the three available magnification factors (which multiply the linear dimensions of the original display of the image by a factor of 2, 4, or 8) generally are selected by depressing one of the *trackball buttons*. Since the magnification is achieved by pixel replication (i.e., by piece-wise linear interpolation)—rather than by a smooth interpolation—the visual impression may be somewhat displeasing. The entire magnified image may not fit on the TV screen, so zoom usually is used in combination with the *TV scroll* feature.

uniform weighting — A *dirty map* obtained by computing the inverse Fourier transform (FT) of a weighted *u-v measurement distribution* in which each visibility sample has been weighted in inverse proportion to the local density of the *u-v coverage* is said to have been computed using *uniform weighting*. When a radio map is computed via the fast Fourier transform algorithm, uniform weighting may be achieved by computing normalized discrete convolution summations $\sum_{i=1}^{N} C(u-u_i, v-v_i)\tilde{V}_i/N$, where (u,v) denotes the spatial frequency coordinates of a given *u-v* grid cell, where C is an appropriately chosen *gridding convolution function*, and where the $\tilde{V}_i$ are the N visibility measurements obtained at positions (u_i, v_i) in some neighborhood of (u,v), the size of which is determined by the *support* of C. The uniform weighted map is given by the inverse discrete FT of data interpolated and smoothed in this manner, onto the lattice points of a rectangular grid. So-called *natural weighting* is achieved by using unnormalized convolution sums, rather than by dividing by N. The AIPS map making tasks use a weighting scheme which is slightly more complicated than that described here.

Since the density of *u-v* coverage typically is greater in the inner regions of the *u-v* plane, a map computed using uniform weighting has finer *spatial resolution* than one computed with natural weighting. With natural weighting, low surface-brightness extended features may be more easily discernible than with uniform weighting. Essentially the same effect can be achieved with uniform weighting, when accompanied by use of a *u-v taper function*.

UNIX — a "universal" computer operating system developed at the Bell Telephone Laboratories. Its virtue is that program packages such as AIPS—once having been made to run under one UNIX-based operating system—ought to run on any other such system, even on a computer of different manufacture, with no alterations. Many Vaxes operate under UNIX, though not the NRAO's. The Convexes C-1 in Charlottesville and at the AOC operates under UNIX. See *operating system*.

user-coded task — an AIPS *task* written by a user, rather than by a professional programmer or a member of the AIPS programming group. One of the design goals for AIPS, not yet fully realized, is that it should be relatively easy for a user who is not an experienced programmer to write an AIPS task suited to his own needs—i.e., that it should be fairly simple for him to make some sense of the AIPS database, and to get at his data and manipulate it as he sees fit. The AIPS task named FUDGE is intended to serve as a paradigm for user-coded tasks for manipulation of *u-v data files*; two other tasks, TAFFY and CANDY, are paradigms for *image file* manipulation. A useful reference is the manual by W. D. Cotton and a 'cast of AIPS' [*Going AIPS! A Programmers Guide to the NRAO Astronomical Image Processing System*, NRAO, Charlottesville, VA, 1990].

The addition to AIPS of new *verbs*, and modification of the functioning of existing verbs, requires modifying the AIPS program itself; this is best left to the AIPS programming group.

u-v coverage — the *support* of the *u-v sampling distribution* (*q.v.*). Also see *conjugate symmetry*.

u-v data file — in AIPS, a *primary data file* designed to accommodate the measurements of the visibility function of a radio source.

u-v data flag — In an AIPS *u-v data file*, each visibility measurement is accompanied by a real-valued weight, which ordinarily is (positive and) proportional to the length of the integration period over which the measurement was obtained. A non-positive weight represents a *u-v* data flag, which signifies that the visibility measurement ought to be ignored. See *flagging* and *clipping*.

u-v FITS format — an extension of the *FITS format* (originally designed for the interchange of image data) to accommodate radio interferometer visibility data [E. W. Greisen and R. H. Harten, An extension of FITS for groups of small arrays of data, *Astron. Astrophys. Suppl. Ser.*, **44** (1981) 371–374]. See *FITS format*.

u-v measurement distribution — in radio interferometry, a linear combination of shifted Dirac δ-functions, one located at the position in the *u-v* plane of each visibility measurement, and each weighted by the visibility measurement obtained at that location. Denoting the *u-v* coverage by $\{(u_i, v_i)\}_{i=1}^{n}$, the visibility function by V, and the measured visibility by $\tilde{V}$, the (two-dimensional) *u-v* measurement distribution S is given by

$S(u,v) = \sum_{i=1}^{n} \tilde{V}(u_i, v_i)\delta(u-u_i, v-v_i)$. Compare *u-v sampling distribution*.

This definition may be modified to incorporate two types of weight function, yielding a *weighted* and/or *tapered* measurement distribution—see *u-v taper function* and *u-v weight function*.

The visibility measurements $\{\tilde{V}(u_i, v_i)\}$ are not actual samples of V, but rather are error-corrupted samples of a function which represents some sort of *local average* of the visibility—this is a distinction which it is worthwhile to note, and then to ignore. Various systematic errors affecting the measurements may be corrected by proper calibration—see *antenna/i.f. gain* and *instrumental polarization*.

u-v sampling distribution — in radio interferometry, a linear combination of shifted Dirac δ-functions, one located at the position in the u-v plane of each visibility measurement. Sometimes termed *u-v transfer function*. See *beam*.

If $\{(u_i, v_i)\}_{i=1}^{n}$ (the *u-v coverage*) is the set of spatial frequency coordinates at which the source visibility has been sampled, then the (two-dimensional) u-v sampling distribution S is given by $S(u,v) = \sum_{i=1}^{n} \delta(u-u_i, v-v_i)$.

Occasionally the term *u-v sampling distribution* is used in the same sense as the term *u-v measurement distribution* (*q.v.*).

u-v taper function — an even, real-valued weight function (typically, an elliptical Gaussian), smooth and peaked at the origin, which may be incorporated into the definition of *u-v measurement distribution* or *u-v sampling distribution*, above, serving to control the spatial resolution of the radio map or the beam; i.e., to enhance the response to extended features in the radio source brightness distribution by giving relatively higher weight to the measurements at short u-v spacings. Compare *u-v weight function.*

u-v transfer function — same as *u-v sampling distribution*, but always explicitly incorporating any u-v weight function or u-v taper function.

u-v weight function — a real-valued function which may be incorporated in the definition, above, of *u-v measurement distribution* or *u-v sampling distribution*, serving to weight each measurement either according to an estimate of the statistical measurement error, or according to the local density of sampling, or both. Compare *u-v taper function* and see *uniform weighting.*

Varian printer — an electrostatic printer/plotter manufactured by the Varian Corp.

Variational Method — the name which applies to Tim Cornwell's AIPS implementation (in the program VM) of the *maximum entropy method*, to solve the image deconvolution problem $g = b * f$, where g and b are given, and f is unknown. The regularizing term $S(\tilde{f})$ (see *regularization method*), a function of the computed approximate solution $\tilde{f}$, is given by the negative of an entropy expression, of the form

$$H(\tilde{f}) = -\int_A \tilde{f}(x) \log \frac{\tilde{f}(x)}{h(x)}\, dx\,.$$

Here A denotes the (assumed known) support of f, and h is a prior estimate of f; when $h \equiv$ constant, this agrees with the standard formulation of the maximum entropy method. A weighted sum $\chi^2(\tilde{f}) + \lambda S(\tilde{f})$ of a χ^2 error term and S is minimized, and the regularization parameter λ is chosen so that the r.m.s. residual corresponding to the final iterate is approximately equal to an input value. For optical data the χ^2 term is taken as $\|g - b * \tilde{f}\|^2$, whereas for radio data the χ^2 term is evaluated in the visibility domain, where the measurement errors may more properly be assumed to be statistically independent. Also, $\int_A \tilde{f}$ is constrained to be near an estimate of the *zero-spacing flux* which is supplied by the user. The minimization is done using a Newton-type method, with a diagonal approximation to the Hessian of the objective function and intricate control of the step length. In terms of execution speed, this method is competitive with the *Clark Clean algorithm*—at least in the case of large objects of complex structure observed with the VLA—and superior results usually are obtained for this class of objects. See [T. J. Cornwell, Deconvolution with a maximum entropy type algorithm, VLA Scientific Memo. No. 149].

verb — See *POPS symbols.*

Versatec printer — an electrostatic printer/plotter manufactured by the Versatec Corp., and used on the NRAO's AIPS computer systems.

Very Long Baseline Interferometry — Techniques and Applications — Proceedings of the NATO Advanced Study Institute held at Castel S. Pietro Terme, Bologna, Italy in 1988. Edited by M. Felli and R. E. Spencer. Kluwer Academic Publishers, Dordrecht (1989). This volume contains much useful information on the planning and execution of VLBI observations as well as on the reduction of VLBI data.

vi — a moderately sophisticated text editor (a *screen editor*) used on computers which run the UNIX operating system. See *text editor*.

virtual memory page — on a computer running under a virtual memory operating system, one unit of *virtual memory storage*. At a typical Vax installation, the size of a virtual memory page is 512 bytes.

virtual memory page swapping — on a computer running under a virtual memory operating system, the action (initiated automatically by the operating system) of reading new virtual memory pages into the physical memory, and storing on disk (i.e., in the *virtual memory*) the data which thus have been displaced. Each occurrence of the displacement of a memory page is referred to as a *page fault*. See *memory thrashing*.

virtual memory storage — computer storage—typically disk storage—in an area apart from the *physical memory* of a computer. Access to virtual memory storage is controlled by the operating system, in a way intended to give the programmer the illusion that a large amount of physical memory is present. Access to virtual memory may be much slower than access to physical memory, and the operating system may incur a significant amount of overhead in managing the virtual memory. See *memory thrashing*.

visibility phase tracking center — In a correlating-type radio interferometer usually the *fringe stopping center* and the *delay tracking center* coincide. When this is the case, both are referred to as the visibility phase tracking center.

VM — See *Variational Method*.

VMS — (Virtual Memory System) the operating system used on the NRAO's Vax computers. See *virtual memory storage* and *operating system*.

wedge — a legend, or scale—generally in the form of a bar graph with gradations in *intensity* and *chromaticity*—which

may be displayed adjacent to a photographic or video display of a digitized *image*. The wedge is a visual representation of the *transfer function* that was used in generating the display. The wedge is either colored or gray, depending on whether the display is a *pseudo-color display* or a *gray-scale display*.

Note that a *false color display* would require more than one wedge (or a multi-tiered wedge) to display the several transfer functions, as well as an additional wedge to display the possible color mixtures.

window Clean — an application of the *Högbom Clean algorithm*, with an explicit specification, by the user, of the Clean window. Generally the user should specify a Clean window whenever it is possible to make a reasonably valid and restrictive estimate of the *support* of the true radio source brightness distribution. At the termination of the algorithm, it is prudent to examine a display of the residual map for the presence of large residuals outside of the Clean window; their presence could suggest that an inappropriate window was selected. See *Clean window.*

working set size — on a computer running under a virtual memory operating system, the amount of *physical memory* allocated to a task. Any program memory requirement in excess of the working set size is relegated to *virtual memory storage*. At a typical Vax installation, the working set size is set at $\frac{1}{4}$ or $\frac{1}{2}$ megabyte.

x-y order — An ordered set of visibility samples $\{V(u_i, v_i, w_i)\}_{i=1}^{n}$ arranged according to descending absolute value of the spatial frequency coordinate u — i.e., with $|u_1| \geq |u_2| \geq \ldots \geq |u_n|$ — is said to be in *x-y order*.

x-y order is a convenient ordering for the operation of gridding convolution; hence the AIPS mapping tasks require that their input *u-v data files* be sorted accordingly. See *sort order*.

Y-routine — in AIPS, a subroutine designed to aid in the use of a specific model of TV display device, such as the I^2S Model 70. AIPS requires a relatively small core of Y-routines implementing basic TV display functions; complicated display functions then are accomplished by combining these basic functions that are supposed to be common to many models of TV display device. At present there are approximately 25 Y-routines for use at those AIPS installations equipped with an I^2S. Compare *Z-routine.*

zero-spacing flux — The visibility $V(u,v) \equiv \hat{f}(u,v)$ ($\hat{}$ denotes Fourier transform) of a source brightness distribution f in a neighborhood of $u = v = 0$ is inaccessible to an interferometer composed of elements of finite collecting area. The *zero-spacing flux* is equal to the total, or integrated flux density of the source—i.e., it is given by $V(0,0) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y)\, dx\, dy$. Because the hole in the *u-v coverage* in the neighborhood of the origin may be fairly large, image reconstruction methods, such as the *Högbom Clean algorithm*, may do a poor job, within this central region, of interpolating the measured data. This frequently is manifested by the appearance of a *negative bowl artifact*—a negative 'baseline' beneath the reconstruction of f —owing to the reconstruction method having underestimated the zero-spacing flux. The *Variational Method* for maximum entropy reconstruction requires that the user supply an estimate of $V(0,0)$. The Clean algorithm, too, may benefit if a datum at $u = v = 0$ is included when the *dirty map* is constructed.

A zero-spacing estimate can be derived from single-dish measurements. Providing a proper estimate is difficult, because of contamination of single-dish measurements by 'confusing sources.' The estimate ought to correspond to a telescope with the same primary beam response as the array elements; and it is not just a single datum $V(0,0)$ which is missing, but rather a region—so proper weighting of the zero-spacing information is tricky. See Tim Cornwell and Robert Braun's Lecture No. 8 in the *Third NRAO Synthesis Imaging Summer School*.

zoom — See *TV zoom.*

Z-routine — in AIPS, a subroutine—generally designed to perform some routine, often needed function—written for a specific model of *host computer* or for a specific host computer operating system. The implementation of certain basic functions, especially those for file access and file management, generally is machine dependent and operating system dependent. The typical AIPS installation requires 50–100 Z-routines. Compare *Y-routine.*

1978 Groningen Conference Proceedings — *Image Formation from Coherence Functions in Astronomy. Proceedings of IAU Colloquium No. 49 held at Groningen, the Netherlands, August 10–12, 1978*, edited by C. van Schooneveld, D. Reidel, Dordrecht, Holland, 1979—contains many papers on aperture synthesis techniques, including some of the early papers on *hybrid mapping*.

1982 Summer Workshop Proceedings — *Synthesis Mapping. Proceedings of the NRAO–VLA Workshop held at Socorro, New Mexico, June 21–25, 1982*, edited by A. R. Thompson and L. R. D'Addario, NRAO, Green Bank, WV, 1982—a collection of the fifteen lectures which comprised this short course on aperture synthesis techniques—a useful introduction to VLA data reduction methods.

1983 Sydney Conference Proceedings — *Indirect Imaging: Measurement and Processing for Indirect Imaging. Proceedings of an International Symposium held in Sydney, Australia, August 30–September 2, 1983*, edited by J. A. Roberts, Cambridge Univ. Press, Cambridge, 1984—contains a number of interesting papers on aperture synthesis techniques.

1985 Summer School Proceedings — lecture notes from the second NRAO summer short course on radio-interferometric imaging (in preparation). This volume supersedes the *1982 Summer Workshop Proceedings*.

1988 Summer School Proceedings — lecture notes from the third NRAO Summer School on radio interferometric imaging. The lectures have been published as *Synthesis Imaging in Radio Astronomy* (*q.v.*).

4012 — See *TEK screen.*

I INDEX

— 1 —

1978 Groningen Conference Proceedings . . . G – 21
1982 Summer Workshop Proceedings . . . . . G – 21
1983 Sydney Conference Proceedings . . . . . G – 21
1985 Summer School Proceedings . . . . . G – 21
1988 Summer School Proceedings . . . . . G – 21
1990 *AIPS* Site Survey . . . . . 1 – 1

— 2 —

2-Gigabyte limit . . . . . 9 – 1

— 4 —

4012 . . . . . G – 21

— A —

ABORT . . . . . 3 – 1
ABOUT . . . . . 3 – 12, 12
. . . . . 6 – 5
ABOUT ADVERB . . . . . 13 – 1
ABOUT ANALYSIS . . . . . 13 – 10
ABOUT AP . . . . . 13 – 13
ABOUT ASTROMETRY . . . . . 13 – 13
ABOUT BATCH . . . . . 13 – 14
ABOUT CALIBRATION . . . . . 13 – 15
ABOUT CATALOG . . . . . 13 – 21
ABOUT COORDINATES . . . . . 13 – 23
ABOUT EDITING . . . . . 13 – 24
ABOUT EXTENSION APPLICATIONS . . . . . 13 – 25
ABOUT FITS . . . . . 13 – 26
ABOUT GENERAL . . . . . 13 – 26
ABOUT HARDCOPY . . . . . 13 – 29
ABOUT IMAGE . . . . . 13 – 30
ABOUT IMAGE-UTILITIES . . . . . 13 – 29
ABOUT IMAGING . . . . . 13 – 30
ABOUT INFORMATION . . . . . 13 – 34
ABOUT INTERACTIVE . . . . . 13 – 34
ABOUT MODELING . . . . . 13 – 36
ABOUT OBSOLETE . . . . . 13 – 37
ABOUT ONED . . . . . 13 – 38
ABOUT OOP . . . . . 13 – 39
ABOUT OPTICAL . . . . . 13 – 40
ABOUT PARAFORM . . . . . 13 – 40
ABOUT PLOT . . . . . 13 – 40
ABOUT POLARIZATION . . . . . 13 – 44
ABOUT POPS . . . . . 13 – 45
ABOUT PROCEDURE . . . . . 13 – 48
ABOUT PSEUDOVERB . . . . . 13 – 49
ABOUT RUN . . . . . 13 – 50
ABOUT SINGLEDISH . . . . . 13 – 51
ABOUT SPECTRAL . . . . . 13 – 51
ABOUT TABLE . . . . . 13 – 53
ABOUT TAPE . . . . . 13 – 54
ABOUT TASK . . . . . 13 – 55
ABOUT TV . . . . . 13 – 66
ABOUT TV APPLICATIONS . . . . . 13 – 69
ABOUT UTILITY . . . . . 13 – 70
ABOUT UV . . . . . 13 – 71
ABOUT VERB . . . . . 13 – 78
ABOUT VLA . . . . . 13 – 84
ABOUT VLBI . . . . . 13 – 85
ACCOR . . . . . 9 – 34
ACFIT . . . . . 9 – 42, 42
ACIMG . . . . . 6 – 8
. . . . . 7 – **17**
. . . . . 9 – **34**
ACLIP . . . . . 4 – 11
ACSCL . . . . . 9 – **38**, 40
ADDBEAM . . . . . 10 – 1
ADDDISK . . . . . 12 – 5, 14
adverb . . . . . 3 – **3**
. . . . . 12 – 1, 8
. . . . . 13 – 1
. . . . . G – 1
AFARS . . . . . 7 – **10**
AGAUS . . . . . 6 – 27
. . . . . 7 – 10
. . . . . 8 – 14
AHIST . . . . . 7 – 16
AIPS . . . . . 1 – **1**, **1**, **1**
AIPS* . . . . . 1 – 1
AIPS . . . . . 3 – 1
. . . . . 12 – 1
AIPS monitor . . . . . G – 1
AIPSLetter . . . . . 3 – 12
ALBUS . . . . . 4 – **32**
. . . . . 9 – **16**
aliased response . . . . . G – 1
aliasing . . . . . G – 1
ALLDEST . . . . . 11 – 2
ALTDEF . . . . . 8 – 9
. . . . . 10 – 1
ALTSW . . . . . 8 – 9
ALU . . . . . G – 1
analysis . . . . . 7 – **12**, 16
. . . . . 8 – 14
. . . . . 10 – 20, 20, 22, 22
. . . . . 13 – 10
ANBPL . . . . . 4 – **24**, 24
. . . . . 6 – 3, **8**
ANTAB . . . . . 9 – 2, 30, **30**, 65
. . . . . C – 7, 12, 13

antenna coordinates . . . 4 – 8
antenna file . . . 4 – 1
. . . G – 1
antenna residual delay . . . G – 1
antenna residual fringe rate . . . G – 1
antenna/i.f. gain . . . G – 1
antenna/i.f. phase . . . G – 1
ANTNUM . . . 9 – 19
. . . O – 24
AP . . . 13 – 13
. . . G – 1
AP–120B array processor . . . G – 1
APCAL . . . 9 – **33**, **40**, 40, **40**
APCLN . . . 5 – 24
. . . 6 – 21, 27
. . . 8 – 6
APROPOS . . . 3 – 12
array processor . . . 13 – 13
. . . G – 1
array processor microcode . . . G – 1
associated file . . . G – 2
astrometry . . . 13 – 13
ATLOD . . . 4 – 2
auto re-boot . . . G – 2
AVEOT . . . 3 – 16
AVFILE . . . 3 – 16
. . . 8 – 1
AVGWT . . . 5 – 3
AVSPC . . . 8 – 1, 1, 1
. . . 9 – 63
. . . C – 6
. . . O – 6, 16

— B —

backup . . . F – 1, 4, 4, 6
back-up . . . G – 2
Backups . . . 11 – **2**
BADDISK . . . 3 – 11
BAMODIFY . . . 12 – 5
bananas . . . 2 – 12
. . . 3 – 21, 22, 22, 22
. . . 4 – 40, 40, 41, 41, 41
. . . 5 – 40
. . . 7 – 18, 18
. . . 8 – 17, 18, 18, 18
. . . 9 – 72, 72
. . . 10 – 25, 25, 26, 26
. . . 13 – 87, 87, 88, 88, 88, 88
. . . 13 – 88
. . . A – 6, 6, 6
. . . B – 8, 8
. . . C – 15, 16, 16, 16, 16
. . . D – 7, 8, 8, 8, 8
. . . L – 14
. . . F – 6, 6
. . . Z – 7, 8, 8, 8
bandpass . . . 4 – **16**, **16**
. . . 9 – 2, 38, 38, 41, 62, 62
bandwidth smearing . . . G – 2
bandwidth synthesis . . . 8 – 1
. . . G – 2
baseline error . . . 4 – 8, 8
baseline-time order . . . G – 2
batch . . . 12 – **5**, **5**, 5
. . . 13 – 14
batch editor . . . G – 2
batch job . . . G – 2
batch processor . . . G – 2
batch queue . . . G – 2
BATCLEAR . . . 12 – 5
BATEDIT . . . 12 – 5
BATLIST . . . 12 – 5
BATQUE . . . 12 – 5
"battery-powered" Clean algorithm . . . G – 2
BCOMP . . . 5 – 12
BDAPL . . . 9 – 63
BDF2AIPS . . . 4 – **3**, 3, 3, **3**, 30
. . . L – 1
. . . O – 18
BDFLIST . . . 4 – **3**, 3
. . . L – 1
beam . . . G – 3
beam patch . . . G – 3
beam squint . . . G – 3
beam-switched continuum . . . 10 – 1, **4**, 5, **16**, **16**, 16
. . . 10 – 16, 16, 21
BLANK . . . 6 – 27
. . . 7 – 5, 5
. . . 8 – 14
blanked pixel . . . 7 – 3, 5, 5
. . . 8 – 14
. . . G – 3
BLAPP . . . 9 – 55, **55**
BLC . . . G – 3
BLCAL . . . 5 – 26
. . . 9 – **63**
BLCHN . . . 5 – 26
. . . 9 – **63**
BLEW CORE! . . . 12 – 9
BLING . . . 9 – 2, **55**
blink . . . G – 3
BLOCKING . . . F – 4, 5
blotch . . . 7 – 4, 5
BLPCL . . . 9 – 63
BLPLT . . . 6 – **16**
. . . 9 – **63**
BLPOLCAL . . . 5 – 26
. . . 9 – 63
BLSUM . . . 6 – 4, 13, 27
. . . 7 – 4, 9
. . . 8 – 14, 14
BMAJ . . . 5 – 12
BMIN . . . 5 – 12

boot . . . G – 3
BOT marker . . . G – 3
BOXES . . . 5 – 14, 16
7 – 8
BOXFILE . . . 5 – 6, 14, 14, 16, 16
BPA . . . 5 – 12
BPASS . . . 4 – 16, 16, 16
9 – 38, 38, 62
L – 1
BPEDT . . . 4 – **16**
6 – 27
9 – **38**, 62
L – 1
BPEPL . . . 4 – **16**
6 – 16
BPERR . . . 4 – **16**
BPFLG . . . 4 – **16**
9 – **38**
bpi . . . G – 3
BPLOT . . . 4 – **16**, 16, 32
6 – 16
BPPLT . . . 4 – **16**
6 – 16
BPSMO . . . 4 – **16**
BPWAY . . . 5 – 8
broadband mapping technique . . . G – 3
BSAVG . . . 10 – **21**
BSCAN . . . 9 – 36
BSCLN . . . 10 – 16
BSCOR . . . 10 – **16**, **16**
BSGEO . . . 10 – **16**, **16**
BSGRD . . . 10 – **16**, **16**
BSMOD . . . 10 – 13
BSPRT . . . 9 – 55
BSTST . . . 10 – 16
b-t order . . . G – 3
bug . . . G – 3
button A . . . 2 – 12
button B . . . 2 – 12
button C . . . 2 – 12
button D . . . 2 – 12
byte . . . G – 3

— C —

CALDIR . . . 4 – 15, 19
CALIB . . . 4 – 20, 21
5 – 26, 26, 28, 28, 30, 30
9 – 64, 67, 67, 67
A – 2, 2
D – 1, 1, 1
L – 1, 1, 12
calibration . . . 4 – **1**, **1**, **6**, 7, 7, **8**
4 – **8**, **11**, **14**, **15**, **15**, **16**
4 – **16**, **19**, **20**, **21**, **22**, **23**
4 – **23**, **24**, **24**, **24**, **28**, **28**
4 – **30**, **32**, **32**, **32**, **32**, **32**
4 – **39**, **39**
9 – 2, 2, 2, 25, 26, 26
9 – 26, 26, 28, 29, 30, 30
9 – 30, 34, 35, 35, 36, 36
9 – 38, 40, 41, 42, 42, 44
9 – 45, 46, 46, 47, 48, 48
9 – 48, 48, 48, 55, 55, 57
9 – 55, 57, 57, 59, 59, 60
9 – 61, 63, 63, 64, 65, 65
9 – 67, 67, 69, 71, 71
10 – 11
13 – 15
A – 2, 2
C – 7, 10, 11, 13, 13
D – 1, 1, 1, 1, 1
O – **6**, 7, **15**, **17**, **18**, **19**
O – **20**
calibration file . . . F – 3
CALRD . . . 4 – **19**
CANDY . . . 12 – 18, 21
CAPLT . . . 6 – 8
9 – 21
carriage-return key . . . G – 3
casa . . . 1 – 1
CAT2LOG . . . 3 – 5
CAT3LOG . . . 3 – 5
CAT4LOG . . . 3 – 5
CAT5LOG . . . 3 – 5
CATALOG . . . 3 – 5
5 – 2
catalog entry . . . G – 3
catalog file . . . 3 – 5
11 – 4
13 – 21
G – 3
catalog slot . . . G – 3
CATOLOG . . . 3 – 5
CCEDT . . . 5 – 23
9 – 67
CCFND . . . 5 – 23
CCMOD . . . 7 – 17
CCMRG . . . 5 – 23, 26
CCNTR . . . 6 – 9
CCRES . . . 5 – **12**, 24
7 – **15**
CCSEL . . . 5 – 23, 30
9 – 67

CELGAL . . . . . 8 – 9
cell-averaging . . . . . G – 3
cellsize . . . . . G – 3
CENTR . . . . . 9 – 48
cereal bowl map defect . . . . . G – 4
character strings . . . . . 12 – 8
characteristic function . . . . . G – 4
CHARMULT . . . . . 2 – 3
. . . . . 6 – 23
CHKFC . . . . . 5 – 6, 14
CHKNAME . . . . . 12 – 9
chromaticity . . . . . G – 4
C.I.E. chromaticity diagram . . . . . G – 4
CIRCLEVS . . . . . 6 – 5
CL . . . . . F – 3
Clark Clean algorithm . . . . . 5 – 11, 12
. . . . . G – 4
CLBOX . . . . . 5 – 16
CLCAL . . . . . 4 – 14, 24, 24
. . . . . 9 – 26, 26
. . . . . A – 2
. . . . . D – 1, 1
. . . . . L – 1, 1
. . . . . O – 18
CLCOR . . . . . 4 – 8
. . . . . 9 – 2, 16, 36, 65
. . . . . A – 5
Clean . . . . . 5 – 11
. . . . . 8 – 6
. . . . . G – 4
Clean beam . . . . . G – 4
Clean box . . . . . G – 4
Clean component . . . . . G – 4
Clean map . . . . . G – 4
Clean speed-up factor . . . . . G – 4
Clean window . . . . . G – 4
CLIP . . . . . 4 – 11
. . . . . 5 – 33
. . . . . 8 – 1
. . . . . 10 – 7
clipping . . . . . G – 4
CLNUP . . . . . 11 – **1**
CLOSE . . . . . 5 – 32
. . . . . 6 – **8**
. . . . . 8 – **4**
closure amplitude . . . . . G – 4
closure phase . . . . . G – 4
CLPLT . . . . . 6 – 8
. . . . . 9 – 21
CLRMSG . . . . . 3 – 4
. . . . . 11 – 2
CLRNAME . . . . . 3 – 5
CLRSTAT . . . . . 3 – 6
CLVLB . . . . . 9 – 40
CMETHOD . . . . . 5 – 12
CMPLX . . . . . 7 – **3**
CNTR . . . . . 6 – 9, 9, 9
COHER . . . . . 9 – **23**, 57
color contour display . . . . . G – 5
color printer . . . . . 6 – 24, 28
. . . . . Z – 1
color triangle . . . . . G – 5
COMB . . . . . 7 – 1, 2, 3
. . . . . 8 – 5, 14
. . . . . 10 – 24
compact support . . . . . G – 5
components file . . . . . G – 5
COMPRESS . . . . . 12 – 9
compressed format . . . . . F – 2, 4, 4
conjugate symmetry . . . . . G – 5
CONPL . . . . . 6 – 17
Conrac monitor . . . . . G – 5
continuum . . . . . 8 – 1, 1, 5, 14
control character . . . . . 2 – 2
CONVL . . . . . 7 – 2, 15
. . . . . 8 – 14
. . . . . 10 – 24
convolution theorem . . . . . G – 5
convolving function . . . . . G – 5
CookBook . . . . . 1 – **3**, **3**, **5**
. . . . . A – 2
coordinate reference pixel . . . . . G – 5
coordinates . . . . . 5 – 6
. . . . . 6 – 2
. . . . . 12 – 2
. . . . . 13 – 23
COPIXEL . . . . . 6 – 22
CORER . . . . . 5 – 33
. . . . . 6 – 3
correction for average spectral index . . . . . 5 – 22
correlator offset . . . . . G – 5
COSTAR . . . . . 6 – 23
Cotton/Schwab Clean algorithm . . . . . 5 – 11
CPASS . . . . . 4 – 16
crash . . . . . G – 5
crosshair . . . . . G – 5
CSCOR . . . . . 10 – **11**
CTRL C . . . . . 2 – 2
. . . . . 11 – 3
CTRL D . . . . . 2 – 2
CTRL Q . . . . . 2 – 2
. . . . . 11 – 3
CTRL S . . . . . 2 – 2
. . . . . 11 – 3
CTRL Z . . . . . 2 – 2
. . . . . 11 – 3
. . . . . Z – 4
cube . . . . . 8 – 5, 7, 7, 9
. . . . . G – 5
CUBIT . . . . . 7 – 17
. . . . . 8 – **14**
cursor . . . . . 2 – 12
. . . . . G – 5
CURVALUE . . . . . 5 – 33
. . . . . 6 – 22

CVEL . . . . . 4 – 36
. . . . . 8 – 1, 1
. . . . . 9 – 41
CXIMAGR . . . . . 5 – **40**

— D —

DAT (4mm) . . . . . F – 5
data cube . . . . . G – 5
data file . . . . . G – 6
data flag . . . . . G – 6
database . . . . . G – 6
DBAPP . . . . . 5 – 3
. . . . . 9 – 14
. . . . . O – 22
DBCON . . . . . 5 – **3**
. . . . . 10 – 3, 24
. . . . . O – 22
DCONV . . . . . 10 – 24
deconvolution . . . . . G – 6
DECSHIFT . . . . . 5 – 6, 6, 14
DEFAULT . . . . . 3 – 9
. . . . . 4 – 6
DEFCOLOR . . . . . 6 – 9
DEFLG . . . . . 4 – 11
. . . . . 5 – 33
. . . . . 9 – 32
DELAY . . . . . 12 – 9
. . . . . G – 6
delay beam . . . . . G – 6
DELBOX . . . . . 5 – 16
. . . . . 6 – 21
delete . . . . . 11 – 2, 2
DELZN . . . . . 9 – **5**, **46**, **46**
. . . . . C – **4**
DFILEBOX . . . . . 5 – 16
. . . . . 6 – 21
DFT . . . . . G – 6
DFTIM . . . . . 6 – 8, 9
. . . . . 7 – 17
DFTPL . . . . . 6 – 8
DFTQU . . . . . 7 – 17
DGAUS . . . . . 5 – **26**
Dicomed Image Recorder (Model D47) . . . . G – 6
DIFUV . . . . . 5 – 26
. . . . . 10 – 7
direct Fourier transform . . . . . G – 6
dirty beam . . . . . G – 6
dirty map . . . . . G – 6
discrete Fourier transform . . . . . G – 6
disk hog . . . . . G – 6
disk space . . . . . 3 – 11
. . . . . 9 – 1
DISKU . . . . . 3 – 11
. . . . . 11 – 4
. . . . . F – 3
DISMOUNT . . . . . 3 – 16
DISPLAY . . . . . 12 – 14
DO3DIMAG . . . . . 5 – 6, 14, 14, 22
DOBAND . . . . . 4 – **16**
DOCRT . . . . . 3 – 4
. . . . . 6 – 3
DOFARS . . . . . 7 – **10**
Doppler . . . . . 10 – 3, 12
DOTV . . . . . 5 – 12, 16
. . . . . 6 – 5
. . . . . 8 – 10
DOWAIT . . . . . 11 – 3
. . . . . 12 – 4
DSKEW . . . . . 6 – 2
DTSIM . . . . . 7 – **17**
DTSUM . . . . . 6 – 3
. . . . . 9 – 19
dynamic range . . . . . G – 6

— E —

EDIT . . . . . 12 – 9
EDITA . . . . . 4 – 11, **23**, **23**, 23, 30
. . . . . 5 – 28, 34
. . . . . 6 – 27
. . . . . 9 – 16, 32
. . . . . L – 1, 1, 1
. . . . . O – 19
editing . . . . . 4 – **11**, **23**, **23**, 28, 28
. . . . . 5 – 33, 34, 34, 34, 34, 34
. . . . . 8 – 1
. . . . . 9 – 2, 22, 32, 32, 69
. . . . . 10 – 7, 7, 7, 7
. . . . . 13 – 24
. . . . . C – 7
. . . . . O – **7**, **9**, **9**, **9**, **9**, **9**
. . . . . O – 16
EDITR . . . . . 4 – 11, 11
. . . . . 5 – 1, **33**, **34**, **34**, **34**, **34**
. . . . . 5 – **34**
. . . . . 6 – 27
. . . . . 9 – 22, 32
. . . . . 10 – 7, 7
. . . . . C – 7
EDT . . . . . G – 7
EHEX . . . . . 3 – 18
. . . . . 12 – 3
ELFIT . . . . . 7 – 17
emacs . . . . . 2 – 10
. . . . . G – 7
ENDBATCH . . . . . 12 – 5
ENDEDIT . . . . . 12 – 9
ERASE . . . . . 12 – 9
EVACL . . . . . 5 – **32**
EVASN . . . . . 4 – 24
EVAUV . . . . . 5 – **32**

EVLA . . . 4 – 1, **2**, **3**, **3**, **3**, **6**
. . . 4 – **7**, **7**, **8**, 10, **11**, **11**
. . . 4 – **14**, **15**, **16**, **16**, **20**, **20**
. . . 4 – **21**, **22**, **23**, **23**, **24**, **24**
. . . 4 – **24**, **28**, **28**, **30**, **30**, **32**
. . . 4 – **32**, **32**, **32**, **32**, **36**, **38**
. . . 4 – **39**, **40**, **41**
. . . L – **1**, **1**, **1**, **1**, **1**, **1**
. . . L – **1**, **1**, **1**, **1**, **12**, **12**
. . . V – 1, 2, 2, 5, 6, 6
. . . V – 6
Exabyte (8mm) . . . F – 5
EXIT . . . 11 – 3
exp × sinc function . . . G – 7
EXPLAIN . . . 3 – **12**
. . . 6 – 5
explain file . . . G – 7
EXPORT format . . . G – 7
EXPORT tape . . . G – 7
expressions . . . 12 – 1, 8
EXTAB . . . 12 – 16
EXTDEST . . . 4 – 6, 30
. . . 6 – 5
. . . 7 – 9
. . . L – 1
. . . O – 7
extended hexadecimal . . . 3 – 18
. . . 12 – 3
extension file . . . G – 7
extension files . . . 3 – 7
. . . 4 – 1
. . . 12 – 2
. . . 13 – 25
. . . A – 2
. . . F – 3
. . . O – 2
EXTLIST . . . 6 – 4, 5
. . . 7 – 9

— F —

F2PS . . . 12 – 16
. . . Z – 4
FACES . . . 5 – 26
FACTOR . . . 5 – 12
false color display . . . G – 7
FARS . . . 7 – **10**
. . . 8 – 14
fast Fourier transform algorithm . . . G – 7
FFT . . . 5 – 33
. . . G – 7
fg . . . Z – 4
FGCNT . . . 4 – 11, **28**
. . . 5 – **33**
FGDIF . . . 4 – 11
FGPLT . . . 4 – 11
. . . 6 – 16
FGPRT . . . 4 – 11, **28**
FGSPW . . . 4 – 11
FGTAB . . . L – 1
FILEBOX . . . 5 – 16
. . . 6 – 21
FILEZAP . . . 3 – 17
. . . 12 – 13
FILIT . . . 5 – **16**
. . . 6 – 21
FILLM . . . 3 – 20
. . . A – 2
. . . D – 1
. . . F – 1, 1
. . . O – **2**, **2**, **2**, 2, 2, **2**
. . . O – 6, 18
filtering . . . 7 – 16
FINDR . . . 4 – 11
. . . 5 – **33**, 33
FINISH . . . 12 – 9
FITAB . . . 3 – 19
. . . 6 – 5
. . . 11 – 2
. . . F – 4, 6
FITDISK . . . 3 – 19
. . . 11 – 2
FITLD . . . 3 – 19
. . . 4 – 2, 5
. . . 5 – 2
. . . 6 – 1
. . . 8 – 1
. . . 9 – 2, **7**, **7**, **7**, **7**
. . . 10 – 4
FITS . . . 1 – 1
. . . 3 – 19
. . . 13 – 26
FITS format . . . G – 7
FITS tape . . . G – 8
FITS-disk . . . 3 – 17, 19
. . . 6 – 1
. . . 12 – 16
fitting . . . 7 – 6, 7, 8, 9
. . . 8 – 14
FITTP . . . 6 – 5
. . . 11 – 2
. . . 12 – 4
. . . A – 5
. . . L – 12
. . . F – 4, 6
FIXBX . . . 5 – **14**, **16**
FIXFR . . . 4 – **3**
FIXWT . . . 5 – 8
flagging . . . 4 – **11**, **23**, **23**, 28, 28
. . . 5 – 33, 33, 34, 34, 34, 34
. . . 8 – 1
. . . 9 – 22, 32, 32, 69
. . . 10 – 7, 7, 7, 7
. . . O – **7**, **9**, **9**, **9**, **9**, **9**
. . . O – 16
. . . G – 8

FLAGR . . . . . 4 – 11
. . . . . 5 – **33**
FLAMLEVS . . . . . 6 – 5
FLATN . . . . . 5 – 14
. . . . . 7 – 3
FLDSIZE . . . . . 5 – 6, 14, 16
FLGIT . . . . . 5 – 33
. . . . . 8 – 1
flux density . . . . . 7 – 4
font . . . . . 1 – 5
FOR . . . . . 12 – 8
FQ number . . . . . 4 – 6
. . . . . O – 2, 2
FQCENTER . . . . . 9 – 48
FQUBE . . . . . 5 – 21
. . . . . 7 – 3
. . . . . 8 – 7
. . . . . 12 – 21
FREESPAC . . . . . 3 – 11
. . . . . 11 – 4
. . . . . Z – 5
FREQID . . . . . 4 – 6
FRING . . . . . 4 – 14
. . . . . 9 – 2, **36**, **36**, 44, 44, **48**
. . . . . 9 – **48**, **48**, **57**, **57**, **57**, 59
. . . . . L – 1
fringe rotator . . . . . G – 8
fringe-fitting . . . . . 9 – 35, 36, 36, 44, 44, 45
. . . . . 9 – 46, 47, 48, 48, 48, 48
. . . . . 9 – 48, 55, 55, 57, 55, 57
. . . . . 9 – 59, 60, 61
. . . . . C – 10
FRMAP . . . . . 6 – 8
FRPLT . . . . . 6 – 8
. . . . . 9 – **24**
FTARS . . . . . 7 – **10**
FTFLG . . . . . 4 – 11
. . . . . 5 – **33**
. . . . . L – 1
ftp . . . . . 12 – 16
FUDGE . . . . . 12 – 18, 21
full-synthesis map . . . . . G – 8
FXPOL . . . . . 9 – **15**
. . . . . O – **23**
FXVLB . . . . . 9 – 7

— G —

GAIN . . . . . 5 – 12
gain file . . . . . G – 8
GAL . . . . . 6 – 17
. . . . . 7 – 17
. . . . . 8 – 14
GAMMASET . . . . . 6 – 28
Gaussian . . . . . 7 – 7, 8, 9, 17
. . . . . 8 – 14
. . . . . 10 – 22, 22
Gaussian-tapered sinc function . . . . . G – 8
general information . . . . . 13 – 26
geometric transformation . . . . . 7 – **12**, **12**
Gerchberg–Saxton algorithm . . . . . G – 8
GET . . . . . 3 – 9
. . . . . 12 – 5, 9
GET2N . . . . . 3 – 6
GET3N . . . . . 3 – 6
GET4N . . . . . 3 – 6
GETHEAD . . . . . 8 – 9
GETITIME . . . . . 9 – **19**, **48**
GETJY . . . . . 4 – 22, 21
. . . . . A – 2
. . . . . D – 1
. . . . . L – 1
GETNAME . . . . . 3 – 6
GETONAME . . . . . 3 – 6
. . . . . 5 – 12
GETPOPSN . . . . . 12 – **9**
GETTHEAD . . . . . 12 – 9
GETVERS . . . . . 12 – 13
Gibbs' phenomenon . . . . . G – 8
GIPSY . . . . . G – 8
global fringe fitting algorithm . . . . . G – 8
GNU . . . . . 2 – 10
GO . . . . . 3 – 1
. . . . . 12 – 4
Going AIPS . . . . . *1 – 3*
. . . . . 12 – 12, 18, 21, 22
graphics functions . . . . . 6 – **29**, **29**, **30**
graphics overlay plane . . . . . G – 8
graphics plane . . . . . G – 9
gray-scale display . . . . . G – 9
gray-scale memory plane . . . . . G – 9
GRCLEAR . . . . . 6 – 19
GRDROP . . . . . 11 – 1
GREAD . . . . . 6 – 19
Green Book . . . . . G – 9
green screen . . . . . G – 9
GREYS . . . . . 6 – 9, **9**, 9
. . . . . 8 – 14
. . . . . F – 3
gridding convolution function . . . . . G – 9
gridding correction function . . . . . G – 9
GRINDEX . . . . . 11 – 1
GRIPE . . . . . 11 – 1
. . . . . Z – 3
. . . . . G – 9
gripe file . . . . . G – 9
GRLIST . . . . . 11 – 1
GROFF . . . . . 6 – 19
GRON . . . . . 6 – 19
GWRITE . . . . . 5 – 34
. . . . . 6 – 19, 28

— H —

HA2TI . . . . . 5 – 3
Hanning smoothing function . . . . . G – 9
hard copy . . . . . 13 – 29
. . . . . G – 9
hardware mount . . . . . G – 9
header . . . . . 8 – 9
header record . . . . . G – 9
HELP . . . . . 3 – 1, 3, 12
. . . . . 6 – 5
HELP ADVERBS . . . . . 3 – 3, 12
HELP ARRAYS . . . . . 3 – 12
help file . . . . . G – 9
HELP NEWTASK . . . . . 12 – 12
HELP PROCS . . . . . 3 – 12
. . . . . 12 – 9
HELP PSEUDOS . . . . . 3 – 12
HELP REALS . . . . . 3 – 12
HELP STRINGS . . . . . 3 – 12
HELP VERBS . . . . . 3 – 2, 12
Hermitian function . . . . . G – 9
HGEOM . . . . . 7 – 3, 12
high-frequency VLA data . . . . . D – **1**, **1**, **1**, **1**, **1**
HINOTE . . . . . 3 – 8
HISEQ . . . . . 7 – 16
histogram . . . . . 7 – 4
. . . . . 8 – 6
history file . . . . . 3 – 8
. . . . . F – 3
. . . . . G – 9
HITEXT . . . . . 3 – 8
HOCLN . . . . . V – **6**
Hogbom Clean algorithm . . . . . G – 9
HOLIS . . . . . V – **6**
HOLOG . . . . . V – **6**, **6**, **6**
holography . . . . . V – **6**, **6**, **6**, **6**, **6**
host computer . . . . . G – 9
HSA . . . . . C – **1**
hue . . . . . G – 10
HUINT . . . . . 6 – 17, **24**
. . . . . 8 – 14
hybrid mapping algorithm . . . . . G – 10

— I —

I^2S . . . . . G – 11
IBLED . . . . . 5 – **33**
. . . . . 10 – 7
IIS . . . . . G – 10
IM2HEAD . . . . . 3 – 7
IM2TV . . . . . 6 – 19
IM2TX . . . . . 7 – 3
. . . . . 8 – 14
IM2UV . . . . . 10 – 24
IM3HEAD . . . . . 3 – 7
IM4HEAD . . . . . 3 – 7
image . . . . . 13 – 29, 30
. . . . . G – 10
image catalog . . . . . G – 10
image file . . . . . G – 10
image plane . . . . . G – 10
image reconstruction . . . . . G – 10
imaging . . . . . 13 – 30
IMAGR . . . . . 4 – 38, 38
. . . . . 5 – **1**, **3**, **4**, **5**, **6**, **8**
. . . . . 5 – **8**, **9**, **11**, **12**, **12**, **12**
. . . . . 5 – **14**, **14**, **16**, 16, **21**, **22**
. . . . . 5 – **23**, 26
. . . . . 6 – 17, 21, 27
. . . . . 8 – 1, 6
. . . . . 9 – 67, 67, 69
. . . . . L – 1, 12
IMCENTER . . . . . 7 – **4**, **7**
IMDIST . . . . . 7 – 4
IMEAN . . . . . 6 – 15
. . . . . 7 – 4
. . . . . 8 – 6
IMFIT . . . . . 6 – 4, 21
. . . . . 7 – 7
IMHEAD . . . . . 3 – 7, 7
. . . . . 4 – 3, 3
. . . . . 8 – 1
. . . . . 9 – 19
. . . . . O – 2, 2
IMLIN . . . . . 8 – 5, 14
. . . . . 10 – 20
IMLOD . . . . . 6 – 1
IMOHEAD . . . . . 3 – 7
import . . . . . Z – 2
IMPOS . . . . . 5 – 33
. . . . . 6 – 22
IMSIZE . . . . . 5 – 6, 14
IMSTAT . . . . . 7 – 4
IMTXT . . . . . 6 – 4
IMVIM . . . . . 6 – 15
IMXY ; IMVAL . . . . . 6 – 22
INCLASS . . . . . 3 – 6
. . . . . 12 – 2
INDXR . . . . . 9 – 7, **14**
. . . . . 10 – 1, 11
. . . . . O – 24
information . . . . . 13 – 34
INNAME . . . . . 3 – 6
. . . . . 12 – 2
INP . . . . . 3 – 3
INPFIT . . . . . 7 – 7
INPUTS . . . . . 3 – 9
inputs file . . . . . G – 10
INSEQ . . . . . 3 – 6
. . . . . 12 – 2
instrumental polarization . . . . . G – 10
intensity . . . . . G – 10
interactive . . . . . 13 – 34
Internet . . . . . 3 – 16, 19
INTYPE . . . . . 3 – 6
invisible distribution . . . . . G – 11

IPL . . . . . G – 11
IRING . . . . . 6 – 17
isoplanaticity assumption . . . . . G – 11
ISPEC . . . . . 6 – 4, 13
. . . . . 7 – 9
. . . . . 8 – 14

— J —

JMFIT . . . . . 6 – 4, 21
. . . . . 7 – 7
jobs . . . . . Z – 4

— K —

KLEENEX . . . . . 11 – 1, 3
KNTR . . . . . 6 – 9, 9
. . . . . 8 – 10
. . . . . 10 – **23**, **23**
. . . . . F – 3

— L —

l_1 solution algorithm . . . . . G – 11
l_2 solution algorithm . . . . . G – 11
LASTEXIT . . . . . 3 – 9
LAYER . . . . . 6 – 17
LGEOM . . . . . 7 – 12
line editor . . . . . G – 11
LISPX . . . . . 6 – 27
. . . . . 7 – **3**
LIST . . . . . 12 – 9
LISTR . . . . . 4 – 6, 11, 11, 20, 21
. . . . . 5 – 12, 28
. . . . . 6 – 3, 4
. . . . . 9 – 19, 48
. . . . . A – 2, 2, 5
. . . . . L – 1
. . . . . O – 2, 7, 7, 19
LISUN . . . . . 4 – **3**
. . . . . 6 – 3
. . . . . 9 – 19
lobe rotator . . . . . G – 11
LOCIT . . . . . 4 – 8
. . . . . 6 – 17
. . . . . V – **2**, **2**, **2**
log in . . . . . 2 – 1
loop gain . . . . . G – 11
LPCAL . . . . . 9 – 59
lpq . . . . . Z – 3
lprm . . . . . 11 – 5
. . . . . Z – 4
LTYPE . . . . . 5 – 12
. . . . . 6 – 9
luminance . . . . . G – 11
LWPLA . . . . . 4 – 16
. . . . . 6 – 5, **5**
. . . . . 8 – 14
. . . . . 10 – 5, 23
. . . . . O – 17

— M —

M2CAT . . . . . 3 – 5
M3CAT . . . . . 3 – 5
M3TAR . . . . . 0 – 20
M4CAT . . . . . 3 – 5
magnetic tape . . . . . 3 – 15, 16, 16
. . . . . 11 – 6
. . . . . 13 – 54
. . . . . F – 4, 4
major cycle . . . . . G – 11
man aips . . . . . 2 – 3
man readline . . . . . 2 – 10
map . . . . . G – 11
MAPPR . . . . . 5 – **5**
MASKS . . . . . 5 – 16
MATCH . . . . . 9 – 7, 14
MATHS . . . . . 7 – 13
MAXFIT . . . . . 7 – 7
Maximum Entropy Method . . . . . 5 – 24
. . . . . G – 11
MAXPIXEL . . . . . 5 – 12
MBDLY . . . . . 9 – **47**, **47**
MCAT . . . . . 3 – 5
. . . . . 5 – 5
MCUBE . . . . . 8 – 7, 7
. . . . . 10 – 16, 16
memory page . . . . . G – 11
memory paging . . . . . G – 11
memory thrashing . . . . . G – 11
message file . . . . . 3 – **4**
. . . . . G – 11
message monitor . . . . . 3 – 4
message terminal . . . . . G – 11
MF2ST . . . . . 6 – 9
. . . . . 7 – 8
MFIMG . . . . . 9 – 16
. . . . . C – 6
MFITSET . . . . . 6 – 21
MFPRT . . . . . 7 – 8
. . . . . 10 – 22
microcode . . . . . G – 11
minimum-match . . . . . 1 – 5
. . . . . 12 – 1
minor cycle . . . . . G – 11
MINPATCH . . . . . 5 – 12
MK3IN . . . . . 0 – 20, **20**, **20**
MK3TX . . . . . 0 – **20**

MkIII . . . . . O – 20, **20**, **20**, **23**, **24**
MOD3D . . . . . 4 – **19**
MODAB . . . . . 7 – 17
model . . . . . 13 – 36
modeling . . . . . 7 – 6, 7, 8, 9, 17
. . . . . 10 – 13, 22, 22
MODIFY . . . . . 12 – 9, 9
MODIM . . . . . 7 – 17
MODSP . . . . . 7 – 17
MODVF . . . . . 8 – 14
MOMNT . . . . . 8 – 14, 14
monitor . . . . . G – 11
Motif . . . . . 2 – 11
MOUNT . . . . . 3 – 15, 16, 16
. . . . . 9 – 7
. . . . . 11 – 6
mouse . . . . . 2 – 3, 11, 12
MOVE . . . . . 3 – 12
MSGSRV . . . . . 2 – 3
. . . . . 3 – 4
MSORT . . . . . 9 – **12**
. . . . . O – 22
MTARS . . . . . 7 – **10**
$m \times n$ map . . . . . G – 11
MULTI . . . . . 4 – 2
multi-band delay . . . . . 9 – 46, **47**
multi-source data set . . . . . 4 – 3
multi-source data sets . . . . . 4 – 1
MWFLT . . . . . 7 – 16
MX . . . . . G – 12

— N —

NAMEGET . . . . . 12 – 2, 13
natural weighting . . . . . 5 – 8
. . . . . G – 12
NBOXES . . . . . 5 – 16
negative bowl artifact . . . . . G – 12
NFS . . . . . 3 – 11
NINER . . . . . 7 – 12, 16
NITER . . . . . 5 – 12, 12
non-closing offset . . . . . G – 12
Nyquist sampling rate . . . . . G – 12

— O —

OBEDT . . . . . 9 – 44
OBIT . . . . . 4 – 3, 30
OBITIMAG . . . . . 5 – 40
OBITMAP . . . . . 5 – 40
OBITPEEL . . . . . 5 – 40
OBITSCAL . . . . . 5 – 40
Object-based tasks . . . . . 13 – 39
OBOXFILE . . . . . 5 – 16
OBPLT . . . . . 9 – 18
obsolete . . . . . 13 – 37
OBTAB . . . . . 9 – **18**
OCAT . . . . . 3 – 5
OFFPSEUD . . . . . 6 – 20
OFFTRAN . . . . . 6 – 20
OFFZOOM . . . . . 6 – 19
OFLAG . . . . . 0 – **2**
OFMCOLOR . . . . . 6 – **20**
OFMCONT . . . . . 6 – 20
OFMDIR . . . . . 6 – 20
OFMGET . . . . . 6 – 20
OFMLIST . . . . . 6 – 4
OFMPL . . . . . 6 – **20**
OFMSAVE . . . . . 6 – 20
OFMSTRCH . . . . . 6 – **20**
OFMTWEAK . . . . . 6 – 20
OFUDG . . . . . 12 – 18
OFUDGE . . . . . 12 – 21
OGEOM . . . . . 7 – 12
. . . . . 8 – 14
. . . . . 10 – 16, **16**
OHGEO . . . . . 5 – 22
. . . . . 7 – 3, 12
. . . . . 8 – 14
OKCOLORS . . . . . 6 – 9
ONEBEAM . . . . . 5 – 6, 14
one-dimensional . . . . . 13 – 38
on-the-fly . . . . . 10 – **2**, **3**, **4**, 7, 12, 16
. . . . . 10 – 16, 16
OOCAL . . . . . 5 – **26**
OOP . . . . . 13 – 39
OOSUB . . . . . 5 – 23, 33
opacity . . . . . 7 – 2
operating system . . . . . G – 12
optical . . . . . 6 – 15
. . . . . 13 – 40
OTF . . . . . 10 – **2**, **3**, 7, 12, 16, 16
OTFBS . . . . . 10 – 2, 4
OTFIN . . . . . 10 – **2**
OTFUV . . . . . 10 – **2**, **3**, **3**
OUTCLASS . . . . . 3 – 6
. . . . . 12 – 2
OUTFILE . . . . . Z – 2
OUTNAME . . . . . 3 – 6
. . . . . 12 – 2
OUTPRINT . . . . . 6 – 3
. . . . . 12 – 5, 16
. . . . . Z – 4
OUTPUTS . . . . . 3 – 9
OUTSEQ . . . . . 3 – 6
. . . . . 12 – 2
OVRSWTCH . . . . . 5 – 14

— P —

page . . . . . G – 12
page swapping . . . . . G – 12
Paley–Wiener theorem . . . . . G – 12
PANEL . . . . . V – **6**, **6**
paraform tasks . . . . . 12 – 18, 19, 19, 21, 21
. . . . . 13 – 40
PATGN . . . . . 7 – 15
. . . . . 10 – 24, 24

PBCOR . . . 7 – 15
PBEAM . . . V – **6**, **6**
PCABL . . . 9 – 35
PCAL . . . 4 – **32**, 32, **32**
. . . 9 – **61**
. . . A – 5
. . . L – 1
PCASS . . . 9 – 35
PCAVG . . . 9 – 35
PCCOR . . . 9 – 35, 35
PCDIV . . . 9 – 35
PCEDT . . . 9 – 35
PCFIT . . . 9 – 35
PCFLG . . . 9 – 35
PCHIS . . . 6 – 17
PCLOD . . . 9 – 35, 35
PCNTR . . . 6 – 9, 9, 9
PCPLT . . . 9 – 35
PCRMS . . . 9 – **35**
PCVEL . . . 4 – 36
PDLT . . . V – **11**
PDPLT . . . 4 – **32**
. . . V – **11**, **11**
PEEK . . . V – **5**
PEELR . . . 5 – **32**
PGEOM . . . 7 – 12
phase referencing . . . 9 – 28, **46**, 48
phase tracking center . . . G – 12
phaseless reconstruction . . . G – 12
physical memory . . . G – 12
pillbox . . . G – 12
PIPEAIPS . . . 4 – 1
pixel . . . G – 12
pixel coordinates . . . G – 12
PLAYR . . . 6 – 24
PLCOLORS . . . 6 – 5, 9
PLCUB . . . 6 – 13
. . . 8 – 10, 14
PLGET . . . 3 – 9
. . . 6 – **5**
plot file . . . 6 – **5**, **8**, **8**, **9**, **9**, **13**
. . . 6 – **13**, **15**, **17**
. . . 10 – **5**, 23
. . . F – 3, 4
. . . G – 12
PLOTC . . . 6 – **8**
PLOTR . . . 6 – 17
plots . . . 13 – 40
PLRFI . . . 6 – **8**
. . . 7 – 17
. . . 9 – 5
PLROW . . . 6 – 13
. . . 8 – 14
point source response . . . G – 12
point spread function . . . G – 12
points per beam . . . G – 12
polarization . . . 4 – 24, **32**, 32, **32**, 32, 32
. . . 5 – 24
. . . 6 – 9
. . . 7 – 1, 1
. . . 9 – 1, 2, 15, 33, 45, **59**
. . . 9 – **60**, **61**
. . . 13 – 44
. . . A – 5
. . . O – 20
POPS . . . 3 – 1, 2
. . . 12 – **1**, **1**, **6**, **8**, **9**, **9**
. . . 12 – **9**, **9**, **12**, **12**
. . . 13 – 45
. . . G – 12
POPS procedure . . . G – 13
POPS symbols . . . G – 13
POPSYM . . . 3 – 1
portability . . . 1 – 1
POSSM . . . 4 – 10, 16, 32
. . . 6 – 8, 16
. . . 8 – 14
. . . 9 – 20, 26, 38, 38, 48, 48
. . . 10 – 5
. . . L – 1, 1
. . . O – 17
PostScript . . . 6 – 5, **5**, 24, 28
. . . 8 – 10, 14
. . . 10 – 5, 23
. . . 12 – 16
. . . Z – 2
PPAPP . . . 4 – **32**
primary beam correction . . . G – 13
primary data file . . . G – 13
primary-beam corrections . . . 5 – 21
principal solution . . . G – 13
print job . . . Z – 3
PRINTER . . . 3 – 4
. . . 6 – 3, 4, 5
. . . 11 – 5
. . . Z – 1
PROC . . . 12 – 9
procedure . . . 12 – **9**, 9, **9**, **12**, **12**
. . . 13 – 48
. . . G – 13
PROFL . . . 6 – 13
prolate spheroidal wave function . . . G – 13
prompt character . . . G – 13
PRPLT . . . 6 – **9**
PRTAB . . . 4 – 1, 15
. . . 6 – 4
. . . 9 – 19
. . . O – 7
PRTAC . . . 6 – 5

PRTAN . . . . . 4 – 6
. . . . . 6 – 4, 16
. . . . . 9 – 19
. . . . . A – 2
. . . . . L – 1
. . . . . O – 22
PRTCC . . . . . 5 – 23, 26
. . . . . 6 – 4
PRTHI . . . . . 3 – 8
. . . . . 6 – 5
PRTIM . . . . . 5 – 12
. . . . . 6 – 4
PRTMSG . . . . . 3 – 4
. . . . . 5 – 2
. . . . . 6 – 4
PRTOF . . . . . 0 – **2**
PRTSD . . . . . 10 – 1, 5
PRTSY . . . . . 4 – **30**
. . . . . 6 – 4
PRTTP . . . . . 3 – 16
. . . . . 5 – 1
. . . . . 6 – 5
. . . . . A – 2
PRTUV . . . . . 5 – 12
. . . . . 6 – 3
Prussian helmet Clean algorithm . . . . . G – 13
pseudo array processor . . . . . 3 – **21**
pseudo-AP . . . . . G – 13
pseudo-array processor . . . . . G – 13
pseudo-color display . . . . . G – 13
pseudo-continuum u-v data file . . . . . G – 13
PSEUDOVB . . . . . 12 – 18
pseudo-verb . . . . . 12 – 1
. . . . . 13 – 49
. . . . . G – 13
PSF . . . . . G – 13
PUTHEAD . . . . . 8 – 9
. . . . . 9 – 15
. . . . . 10 – 1

— Q —

Q2HEADER . . . . . 3 – 7
Q3HEADER . . . . . 3 – 7
Q4HEADER . . . . . 3 – 7
QBEAM . . . . . V – **6**
QCREATE . . . . . 12 – 4
QGETVERS . . . . . 12 – 13
QHEADER . . . . . 3 – 7
QINP . . . . . 3 – 9
QOHEADER . . . . . 3 – 7
Q-routine . . . . . G – 13
QUACK . . . . . 9 – 32
. . . . . A – 2
QUEUES . . . . . 12 – 5
quick boot . . . . . G – 14
QUXTR . . . . . 7 – **10**

— R —

RAINLEVS . . . . . 6 – 5
RANCID . . . . . G – 14
RASHIFT . . . . . 5 – 6, 6, 14
READISK . . . . . 3 – 19
. . . . . 12 – 17
readline . . . . . 2 – **10**
re-boot . . . . . G – 14
REBOX . . . . . 5 – 16
. . . . . 6 – 21
REBYTE . . . . . 3 – 19
. . . . . 12 – **17**
RECAT . . . . . 3 – 6
recipe . . . . . 2 – 12
. . . . . 3 – 21, 22, 22, 22
. . . . . 4 – 40, 40, 41, 41, 41
. . . . . 5 – 40
. . . . . 7 – 18, 18
. . . . . 8 – 17, 18, 18, 18
. . . . . 9 – 72, 72
. . . . . 10 – 25, 25, 26, 26
. . . . . 13 – 87, 87, 88, 88, 88, 88
. . . . . 13 – 88
. . . . . A – 6, 6, 6
. . . . . B – 8, 8
. . . . . C – 15, 16, 16, 16, 16
. . . . . D – 7, 8, 8, 8, 8
. . . . . L – 14
. . . . . F – 6, 6
. . . . . Z – 7, 8, 8, 8
REFANT . . . . . 5 – 30
REFLG . . . . . 4 – 11, **28**
regularization method . . . . . G – 14
regularization parameter . . . . . G – 14
REHEX . . . . . 3 – 18
. . . . . 12 – 3
re-IPL . . . . . G – 14
REMAG . . . . . 7 – 3, 5
. . . . . 10 – 24
REMDISK . . . . . 12 – 14
REMHOST . . . . . 3 – 16
REMOTE tapes . . . . . 3 – **16**
remote user . . . . . 12 – **14**, **14**, **15**, **16**
REMOVIE . . . . . 8 – 10
REMTAPE . . . . . 3 – 16
RENAME . . . . . 3 – 6
. . . . . 10 – 21
RENUMBER . . . . . 3 – 6
REROAM . . . . . 6 – 18
RESEQ . . . . . 9 – **14**
residual delay . . . . . G – 14
residual fringe rate . . . . . G – 14
resolution . . . . . G – 14
RESOU . . . . . 9 – **14**
RESTORE . . . . . 3 – 9

restoring beam . . . . . G – 14
RETURN . . . . . 12 – 9
REWAY . . . . . 4 – 11, **36**, 36
. . . . . 5 – 8
. . . . . L – 1
REWIND . . . . . 3 – 16
RFARS . . . . . 7 – **10**
RFI . . . . . 6 – 3
RFLAG . . . . . 4 – 11, **28**, 28, **28**
. . . . . 5 – 33
. . . . . L – 1, 1, 1, 1
RGBMP . . . . . 6 – 17, 24
. . . . . 8 – 14
RIRMS . . . . . 6 – **8**
. . . . . 7 – **17**
RLDIF . . . . . 4 – **32**, **32**, **32**
. . . . . A – 5
RLDLY . . . . . 4 – **32**
. . . . . 9 – **60**, 60
. . . . . L – 1
RM . . . . . 7 – 10
RM2PL . . . . . 6 – **27**
. . . . . 7 – **10**
. . . . . 8 – 14
RMFIT . . . . . 6 – 27
. . . . . 8 – 14
RMSD . . . . . 7 – 3, 5
. . . . . 8 – 14
roam . . . . . G – 14
ROAMOFF . . . . . 6 – 18
ROBUST . . . . . 5 – 8, 8
ROTATE . . . . . 5 – 5, 6, 14
rotation measure . . . . . 7 – 2
rotation-measure synthesis . . . . . 7 – **10**
RSPEC . . . . . 6 – 13
RSTOR . . . . . 5 – **12**, 24
RUN . . . . . 3 – 18
. . . . . 12 – 3, 3
. . . . . 13 – 50
run file . . . . . G – 14

— S —

SABOX . . . . . 5 – 16
SAD . . . . . 6 – 4, 9, 9, 27
. . . . . 7 – 8
. . . . . 10 – 22, 22
sampling theorem . . . . . G – 14
saturation . . . . . G – 14
SAVE . . . . . 3 – 9
. . . . . 12 – 5, 9
SBCOR . . . . . 0 – **24**
SCAT . . . . . 3 – 5, 11
SCIMG . . . . . 5 – 26
. . . . . 6 – 27
. . . . . 9 – 67, 69
SCLIM . . . . . 6 – 17
SCMAP . . . . . 5 – 1, 26
. . . . . 6 – 27
. . . . . 9 – 67, 69
scratch . . . . . G – 14
scratch file . . . . . G – 14
SCRDEST . . . . . 3 – 11
. . . . . 11 – 4
screen editor . . . . . G – 15
scroll . . . . . G – 15
SDCAL . . . . . 10 – 11
SDCLN . . . . . 5 – 24
. . . . . 6 – 27
SDGRD . . . . . 10 – **14**, **14**, 14, 16
SDIMG . . . . . 10 – **14**
SDLSF . . . . . 10 – **12**
SDMOD . . . . . 10 – 13
SDTUV . . . . . 10 – **4**
SDVEL . . . . . 10 – 1, 3, **12**
self-calibration . . . . . 5 – **26**, **26**, **26**, **28**, **28**, **30**
. . . . . 5 – **30**
. . . . . 8 – 4
. . . . . 9 – 67, 67, 69
self-calibration algorithm . . . . . G – 15
self-calibration gain solution algorithm . . . . G – 15
SERCH . . . . . 8 – 14
SETFC . . . . . 5 – 5, 6, 14
. . . . . L – 12
SETJY . . . . . 4 – 6, 6, **15**, 15
. . . . . 9 – 41
. . . . . A – 2
. . . . . D – 1
. . . . . L – 1
. . . . . O – **16**
SETMAXAP . . . . . 3 – **21**
SETRGBL . . . . . 6 – 5, 30
SETSLICE . . . . . 6 – 21
. . . . . 7 – 9
SETXWIN . . . . . 6 – 21
SG2RUN . . . . . 3 – 9, 18
. . . . . 12 – 3, 17
SGUGET . . . . . 3 – 9
SGUINDEX . . . . . 3 – 9
SHADW . . . . . 7 – 16
Shannon sampling theorem . . . . . G – 15
Shannon series . . . . . G – 16
shed . . . . . G – 16
SHOUV . . . . . 6 – 3
. . . . . 9 – 20
SHOW . . . . . 5 – 12
single-dish . . . . . 10 – **1**, **2**, **3**, **4**, **5**, **5**
. . . . . 10 – **7**, **7**, **11**, **11**, **12**, **14**
. . . . . 10 – **14**, **16**, **16**, **16**, **16**, **20**
. . . . . 10 – **20**, **22**, **22**, **24**
. . . . . 13 – 51
SIVPSF . . . . . G – 16
SIZEFILE . . . . . 3 – **21**
SKYVE . . . . . 7 – 12

SL2PL . . . . . 6 – 13, 16
. . . . . 7 – 9, 9
. . . . . 8 – 14
SLFIT . . . . . 7 – 9
SLICE . . . . . 7 – 9
. . . . . 8 – 14
. . . . . G – 16
slice file . . . . . G – 16
SLPRT . . . . . 7 – **9**
SMODEL . . . . . 5 – 28
snapshot . . . . . G – 16
SNBLP . . . . . 6 – **16**
. . . . . C – 10
. . . . . D – 1, 1
SNBPL . . . . . 4 – 24
. . . . . 9 – 23
SNCOP . . . . . 9 – 57
SNEDT . . . . . 4 – 30
. . . . . 5 – 28
. . . . . 6 – 27
. . . . . 9 – 26, 40, 40, 48
. . . . . O – 6
SNFLG . . . . . 4 – 11
. . . . . 5 – 33
. . . . . 9 – 32
. . . . . D – 1
SNIFS . . . . . 6 – 16
SNP2D . . . . . 4 – **36**
SNPLT . . . . . 4 – 11, 20, 21, 24, 30
. . . . . 5 – 26, 28
. . . . . 6 – 16
. . . . . 9 – 23, 34, 48
. . . . . L – 1, 1, 1
. . . . . O – 19
SNRMS . . . . . 4 – **30**
. . . . . 9 – 40
. . . . . C – 4
SNSMO . . . . . 9 – 26, 26, 42, 48
software mount . . . . . G – 16
Solar data . . . . . 4 – 30
. . . . . O – **18**, **19**, **20**
SOLCL . . . . . 0 – 18, 20
SOLINT . . . . . 5 – 30
. . . . . 9 – 48
SOLMODE . . . . . 5 – 28
SOLTYP . . . . . 5 – 28
sort order . . . . . G – 16
source editor . . . . . G – 16
SOUSP . . . . . 4 – **16**, 22
spatial resolution . . . . . G – 16
spawn . . . . . G – 16
SPCOR . . . . . 7 – **15**
SPECR . . . . . 8 – 1
spectral index . . . . . 4 – 15, 15, 16, 22, 32
. . . . . 7 – 2
spectral-line . . . . . 4 – 16
. . . . . 8 – **1**, **1**, **1**, **4**, **5**, **6**
. . . . . 8 – **7**, **7**, **9**, **10**, **14**, **14**
. . . . . 8 – **14**
. . . . . 9 – 2, 41, 42, 42, 45, 57
. . . . . 9 – 57, 57
. . . . . 10 – 3, 3, 7, 7, 12, 20
. . . . . 10 – 20, 22
. . . . . 13 – 51
. . . . . O – 15, 16, 16, 17, 17
SPFIX . . . . . 7 – **3**
SPFLG . . . . . 4 – 11, 11
. . . . . 5 – 33
. . . . . 6 – 27
. . . . . 8 – 1
. . . . . 10 – 7, 7
. . . . . L – 1, 1
. . . . . O – 16
spheroidal function . . . . . G – 16
SPIXR . . . . . 5 – 21
. . . . . 7 – **3**
SPLAT . . . . . 4 – 11
. . . . . L – 1
SPLIT . . . . . 4 – 39, 39
. . . . . 8 – 1, 1, 4
. . . . . 9 – 42, 63
. . . . . A – 5
. . . . . C – 11
. . . . . L – 1
. . . . . F – 6
SPMOD . . . . . 7 – 17
SPRMS . . . . . 6 – **8**
. . . . . 7 – **17**
. . . . . 9 – **65**
SPXMD . . . . . 7 – **3**
SPY . . . . . 3 – 11
SQASH . . . . . 7 – 10
. . . . . 8 – 5, 14
STACK . . . . . 7 – **3**
STALIN . . . . . 3 – 8
STARS . . . . . 6 – 9
STARTTV . . . . . 2 – **12**
. . . . . 3 – 4
. . . . . 6 – **18**, **19**, 29
statistics . . . . . 7 – 4
status . . . . . 3 – 6
STEPLEVS . . . . . 6 – 5, 30
Stokes' parameters . . . . . G – 16
Stokes' visibility functions . . . . . G – 17
STORE . . . . . 3 – 9
STQUEUE . . . . . 5 – 12
structure of *AIPS* . . . . . 1 – 5
SUBIM . . . . . 3 – 12
. . . . . 8 – 7
sub-image . . . . . G – 17

SUBMIT . . . . . 12 – 5
sub-task . . . . . G – 17
sum . . . . . 7 – 2
SUMIM . . . . . 7 – 3
super-resolution . . . . . G – 17
support . . . . . G – 17
support width . . . . . G – 17
SVLBI . . . . . 9 – 2, 18, 44, 57
SVPSF . . . . . G – 17
SYHIS . . . . . 4 – **30**
Synthesis Imaging in Radio Astronomy 5 – 4, 24, 26, 30, 30
7 – 12
8 – 1
9 – 48
O – 2, 18
G – 17
synthesized beam . . . . . G – 17
SYPRT . . . . . 4 – **30**
6 – 4
SYSOL . . . . . 4 – 30
SysPower . . . . . 4 – **30**

— T —

TAB . . . . . 2 – 10
TABED . . . . . 4 – 1
O – 7, 16
TABGET . . . . . 12 – 9
tables . . . . . 4 – 1
9 – 7, 23, 71, 71
13 – 53
TABPUT . . . . . 4 – 15
9 – 13
TACOP . . . . . 4 – 1
6 – 5
8 – 4
L – 1, 1
O – 16, 17
TAFFY . . . . . 12 – 18, 21
TAFLG . . . . . 4 – 1
5 – 23
TAMRG . . . . . 4 – 1
O – 23
tape . . . . . 3 – 15, 16, 16
11 – 6
13 – 54
F – 4, 4
tape blocking efficiency . . . . . G – 18
taper . . . . . G – 18
TAPES . . . . . 3 – 15
TAPLT . . . . . 5 – 23, 26
6 – 16
9 – 18, 18
TAPPE . . . . . 4 – 1
5 – 33
8 – 4
L – 1
TARS . . . . . 7 – **10**
TASAV . . . . . 4 – 32
9 – 26
A – 2, 2, 5
C – 7
F – 6
O – 18
task . . . . . 3 – **1**
12 – 18, **19**, 19, 20, 21, 21
G – 18
tasks . . . . . 13 – 55
TASRT . . . . . 4 – 1
t-b order . . . . . G – 18
TBDIF . . . . . 6 – 4
TBIN . . . . . 12 – 16
TBOUT . . . . . 12 – 16
TDEPEND . . . . . 5 – 16
TECOR . . . . . 4 – **32**
9 – 2, **16**, **16**
C – 6
TEK screen . . . . . G – 18
TEK4012 . . . . . G – 18
TEKSRV . . . . . 2 – 3
6 – 5, 29
television . . . . . 1 – 1
Telex 6250 tape drive . . . . . G – 18
TELL . . . . . 5 – 12
12 – 5
TEPLT . . . . . 4 – **32**
9 – 16
C – 6
terminal page . . . . . G – 18
terminal scroll . . . . . G – 18
text editor . . . . . G – 18
text file . . . . . G – 18
text files . . . . . 3 – 17, 17
TFILE . . . . . 0 – 20
TGET . . . . . 3 – 9
12 – 4, 5
TGINDEX . . . . . 3 – **9**
TGUINDEX . . . . . 3 – 9
Third NRAO Synthesis Imaging Summer School G – 18
thrashing . . . . . G – 18
TI2HA . . . . . 5 – 3
time smearing . . . . . G – 18
time-baseline order . . . . . G – 18
TKASLICE . . . . . 6 – 30
TKERASE . . . . . 6 – 29
TKGUESS . . . . . 6 – 30
TKMODEL . . . . . 6 – 30
TKPL . . . . . 6 – 5, 29
7 – 4
12 – 15
TKPOS . . . . . 6 – 29
TKRESID . . . . . 6 – 30
TKSET . . . . . 6 – 30

TKSLICE . . . . . 6 – 30
. . . . . 8 – 14
TKVAL . . . . . 6 – 30
TKWIN . . . . . 6 – 29
TKXY ; IMVAL . . . . . 6 – 29
TLCAL . . . . . V – **1**
TPHEAD . . . . . 3 – 16
. . . . . 6 – 1
. . . . . 8 – 1
TPMON . . . . . 3 – 15, 16
. . . . . 11 – 6
TPUT . . . . . 3 – 9
. . . . . 12 – 4, 5
trackball . . . . . G – 18
trackball button . . . . . G – 18
TRANS . . . . . 7 – 3
. . . . . 8 – 5, 9, 10
. . . . . 10 – 20, 20
transfer function . . . . . G – 18
TRC . . . . . G – 18
true color display . . . . . G – 18
TU77 tape drive . . . . . G – 18
TU78 tape drive . . . . . G – 19
TUGET . . . . . 3 – **9**
TV blink . . . . . G – 19
TV cursor . . . . . 2 – 12
. . . . . G – 19
TV functions . . . . . 6 – **5**, **18**, **18**, **20**, **23**, **24**
. . . . . 6 – **24**, **28**
. . . . . 13 – 66, 69
TV image catalog . . . . . G – 19
TV look-up table . . . . . G – 19
TV roam . . . . . G – 19
TV scroll . . . . . G – 19
TV zoom . . . . . G – 19
TV2COLOR . . . . . 6 – **20**
TV3COLOR . . . . . 6 – 24
TV3LOD . . . . . 6 – 24
. . . . . 8 – 14
TVALL . . . . . 6 – 18, 18, 20
. . . . . 7 – 7, 9
. . . . . 8 – 10
TVANOT . . . . . 6 – 23
TVASLICE . . . . . 6 – 26
TVBANDW . . . . . 6 – **20**
TVBLINK . . . . . 6 – 24
TVBOX . . . . . 5 – 16
. . . . . 6 – 21
TVCLEAR . . . . . 6 – 19
TVCOLORS . . . . . 6 – 9
TVCOMPS . . . . . 6 – 26
TVCPS . . . . . 6 – 5, 28
. . . . . 8 – 10
. . . . . 10 – 7
. . . . . Z – 2
TVCUBE . . . . . 6 – 24
. . . . . 8 – 10
TVDATA . . . . . 6 – 19
TVDIST . . . . . 6 – 22
TVFACTS . . . . . 6 – 19
TVFLG . . . . . 4 – 11, 11
. . . . . 5 – 12, 33
. . . . . 6 – 27
. . . . . A – 2
. . . . . O – **9**, **9**, **9**, **9**, **9**, 18
TVFLUX . . . . . 6 – 22
TVGUESS . . . . . 6 – 26
TVHELIX . . . . . 6 – 18, 20
TVHLD . . . . . 6 – **18**
TVHUEINT . . . . . 6 – **24**
. . . . . 8 – 14
TVHUI . . . . . 6 – 17, 24
. . . . . 8 – 14
TVIEW . . . . . 6 – **27**
. . . . . 7 – **10**
. . . . . 8 – **14**
TVILINE . . . . . 6 – 23
TVINIT . . . . . 6 – 19
TVLABEL . . . . . 6 – 9, 23
TVLAYOUT . . . . . V – **6**
TVLINE . . . . . 6 – 23
TVLOD . . . . . 6 – 18, 24
. . . . . 7 – 9
TVMAXFIT . . . . . 6 – 22
TVMODEL . . . . . 6 – 26
TVMOVIE . . . . . 6 – 24
. . . . . 8 – 10
TVNAME . . . . . 6 – 21
TVOFF . . . . . 6 – 19
TVON . . . . . 6 – 19
TVPHLAME . . . . . 6 – 18, 20
TVPL . . . . . 6 – 5
TVPOS . . . . . 6 – 19
TVPSEUDO . . . . . 6 – 18, 20
TVRESID . . . . . 6 – 26
TVRESTOR . . . . . 2 – **12**
. . . . . 6 – **18**
TVRGB . . . . . 6 – 24
. . . . . 8 – 14
TVROAM . . . . . 6 – 18
TVSAD . . . . . 6 – **27**
. . . . . 7 – **8**
TVSET . . . . . 6 – 26
. . . . . 7 – 9
TVSLICE . . . . . 6 – 26
. . . . . 7 – 9, 9
. . . . . 8 – 14
TVSPC . . . . . 6 – 27
. . . . . 7 – 9, **10**
. . . . . 8 – **14**
. . . . . L – 12, 12
TVSPX . . . . . 6 – 27
. . . . . 7 – **3**
TVSTAR . . . . . 6 – 23
TVSTAT . . . . . 6 – 22
. . . . . 7 – 4

TVTRAN . . . . . 6 – 24
TVTRANSF . . . . . 6 – 20
TVWEDGE . . . . . 6 – 18
TVWIN . . . . . 6 – 21
. . . . . 7 – 4
TVWINDOW . . . . . 7 – 7
TVWLABEL . . . . . 6 – 18, 23
TVZOOM . . . . . 6 – 19
twm . . . . . 2 – 11
TX2IM . . . . . 7 – 3
. . . . . 8 – 14
TXPL . . . . . 6 – 5, 5
. . . . . 12 – 15
TYAPL . . . . . 4 – 10, 30
. . . . . 5 – 8
. . . . . L – 1
. . . . . O – 6
TYCOP . . . . . 4 – 30
. . . . . 9 – 40
TYSMO . . . . . 4 – 30
. . . . . 9 – 40
. . . . . O – 6

— U —

U2CAT . . . . . 3 – 5
U3CAT . . . . . 3 – 5
U4CAT . . . . . 3 – 5
UBAVG . . . . . 5 – 3
UCAT . . . . . 3 – 5
. . . . . 4 – 3
. . . . . 5 – 2
. . . . . O – 2
UFLAG . . . . . 5 – **33**
. . . . . 6 – 27
undefined pixel . . . . . 7 – 3, 5, 5
. . . . . 8 – 14
uniform weighting . . . . . 5 – 8
. . . . . G – 19
UNIX . . . . . G – 19
UNQUE . . . . . 12 – 5
UNSPX . . . . . 7 – **3**
UOCAT . . . . . 3 – 5
URL . . . . . 1 – 1
. . . . . 3 – 12
user-coded task . . . . . G – 19
USUBA . . . . . 9 – 7, **13**
UTESS . . . . . 5 – 24
utilities . . . . . 13 – 70
UV . . . . . 13 – 71
u-v coverage . . . . . G – 19
u-v data file . . . . . G – 19
u-v data flag . . . . . G – 19
u-v FITS format . . . . . G – 19
u-v measurement distribution . . . . . G – 19
u-v sampling distribution . . . . . G – 20
u-v taper function . . . . . G – 20
u-v transfer function . . . . . G – 20
u-v weight function . . . . . G – 20
UV2MS . . . . . 4 – 2
UVADD . . . . . 5 – 26
UVAVG . . . . . 9 – 64, 64
. . . . . O – 23
UVBAS . . . . . 8 – 5
UVBOX . . . . . 5 – 8
UVBXFN . . . . . 5 – 8
UVCON . . . . . 7 – **17**
UVCOP . . . . . 3 – 12
. . . . . 4 – 6, 39
. . . . . 8 – 1
. . . . . 9 – 32
. . . . . A – 2
. . . . . L – 1, 1
. . . . . F – 6
. . . . . V – 6
UVCRS . . . . . 9 – 65
UVDIF . . . . . 5 – 26
. . . . . 6 – 3
UVFIL . . . . . 12 – 18, 21
UVFIT . . . . . 7 – 17
. . . . . 9 – 65
UVFLG . . . . . 4 – 11
. . . . . 5 – 33
. . . . . 8 – 1
. . . . . 10 – 7
. . . . . A – 2
. . . . . L – 1, 1
. . . . . O – 7
UVFND . . . . . 5 – 12, 33
. . . . . 6 – 3
UVFRE . . . . . 8 – **1**
UVHGM . . . . . 6 – 8
UVHIM . . . . . 6 – **8**
UVHOL . . . . . V – **6**, **6**, **6**, **6**
UVIMG . . . . . 6 – 8
UVLIN . . . . . 8 – 1, 5
UVLOD . . . . . 4 – 2, 5
. . . . . 5 – 2
UVLSF . . . . . 8 – 1, 1, 5
UVMAP . . . . . 6 – 27
. . . . . 8 – 6
UVMLN . . . . . 5 – 33
. . . . . 8 – 1
UVMOD . . . . . 7 – 17
UVPLT . . . . . 4 – 24
. . . . . 5 – 11, 12, 26, 33
. . . . . 6 – 8
. . . . . 9 – 65
. . . . . 10 – 5
. . . . . A – 2
. . . . . O – 18
UVPRT . . . . . 6 – 3
UVRANGE . . . . . 5 – 8
UVRMS . . . . . 6 – **8**
. . . . . 7 – **17**
. . . . . 9 – **65**
UVSIZE . . . . . 5 – 8

UVSRT . . . . . 5 – 3
. . . . . 9 – 7
. . . . . 10 – 7
. . . . . O – 22
UVSUB . . . . . 5 – 23, 33
. . . . . L – 12
UVTAPER . . . . . 5 – 8
UVWTFN . . . . . 5 – 8, 8

— V —

Varian printer . . . . . G – 20
Variational Method . . . . . G – 20
VBCAL . . . . . 9 – 65
VBGLU . . . . . 5 – **3**
. . . . . 9 – **14**
VBMRG . . . . . 9 – **14**
VBRFI . . . . . 6 – **8**
. . . . . 7 – 17
. . . . . 9 – 5
velocity . . . . . 9 – 41
VERB . . . . . 12 – 18
. . . . . 13 – 78
. . . . . G – 20
verbs . . . . . 3 – **2**
Versatec printer . . . . . G – 20
VERSION . . . . . 3 – 9
. . . . . 12 – 4
Very Long Baseline Interferometry — . . . . . –
Techniques and Applications . . . . . G – 20
VGET . . . . . 3 – **9**
VGINDEX . . . . . 3 – **9**
VH2RL . . . . . 4 – **32**
VHCAL . . . . . 4 – **15**
VHCALIB . . . . . 4 – **15**
VHDIF . . . . . 4 – **32**
. . . . . L – 1
VHMOD . . . . . L – 1
vi . . . . . G – 20
virtual memory page . . . . . G – 20
virtual memory page swapping . . . . . G – 20
virtual memory storage . . . . . G – 20
visibility phase tracking center . . . . . G – 20
VLA . . . . . 9 – 31
. . . . . 13 – 84
. . . . . C – **7**, 13, 13
. . . . . O – 1, 2, 2, 2, 2, 6
. . . . . O – 7, 7, 9, 9, 9, 9
. . . . . O – 9, 9, 16, 17, 18, 19
. . . . . O – 20
VLACALIB . . . . . 4 – 20, 21
. . . . . A – 2
VLACLCAL . . . . . A – 2
VLAMP . . . . . 9 – **31**
VLANT . . . . . 4 – **8**
. . . . . D – 1
VLAPROCS . . . . . 4 – 1, 20, 32
. . . . . A – 2
. . . . . O – 1
VLARESET . . . . . 4 – 24
VLARUN . . . . . A – **1**, 1
. . . . . O – **1**
VLATECR . . . . . 4 – **32**
. . . . . L – 1
VLBA . . . . . 9 – **1**, **2**, **2**, **7**, **7**, **7**
. . . . . 9 – **7**, **11**, **13**, **14**, **15**, **18**
. . . . . 9 – **29**, **30**, **35**, 41
. . . . . C – **1**, **7**, **10**, **13**, **14**, **15**
. . . . . O – 24
VLBAAMP . . . . . 9 – **40**
. . . . . C – 3, **7**
VLBABPSS . . . . . C – 3, 7
VLBACALA . . . . . C – 12, **13**
VLBACCOR . . . . . 9 – **34**
. . . . . C – 3
VLBACPOL . . . . . C – 3
VLBACRPL . . . . . 9 – **20**
. . . . . C – 3, **7**
VLBAEOPS . . . . . 9 – **17**
. . . . . C – 3, 6
VLBAFIX . . . . . 9 – **7**, **11**, **12**, **25**
. . . . . C – 3
VLBAFPOL . . . . . 9 – **15**
VLBAFRGP . . . . . C – 3, **10**
VLBAFRNG . . . . . C – 3, **10**
VLBALOAD . . . . . 9 – **7**
. . . . . C – 3, **6**
VLBAMPCL . . . . . 9 – **36**
. . . . . C – 3, **7**
VLBAMPHC . . . . . 9 – **5**
. . . . . C – **4**
VLBAPANG . . . . . 9 – **33**
. . . . . C – 3, **10**
VLBAPCOR . . . . . 9 – **35**
. . . . . C – 3, **7**
VLBAPIPE . . . . . 9 – **5**, 5, **5**
. . . . . C – 1, **4**, 4, **4**
VLBARFI . . . . . 9 – **5**
. . . . . C – 4
VLBARUN . . . . . 9 – **5**
. . . . . C – **4**, **4**
VLBASNPL . . . . . C – 3, **7**, **10**
VLBASUMM . . . . . 9 – **19**
. . . . . C – 3, **6**
VLBATECR . . . . . 9 – **16**
. . . . . C – 3, 6
VLBAUTIL . . . . . 9 – 5, 5, 7, 11, 12, 13
. . . . . 9 – 15, 16, 17, 19, 20, 23
. . . . . 9 – 25, 33, 34, 35, 36, 40
. . . . . 9 – 48
. . . . . C – 4, 4, **6**, **7**, 7, **10**
. . . . . C – **10**
. . . . . O – 24

VLBCD . . . . . 9 – 35
VLBI . . . . . 9 – **1**, **1**, **2**, **2**, **7**, **7**
. . . . . 9 – **7**, **7**, **7**, **11**, **12**, **13**
. . . . . 9 – **14**, **15**, **16**, **17**, **16**, **18**
. . . . . 9 – **19**, **20**, **20**, **21**, **23**, **23**
. . . . . 9 – **25**, **26**, **26**, **26**, **29**, **29**
. . . . . 9 – **30**, **31**, **32**, **32**, **33**, **35**
. . . . . 9 – **35**, **36**, **36**, **40**, **41**, **42**
. . . . . 9 – **42**, **44**, **45**, **46**, **46**, **48**
. . . . . 9 – **48**, **48**, **48**, **48**, **55**, **55**
. . . . . 9 – **57**, **57**, **59**, **61**, **61**, **63**
. . . . . 9 – **63**, **64**, **65**, **67**, **67**, **69**
. . . . . 9 – **71**
. . . . . 13 – 85
. . . . . C – 1, **6**, 6, **7**, 7, 7
. . . . . C – 10, **10**, 11, **13**, 13, 13
. . . . . C – 14, 15
. . . . . O – **20**, **22**, **23**, **24**
VLBRF . . . . . 9 – **5**
VLBSY . . . . . 9 – 35
VLOG . . . . . 9 – 29
VM . . . . . G – 20
VMS . . . . . G – 20
VPFLG . . . . . 4 – 11
VPLOT . . . . . 5 – 23
. . . . . 6 – 8
. . . . . 9 – **21**, **21**, 26, 32, 48, 48
. . . . . 9 – 48, 64
. . . . . D – 1
VPUT . . . . . 3 – **9**
VTESS . . . . . 5 – 24
. . . . . 6 – 27
VTEST . . . . . 10 – **12**

— W —

WAIT . . . . . 3 – 1
wc . . . . . Z – 4
wedge . . . . . G – 20
WETHR . . . . . 4 – 11
. . . . . 6 – 16
wild-card . . . . . 12 – 2
window Clean . . . . . G – 21
window management . . . . . 2 – **11**
window manager . . . . . 2 – 1
WIPER . . . . . 4 – 11, 11
. . . . . 5 – **33**, **33**
. . . . . 6 – 27
workfile . . . . . 5 – 12, 23, 33
working set size . . . . . G – 21
workstation . . . . . 2 – 1
World-Wide Web . . . . . 1 – 1
. . . . . 3 – 12
WRTDISK . . . . . 3 – 19
. . . . . 11 – 2
. . . . . 12 – 17
WRTPROCS . . . . . 12 – 17
WTSUM . . . . . 10 – 14, 21

— X —

XAS . . . . . 2 – 3, **12**
. . . . . 6 – 5, 18, 24
XBASL . . . . . 6 – 27, 30
. . . . . 8 – 5, 14
.Xdefaults . . . . . 2 – 2, 3
. . . . . 3 – 4
XG2PL . . . . . 6 – **27**
. . . . . 7 – **10**
. . . . . 8 – 14
XG2XG . . . . . 7 – **10**
. . . . . 8 – 14
XGAUS . . . . . 6 – 27
. . . . . 7 – 10
. . . . . 8 – 14, 14
XHELP . . . . . 3 – 12
XMOM . . . . . 8 – 14, 14
. . . . . 10 – 22
XPLOT . . . . . 6 – 27, 30
. . . . . 7 – 10
. . . . . 8 – 10, 14
.Xresources . . . . . 2 – 2, 3
XSMTH . . . . . 8 – 14
xv . . . . . Z – 2
x-y order . . . . . G – 21

— Y —

Y-routine . . . . . G – 21

— Z —

ZAMAN . . . . . 6 – 27
. . . . . 7 – 10
. . . . . 8 – 14
ZAP . . . . . 3 – 6
. . . . . 5 – 12
. . . . . 11 – 2, 4
ZEMAN . . . . . 6 – 27
. . . . . 7 – 10
. . . . . 8 – 14
ZEROSP . . . . . 5 – 11
zero-spacing flux . . . . . G – 21
zoom . . . . . G – 21
Z-routine . . . . . G – 21