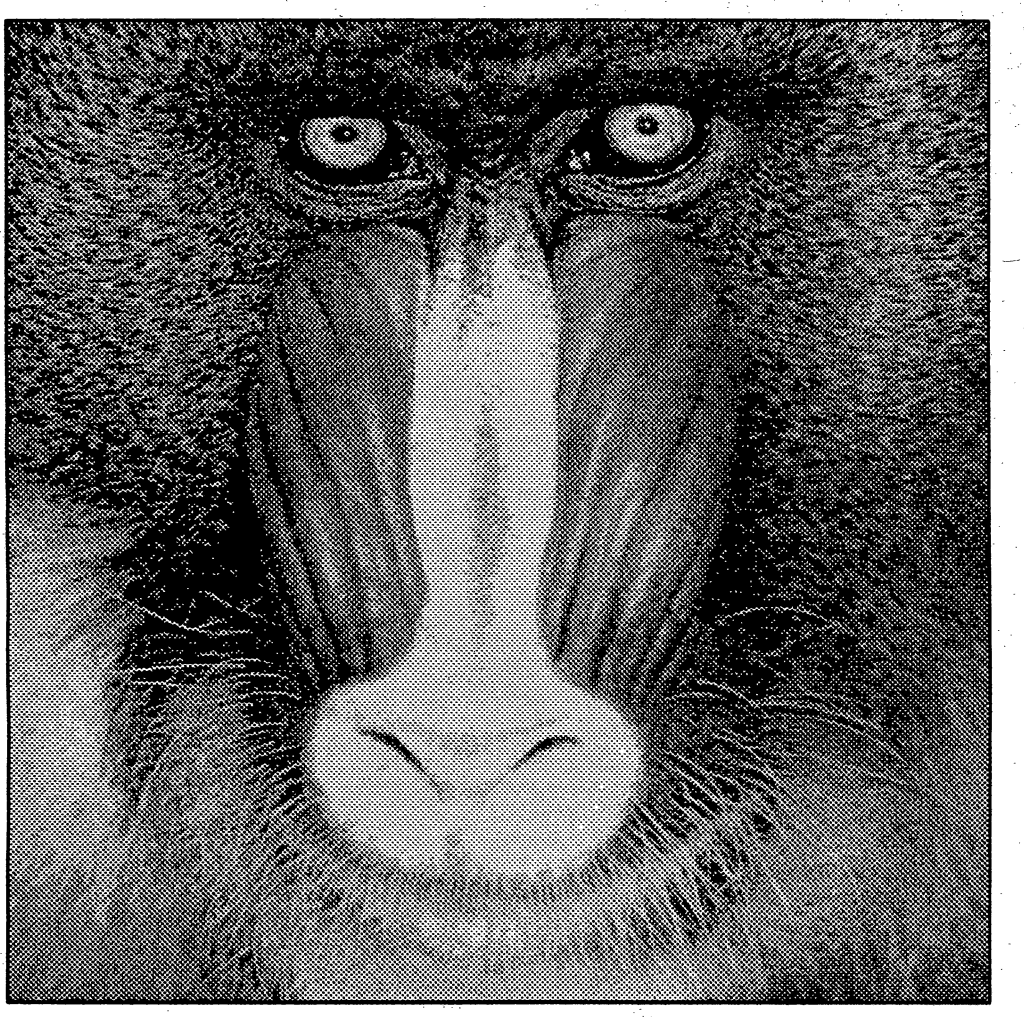
AIPS COOKBOOK

15-Oct-1990



The National Radio Astronomy Observatory
Edgemont Road
Charlottesville, VA 22903–2475
Operated by Associated Universities, Inc.
under contract with the National Science Foundation

ACKNOWLEDGEMENTS

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 plotted on a QMS Lasergrafix 800 by the *AIPS* tasks `GREYS` and `QMSPL`. It is the red portion of the digitized image of a Mandrill which has become a standard in the image-processing field.

The plots at the ends of Sections 8, 9 and 10 were also generated on a QMS Lasergrafix 800 using various *AIPS* plotting tasks and `QMSPL`. The data displayed in § 8 were provided by S. Baum, A. Bridle, T. Heckman, G. Miley and W. van Breugel. The data displayed in § 9 were provided by A. Bridle and R. Perley and those displayed in § 10 by D. Wells. The editors thank these people for providing their data prior to publication.

This *COOKBOOK* itself was typeset by Eric W. Greisen and Bill Junor using the TEX program, version 2.95, initially developed by Donald E. Knuth (*The TEXbook*, 1984, Addison-Wesley Publishing Company, Reading, Massachusetts). The output of TEX was then printed on an IBM Laserwriter. The editors are grateful to Knuth for this program and, especially, for his decision to place it in the public domain.

or at the VLA. A faster method is to use your own copying machine. *AIPS* sites which have a PostScript printer may also have files on disk from which up-to-date copies of the *CookBook* may be printed. Consult your local *AIPS* Manager for details on how to do this. Much of the *AIPS* documentation, including the *CookBook*, is now available to the "World-Wide Web" so that it may be examined and retrieved over the Internet (start with "URL" `http://info.cv.nrao.edu/aips/aips-home.html`).

This edition of the *CookBook* is issued in a ring-binder format with a chapter-nased 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.

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. The first volume is intended for applications programmers, while the second volume is needed by programmers developing *AIPS* for new peripheral devices or computers. Printed copies of *Going AIPS* may be obtained by writing to the NRAO Computer Division in Charlottesville.

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, call the *AIPS* programmers in Charlottesville. 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 § 12.3). 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. A "Gripes" database has been constructed to improve "gripe" response times and to aid the programmers in reviewing outstanding complaints and suggestions.

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 *uv* data. Chapter 5 explains the basic operations required to make images. Chapters 3, 4 and 5 contain the information which beginning users are most likely to require. Chapters 6 through 9 introduce the basic *AIPS* tools for improving images, for making interactive and hard-copy displays of them, and for analyzing them. Chapter 10 contains 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. Similarly, Chapter 11 is aimed primarily at users of VLB interferometers. Chapter 12 deals with exiting from *AIPS* and tidying up your disk areas to be polite to the other users. Chapter 13 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 14 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 9. Chapter 15 contains lists of all available routines broken down by categories. Appendix G presents Fred Schwab's Glossary of radio astronomy data processing terminology. Appendix Y gives some useful recipes for estimating disk files sizes and for saving data and images on tape.

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 § 15.

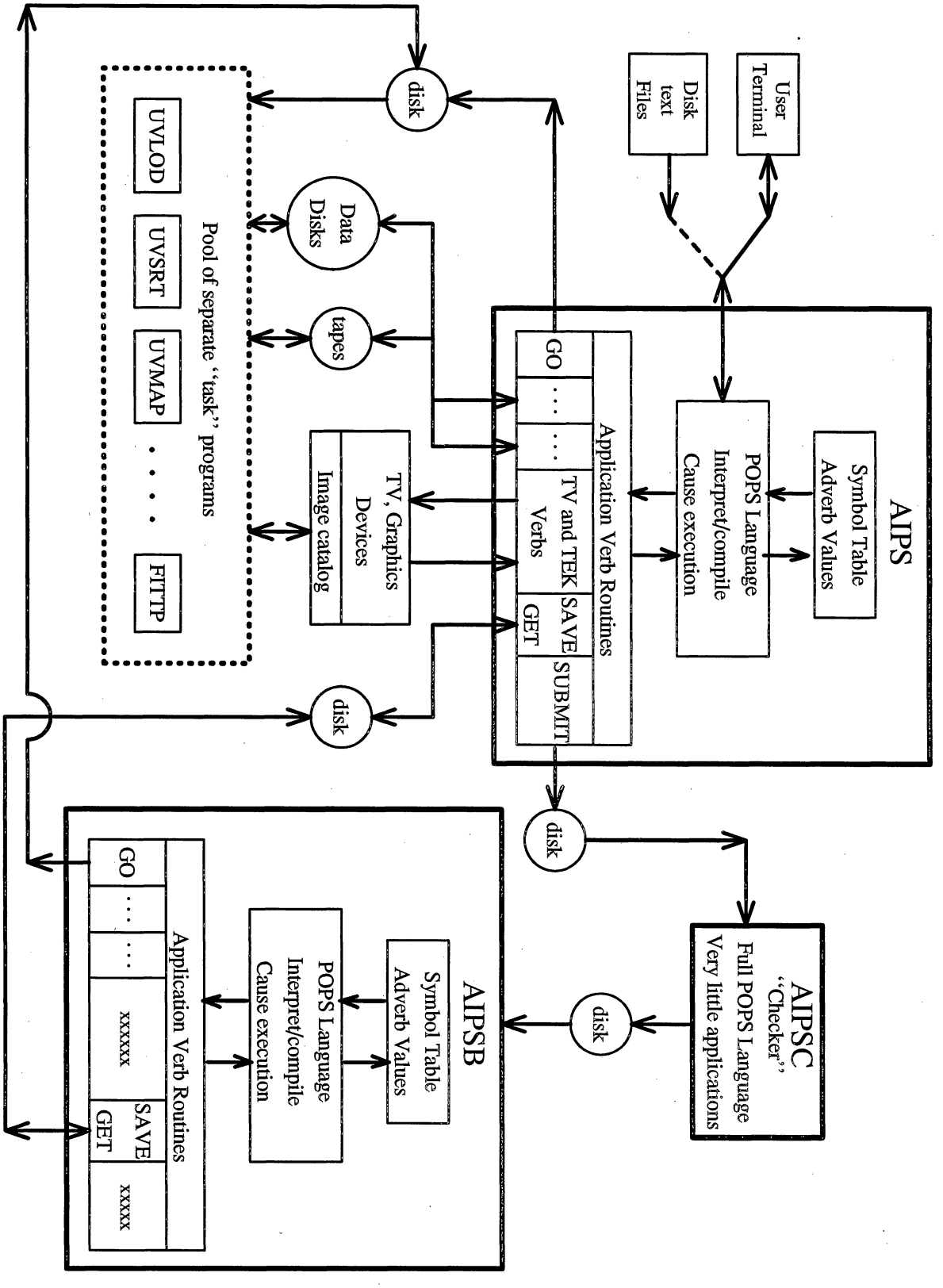
1.3.3. Fonts and what they signify

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 key on the terminal."

The symbol § means Section and refers to the various chapters and sub-chapters of this *CookBook*.

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 `SUBMIT`ting 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.


AIPS
Symbol Table
Adverb Values
POPS Language
Interpret/compile
Cause execution
Application Verb Routines
GO
TV and TEK Verbs
SAVE
GET
SUBMIT
User Terminal
Disk text Files
disk
Data Disks
tapes
TV, Graphics Devices
Image catalog
Pool of separate "task" programs
UVLOD
UVSRT
UVMAP
FITTP
AIPSC
"Checker"
Full POPS Language
Very little applications
AIPSB
Symbol Table
Adverb Values
POPS Language
Interpret/compile
Cause execution
Application Verb Routines
GO
xxxxxx
SAVE
GET
xxxxx

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 $\subset_R$ 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:

```
> EXIT ⊂R
```

which will get out of **AIPS**. Then once at the system prompt (Unix), you can type exit $\subset_R$ (lowercase!) to make the window go away. For the **XAS** *AIPS* TV, 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. In Charlottesville, the aips account uses the **twm** or **tvtwm** window managers on all systems (Sun, IBM, DEC Alpha, Ultrix); on these, pressing the left mouse button on the root window (the background) will reveal an **EXIT X11** option. At the AOC, the IBM systems use the Motif window manager (**mwm**) and the left mouse button will get you the menu with the **Exit Windows** option. Logout, if necessary, by typing exit $\subset_R$. Log back in using the procedure given above.

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** has no definition and is harmless, while **CTRL Y** does not do anything useful. **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 $\subset_R$.

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

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 version, YOU have to keep track of your site's disk configuration!

The format for these files is all the same: a list of disk (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. There is usually one site-wide required disk specified in $NET0/NETSP.

There is also a $NET0/NETSP file that is maintained by the AIPS manager and controls TIMDEST and aips user-number access to the disks. You will get error messages if your private .dadevs file includes AIPS data areas ("disks") that are not in the NETSP file.

The order of data areas, i.e. which is disk 1, etc., is determined by the order in the DADEVS.LIST or .dadevs 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 man pages on rsh, rhosts, and hosts.equiv. The tp= option uses rsh to issue commands to remote hosts.

`tv=[tvdisp][:][tvhost]`

TV display server to use instead of the default. The AIPS startup script attempts to deduce which workstation the user is sitting in front of. This may not be the same as the machine on which AIPS is to be run if, for example, the user has done an rlogin to another machine within a terminal window.

The full syntax of this option is TV=tvdisp:tvhost, where tvdisp is the name of the machine on which the TV display server (XAS, XVSS, or SSS), Tektronix graphics server (TEKSRV), and text message server (MSGSRV) are to run, and tvdisp indicates the machine to which the DISPLAY environment variable should point for XAS.

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 **AIPSn** 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 13 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 $\subset_R$ will list the major *POPS* language features on your terminal, and Chapter 14 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 $\subset_R$ and ABOUT GENERAL $\subset_R$. See also § 15 for the **15JUL94** 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:

> HELP *task_name* $\subset_R$ where *task_name* is the name of a task you want to use.

This writes helpful text on your terminal about the purpose of the task and about its input parameters.

> TASK '*task_name*' $\subset_R$ to make *task_name* the default for later commands; note the quote ' marks.

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

> INP $\subset_R$ to review the parameter values that you have set and

> GO $\subset_R$ to send the task into execution.

You may also specify which task you want to execute by an immediate argument, *e.g.*, GO UVSRT C_R 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 C_R to kill the task named by **TASK**, or

> ABORT *task_name* C_R 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.

The current full list of tasks may be obtained on your terminal (or workstation window) by typing ABOUT TASKS C_R. Since this list runs for many pages, you may wish to direct the output to the line printer (with DOCRT = -2 C_R) or to consult the **15JUL94** list in § 15 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 C_R. (The sequence GO *verb_name* C_R 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 C_R 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 C_R, but the output lists only the names. To find out more, type ABOUT VERBS C_R 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) **15JUL94** list printed in § 15 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

The output of **IMHEAD** and **QHEAD** 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-1993
# 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       1  -2.0000000E+00    1.00-1.0000000E+00    0.00
FREQ         4   2.2228990E+10    0.50 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  .ICLN  .   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 **QHEAD** 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*.

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 previously as important, 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    Pseudoverbs 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
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
```

Even veteran *AIPS* users should use HELP WHATSNEW $^{C}_{R}$ when a new release of *AIPS* is installed on their computer. When it is current, this file provides brief descriptions of recent developments in *AIPS*. Reading it may bring pleasant surprises and avoid unpleasant ones! More detailed descriptions of new developments in *AIPS* can be found in the *AIPSLetter* published by the NRAO with each *AIPS* software release. 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 **baboon.cv.nrao.edu**) and via the Internet and the "World-Wide Web" starting with "URL" (Universal Resource Location) **http://info.cv.nrao.edu/aips/aips-home.html**).

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, 520 Edgemont Road, Charlottesville, VA 22903-2475) or send electronic mail to **aips-mail@nrao.edu** for further information about these services.

> DENSITY *dddd* $\subset_R$ to set the density to *dddd* bpi if needed.

> MOUNT $\subset_R$ 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 $\subset_R$ 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 $\subset_R$

> DENSITY *dddd* $\subset_R$ to set the density to *dddd* bpi if needed.

> INTAPE *n* ; MOUNT $\subset_R$ 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.

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**, and **FITTP**. **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.

calibration-related symbols is given in § 15.6, 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.

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 VLA observations on or after January 1, 1988, use **FILLM** to read the VLA archive tape (or a copy thereof) directly.
2. For VLA observations before January 1, 1988, use **FILLM** on a translation of the original archive tape. All VLA archive data are being copied to Exabyte tapes, and are being translated to the modern format as needed. Contact the VLA data analysts (phone 505–835–7359, e-mail **analysts@nrao.edu**) to obtain a translated copy of any old observing files.
3. For an *AIPS* multi-source data set written to a FITS tape during an earlier *AIPS* session, use **UVLOD** or **FITLD** to read the tape.
4. For single-source data sets that are already on disk and are very similar in structure, use **UV2MS** on one of them to create a multi-source data set, and then on each of the others to append them to that multi-source data set. Each of the input data sets should have the same number of polarizations, IFs, spectral channels, and "random parameters." **UV2MS** also makes no corrections for differences in observed source positions or frequencies. After all are appended, use **UVSRT** to put the data in time-baseline order and **INDXR** to make an index and initial (null) calibration file.
5. For single-source data sets that are already on disk and are not sufficiently similar in structure for the method above, use **MULTI** on each single-source file to convert to multi-source format. Then use **DBCON** to concatenate the individual multi-source files into one big multi-source file. Finally use **UVSRT**, if needed, to put the data in time-baseline order and **INDXR** to make an index and initial (null) calibration file.

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. Translation tasks for the Westerbork Synthesis Telescope (**WSLOD**) and the Australia Telescope (**ATLOD**) are available from the Dutch and Australians, respectively, but are not distributed by the NRAO with the normal *AIPS* system.

4.1.1. Reading from a VLA archive tape using FILLM

To load a *uv* data file to disk from a VLA archive tape, you must (hardware) mount the tape on a tape drive and then (software) mount the tape inside the **AIPS** program. See § 3.9 for a discussion of this process. It is strongly recommended that you begin by obtaining an index of the contents of your data tape. Reference dates, time ranges, file numbers, frequencies observed, and the like are reported in the index and are needed to guide the actual loading of the data. To print an index of the VLA archive tape, use task **PRTTP**:

> TASK 'PRTTP' ; INP C_R	to review the inputs needed.
> NFILES 0 C_R	to start at the beginning of tape.
> PRTLEV 0 C_R	to give complete summaries; only **PRTLEV = -3** actually affects the output (adversely).
> DOCRT FALSE C_R	to send output to the line printer.
> GO C_R	to index the tape.

Typical inputs to **FILLM** might be:

> TASK 'FILLM' ; INP C_R	to review the inputs needed.
> OUTNA ' ' C_R	to take the default output file name.
> OUTDI 3 C_R	to write the data to disk 3 (one with enough space).
> DOUVCOMP TRUE C_R	to write visibilities in compressed format to save disk space.
> DOCONCAT TRUE C_R	to concatenate files if this is second tape.
> DOALL TRUE C_R	to include data from all frequency bands, source qualifiers, and numbers of spectral channels, writing as many output data sets as needed.
> VLAOBS 'AC238' C_R	to select only data from observing program AC238. The default is to load data from *all* programs.
> NFILES 4 C_R	to skip the first 4 files on the archive tape.
> CPARM 30,0 C_R	to average the data for for 30 seconds; default is no averaging.
> CPARM(6) 1 C_R	to select VLA sub-array 1.
> CPARM(7) 2000 C_R	to have observations within 2 MHz be regarded as being at the same frequency.
> CPARM(8) 2 C_R	to use a 2-minute interval for the CL table; default is 5 min.
> DPARM 0 C_R	to have no selection by specific frequency.
> REFDATE '*dd/mm/yy*' C_R	to specify the day, month and year 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 tape.
> 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. The default is to include all times.
> INP C_R	to review the inputs.
> GO C_R	to run the program when you're satisfied with inputs.

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** for general guidance about the choice of averaging time given the size of the required field of view and the observing bandwidth.

Some words of warning about the use of **NFILES** are appropriate here. VLA archive tapes used to contain 3 tapes files for every actual data file. Most archive tapes today do not have these ANSI standard-label files, but still tend to begin with a header (non-data) tape file. If you set the **NFILES** adverb carefully based on the index printed by **PRTTP**, you should have no problem with these "excess" tape files.

FILLM is designed to read all your data from tape in one pass. All data meeting the selection criteria will be read from the input tape and filled into a *uv* multi-source file. Three selection criteria are always active: (a) **TIMERANG**, if non-zero, will restrict processing of data to the specified range of times (with respect to **REFDATE**); (b) **VLAOBS**, will restrict processing to the specified VLA observing code (which will be set to the code in the first valid data record if you do not specify it); and (c) **CPARM(6)** will restrict processing to the specified VLA sub-array (which, if you do not specify it, will be set to 1 if **VLAOBS** is not specified or to the first sub-array belonging to **VLAOBS**). All other selection criteria may be overridden by setting **DOALL** to **TRUE**.

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. I n 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.

If your data are on multiple tapes, you can write them all into the same data file by specifying **DOCONCAT** on the second (and subsequent) runs of **FILLM**.

A significant reduction in the disk space used may be achieved using the compressed format invoked by adverb **DOUVCOMP**; this factor is 1.89 for 2-IF continuum data and approaches 3.0 for line data. Most, but not quite all, tasks can process compressed *uv* data. The task **UVCMP** allows you to change the formats of *uv* data sets between *compressed* and *uncompressed*, if required to use one of these aberrant tasks.

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 do this, (a) use the **QUAL** adverb in **FILLM**, assuming that you have used separate qualifiers in **OBSERVE** for each

* *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)

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 tape 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. The default (**CPARM(7)** = 0) assigns the same FQ number to all data in spectral-line data sets. If **CPARM(7)** is positive, a scan will be assigned to an existing **FQ** number if

$$|\nu_{current} - \nu_{firstFQ}| < \mathbf{CPARM(7)}$$

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 negative 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 -1 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. The default setting of **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 is prepared to try to read past up to 50 parity or other tape errors. Do not be alarmed by a few warning messages, especially at the end of tape on old 9-tarck, half-inch tapes. These are relatively normal and will cause not harm.

If **FILLM** is executing correctly, your message terminal will report the number of your observing program, the VLA archive tape format revision number, and then the names of the sources as they are found on the tape. Once **FILLM** has completed, you can find the data base on disk using:

```
> INDI 0 ; UCAT CR
```

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 05-FEB-1994 12:34:16
```

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

```
> INDI 3 ; GETN 1 ; IMH CR
```

This should produce a listing like:

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.

As a practical note, setting the start and stop times of the data for your experiment with **TIMERANG** will cause **FILLM** to read all valid data up to the stop time you have specified and then exit normally. This way, it will not read to the end of the VLA archive file. The default action of **FILLM** (to read to the end of the tape if **TIMERANG = 0**) can be particularly annoying if your data are at the very start of a large archive data file.

When the data are successfully loaded to disk, type DISMOUNT ↵ to dismount your input tape.

4.1.2. Reading from a FITS tape with FITLD

FITLD is used to read FITS-format tapes into *AIPS*. It recognizes images, single- and multi-source *uv* data sets, and the special FITS *uv*-data tables produced by the VLBA correlator. 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** 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 VLBA-format files.)

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

> TASK 'FITLD' ; INP ↵	to review the inputs needed.
> INTAPE *n* ↵	to specify the tape drive for input from tape.
> INFILE '*filename*' ↵	if the input is from a FITS disk file (see § 3.10.3).
> DOUVCOMP TRUE ↵	to write visibilities in compressed 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.

FITLD is the equivalent of **FILLM**, but for output from the VLBA, rather than the VLA, correlator. The data-selection adverbs **SOURCES**, **QUAL**, **CALCODE**, and **TIMERANG** and the table-control adverbs **CLINT** and **FQTOL** are used, for VLBA-format data only, in **FITLD** in ways similar to the data-selection and control adverbs of **FILLM**. See Chapter 11 for more specific information.

4.2. Record keeping and data management

4.2.1. Calibrating data with multiple FQ entries

In general an observing run with the VLA, especially a spectral-line run, will result in a *uv* data file containing multiple **FQ** entries. In early versions of the *AIPS* software, the different **FQ** entries would automatically have been placed in different physical files. Now, **FILLM** allows you to place all of them in the same file. 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 (as described in § 4.1.1). 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.

A practical note: it is often useful to have used different qualifiers for different frequencies. This gives you another "handle" on the data. Unfortunately, not all programs use the **QUAL**, or even the **CALCODE**, adverb.

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:

> RESTORE 0 ↵	to set all inputs to null (default) values.
> TASK 'LISTR' ; INP ↵	to review the inputs needed.
> INDI *n*; GETN *m* ↵	to select the data set, $n = 3$ and $m = 1$ in **FILLM** example above.
> TPUT CALIB ↵	to store null values for later use with **CALIB**.
> OPTYP 'SCAN' ↵	to select scan summary listing.
> INP ↵	to review the inputs for **LISTR**.
> GO ↵	to run the program when the inputs are set correctly.

Note that the **RESTORE 0** sets the adverbs to select all sources and all times and to send printed output to the printer rather than the terminal. It is also very useful to have a printed summary of your antenna locations, especially a list of which ones you actually ended up using. To do this, enter

> NCOUNT 0 ↵	to do all antennas
> INVERS *n* ↵	to do sub-array *n*
> GO PRTAN ↵	to print the list and a map of antenna locations.

> OPTYPE 'LIST' C_R	to select column listing format
> ANTEN *a1* , 0 C_R	to select one reliable antenna to display.
> BASEL 0 C_R	to select all baselines to this antenna.
> SOURCES ' ' ; CALCODE '*' C_R	to select all calibrator sources only.
> TIMER 0 C_R	to select all times.
> STOKES 'RR C_R	to examine only one Stokes at a time.
> BIF 1 C_R	to specify the "AC" IFs (note that IFs must be listed separately).
> FQID 1 C_R	to select **FQ** number 1 (note that **FQ** numbers must also be done separately).
> DOCRT 132 C_R	to see full width display on the terminal. Use your window manager to stretch the window to $\geq$ 132 characters width.
> DOCALIB -1 C_R	to turn off calibration.
> DPARM 0 C_R	to select amplitudes with no averaging.
> INP C_R	to re-check *all* the inputs parameters.
> GO C_R	to start the task.

The task will prompt you for a C_R after each "page full" of output. When you have seen enough, enter Q. This display will let you determine whether start-of-scan problem infects your data and, if so, how badly. If it is rare, forget it for now and use manual flagging methods later if needed. If it is widespread, use the *AIPS* task **QUACK**:

> TASK 'QUACK' C_R	
> SOURCES ' ' C_R	to select all sources.
> TIMER 0 C_R	to select all times.
> ANTENNAS 0 C_R	to select all antennas.
> FLAGVER 1 C_R	to insert flagging information in FG table 1.
> OPCODE 'BEG' C_R	flag first **APARM(2)** min of each scan.
> REASON 'BAD START OF SCAN' C_R	reason for the flagging.
> APARM 0 , 1/6 , 0 C_R	flag first 10 seconds of each scan.
> GO C_R	

The display generated above will also allow you to determine quickly which antennas are absent, which antennas are present but dead, and, with more careful examination, which antennas are flaky and may need special consideration. "Dead" antennas are visible in this display as columns with small numbers — columns that differ by factors of two or so from the others are generally fine. To be thorough, it is probably best to check the other IF:

> BIF 2 C_R	to specify the "BD" IFs.
> GO C_R	to run the program again.

as well as **STOKES = 'LL'**.

To remove the dead antennas, run **UVFLG**. For example, if antennas 6, 9, and 22 were bad for the full run in both IFs and Stokes, they could be deleted with

> TASK 'UVFLG' ; INP C_R	to select the editor and check its inputs.
> TIMER 0 C_R	to select all times.
> BIF 1 ; EIF 2 C_R	to specify the "AC" and "BD" IFs.

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. 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. **CALIB** also has an option that will allow you to make use of CLEAN component models for primary flux density calibrators, in which case this step is redundant. An example of the inputs would be:

```
> TASK 'SETJY' ; INP CR
> SOURCES '3C286' , ' ' CR          if you used 3C286 as the source name.
> ZEROSP 7.41 , 0 CR                I flux 7.41 Jy, Q, U, V fluxes 0.
> BIF 1 ; EIF 1 CR                  selects "AC" IF.
> INP CR                            to review inputs.
> OPTYPE ' '                        use values given in ZEROSP.
> GO CR                             when inputs okay.
> BIF 2 ; EIF 2 CR                  selects "BD" IF.
> ZEROSP 7.46, 0 CR                 I flux 7.46 Jy at the 2nd IF, Q, U, V fluxes 0.
> GO CR
```

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.

If you wish, you can let **SETJY** calculate the fluxes, in which case it is able to do both IFs together.

```
> TASK 'SETJY' ; INP CR
> SOURCES '3C286' , ' ' CR          if you used 3C286 as the source name.
> BIF 1 ; EIF 2 CR                  will calculate for both "AC" and "BD" IFs.
> OPTYPE 'CALC' CR                  perform the calculation.
> APARM(2) = 0 CR                   to use the VLA "1990" coefficients.
> INP CR                            to review inputs.
> GO CR                             when inputs okay.
```

4.3.3. First pass of the gain calibration

You are now ready to begin the actual calibration process. The first step is to determine a set of antenna gain solutions for both the primary and secondary flux density calibrator(s) within any *uv* limits that apply. This will be your first contact with **CALIB**, the central task in the *AIPS* calibration program. Most of the complexity of **CALIB** can be hidden using the procedure **VLACALIB**. Before attempting to invoke this procedure, you must first

```
> RUN VLAPROCS CR                   to compile the procedures.
```

Both 3C286 and 3C48 are resolved by the VLA in some configurations and frequencies. Point models for these sources are therefore only accurate over a limited range of baseline length. The range of baseline length used can be controlled by the adverb **UVRANGE**. If there are too few baselines to a given antenna, accurate solutions may not be possible; therefore, it is frequently necessary to limit the antennas used to the inner antennas on each arm. (The antenna pad numbers which include the order number from the array center on each arm can be determined by running **PRTAN**; see §4.2.2.) **CALIB** may fail to produce any valid solutions if antennas with no data are included in the solution. The VLA Calibrator Manual suggests the following sets of **UVRANGE** (in kiloλ) and inner number of antennas.

TVFLG is the one used for continuum and channel-0 data from the VLA, while **SPFLG** is only used to check for channel-dependent interference. **SPFLG** is useful for spectral-line editing in smaller arrays, such as the Australia Telescope. (The redundancy in the spectral domain on calibrator sources helps the eyes to locate bad data.) **IBLED** is more useful for small arrays such as those common in VLBI experiments. All three tasks have the advantage of being very specific in displaying the bad data. Multiple executions should not be required. However, they require you to look at each IF, Stokes, channel (or baseline) separately (unless you make certain broad assumptions) and they 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 highly corrupted data (say with considerable RF interference, significant cross-talk between antennas, or erratic antennas) **TVFLG** is definitely preferred. 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 and power of **TVFLG** are not needed. In such cases, **LISTR** with **OPCODE = 'MATX'** can efficiently find scans with erroneous points.
- 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 **IBLED** 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 **TVFLG**.

4.4.1. 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 C_R	to review the inputs needed.
> INDI *n*; GETN *m* C_R	to select the data set, $n = 3$ and $m = 1$ above.
> SOURCES ' ' ; CALCODE '*' C_R	to select calibrators.
> TIMER 0 C_R	to select all times.
> ANTENNAS 0 C_R	to list data for all antennas.
> OPTYPE 'MATX' C_R	to select matrix listing format.
> DOCRT FALSE C_R	to route the output to printer, not terminal.
> DPARM 3 , 1 , 0 C_R	amplitude and *rms*, scalar scan averaging.
> BIF 1 C_R	to specify the "AC" IFs (note that IFs must be listed separately).
> FQID 1 C_R	to select **FQ** number 1 (note that **FQ** numbers must also be done separately).
> INP C_R	to review the inputs.
> GO C_R	to run the program when inputs set correctly.
> BIF 2 C_R	to specify the "BD" IFs.
> WAIT ; GO C_R	to wait for the previous execution to finish and then run the program again.

Procedure **VLACALIB** may be used for your gain calibration sources as you did previously.

> INDI *n* ; GETN *m* ↵	to select the data set, $n = 3$ and $m = 1$ above.
> CALSOUR = '3C286' , ' ' ↵	to name primary flux calibrator(s) individually, or
> CALCODE = '*' ↵	to select any non-blank calibrator code.
> 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.
> MINAMPER 10 ↵	display warning if baseline disagrees in amplitude by more than 10% from the model.
> MINPHERR 10 ↵	display warning if baseline disagrees by more than 10° of phase from the model.
> FREQID 1 ↵	use **FQ** number 1.
> INP VLACALIB ↵	to review inputs.
> VLACALIB ↵	to make the solution and print results.

If there are different *uv* ranges for different sources, then re-run the procedure with changed parameters, such as:

> CALSOUR = '*cal*1' , '*cal*2' , '*cal*3' ↵	to name secondary flux calibrator(s).
> ANTENNAS 0 ↵	solutions for all antennas.
> UVRANGE 0 ↵	no *uv* limits, or range if any, in kiloλ.
> INP VLACALIB ↵	to review inputs.
> VLACALIB ↵	to process the secondary calibrators.

At this time, you should use as many antennas and as large a **UVRANGE** as you can for each calibrator, consistent with its spatial structure.

4.5.3. Final (?) initial global calibration

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 table with **SNVER**, **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 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**.

To use **CLCAL**:

> TASK CLCAL ; INP ↵	to review the inputs.
> SOURCES '*sou*1' , '*sou*2' , '*sou*3' , ... ↵	sources to calibrate, ' ' means all.
> CALSOUR '*cal*1' , '*cal*2' , '*cal*3' , ... ↵	calibrators to use for **SOURCES**.
> FREQID *n* ↵	use **FQ** number *n*.
> OPCODE 'CALI' ↵	to combine **SN** tables into a **CL** table.
> GAINVER 1 ↵	to select the input **CL** table; 1 for first calibration, 2 if there are baseline corrections.
> GAINUSE 2 ↵	to select the output **CL** table; 2 is normal, 3 if there are baseline corrections.

> BIF 2 ↵ to specify the "BD" IFs.

> WAIT ; GO ↵ to run the program again after the first job is done.

The matrix average amplitudes for the calibrators in this listing should be very close to the values that you entered with **SETJY** (or which were derived by **GETJY**) and the phases in all rows and columns for these sources should be very close to zero.

If some rows and columns of the amplitude matrices are systematically different from the mean, the amplitude calibration for the associated antennas is imperfect. The reasons for this should be investigated. More flagging of visibilities, scans, or antennas, may be indicated. If the phase matrices have all elements near zero, then the phase calibration is in good shape. If some calibrators have discrepant phases and others do not, the discrepant calibrators are probably resolved. Note that you will not be able to detect errors in the assumed positions of your calibrators at this stage if you have used the usual 2-point interpolation of the calibration. Position errors in the calibrators have now become phase and position errors in the target sources.

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 ↵ to verify the data set to be reset.

> VLARESET ↵ to reset **SN** and **CL** tables.

4.6. Polarization calibration

The calibration of visibility data sensitive to linear polarization involves two distinct operations: (1) determing and correcting the data for the effects of imperfect telescope feeds and (2) removing any systematic phase offsets between the two systems of orthogonal polarization. These two components of polarization calibration will be considered separately.

The effective feed response is parameterized most generally by its polarization ellipticity and the orientation of the major axis of that ellipse. For the VLA, it appears to be adequate to make the simpler assumption that each polarization is corrupted by a small complex gain times the orthogonal polarization.

In general, the polarization of the calibrator(s) to be used to determine the feed parameters will not be known *a priori* and must be determined along with the feed parameters. Observations of a given source (or sources) over a wide range ($\geq 90°$) of parallactic angles is necessary to separate calibrator polarization from the feed parameters. Task **LISTR** may be used to determine the parallactic angles at which data have been taken:

> TASK 'LISTR' ↵

> SOURCES '*cal*1' , '*cal*2' , '*cal*3' , ... ↵ list all calibrators to be used.

> INEXT 'CL' ↵ to determine parallactic angle at times in **CL** table.

> INVER 1 ↵ **CL** version 1.

> FREQID *n* ↵ to use **FQ** number *n*.

> OPTYPE 'GAIN' ↵ to use gain table rather than visibility data.

> DPARM = 9 , 0 ↵ to display parallactic angle.

> INP ↵ to review the inputs.

> GO ↵ to run the program when inputs set correctly.

Step 1: Run **PCAL** on one or more phase calibrator sources observed with a wide range of parallactic angles:

> TASK 'PCAL' C_R	
> CALSOUR *'cal1'* , *'cal2'* , *'cal3'* , ... C_R	list all calibrators to be used.
> TIMERANG 0 C_R	to use all times.
> ANTENNAS 0 C_R	to solve for all antennas.
> UVRANGE *uvmin uvmax* C_R	to set *uv* limits, if any, in kiloλ.
> BIF 1 ; EIF 2 C_R	to do both IFs.
> DOCALIB TRUE C_R	to apply the calibration to the sources (very important!).
> GAINUSE 2 C_R	or 3 or whatever, to use the latest **CL** table.
> CLR2N C_R	to clear **IN2NAME** *etc.* since there is no Clean-image model.
> FREQID *n* C_R	to use **FQ** value *n*; only one polarization solution can be stored.
> PMODEL 0 C_R	to use polarization parameters in the source table.
> SOLINT 2 C_R	to use a 2-minute solution interval; scan averages are usually sufficient.
> SOLTYPE 'APPR' C_R	to use linear approximation model.
> PRTLEV 1 C_R	to display the results and some diagnostic information.
> REFANT n C_R	only if **REFANT** reset since **CALIB** run.
> INP C_R	to review the inputs.
> GO C_R	to run the program when inputs set correctly.

PCAL will list the fitted values of the antenna polarization parameters and the source polarizations with estimates of the uncertainties. If these results do not appear reasonable (*e.g.*, large errors or large corrections or inconsistent solutions for the calibrator polarizations at neighboring frequencies), more editing and a rerun of **PCAL** may be necessary. **PCAL** puts the derived source polarizations in the **SU** table and the antenna feed values in the **AN** table. These values may be examined later with **PRTAN** and **PRTAB**.

Step 2:. Use **LISTR** to determine the apparent right minus left phase angle of the polarization calibrator source, *e.g.*, 3C286 or 3C138:

> TASK 'LISTR' C_R	
> SOURCE '3C286' , ' ' C_R	to view only the polarization angle calibrator.
> TIMERANG 0 C_R	to check all times.
> ANTENNAS *list of antennas* C_R	antennas to use; the list used for **CALIB**.
> UVRANGE *uvmin uvmax* C_R	to limit *uv*, if appropriate.
> BIF 1 C_R	to view the first ("AC") IF.
> FREQID *n* C_R	to view the current **FQ** value (*n*).
> DOCALIB TRUE C_R	to list with calibration applied.
> DOPOL TRUE C_R	to correct for feed polarization and Faraday rotation.
> GAINUSE 2 C_R	to use the latest **CL** table.
> OPTYPE 'MATX' C_R	to get matrix listing.
> STOKES 'POLC' C_R	to display cross-polarized data.
> DPARM 1,0 C_R	to display phases.
> INP C_R	to review the inputs.
> GO C_R	to run the program when inputs set correctly.

> BIF 1 ; EIF 2 C_R	to correct both IFs.
> FREQID *n* C_R	to correct only the current **FQ** value.
> GAINVER 3 C_R	to modify the **CL** table just produced by **TACOP**.
> OPCODE 'POLR' C_R	to do right minus left phase offset correction.
> STOKES 'L' C_R	correction applied to left circular polarization.
> CLCORPRM *cor*1, *cor*2 C_R	to set the phase correction for "AC" and "BD" IFs.
> INP C_R	to review the inputs.
> GO C_R	to run the program when inputs set correctly.

This will cause **CLCOR** to apply appropriate corrections to the **CL** and **AN** tables. If the **CL** table becomes hopelessly corrupted, delete it and return to Step 3. If the **AN** table is corrupted, then **PCAL** must be re-run. If more than one **CL** table needs to be corrected, use the OPCODE='POLR' option only once; other **CL** tables should be corrected using OPCODE='PHAS' and correcting 1 IF at a time. Corrections applied by **CLCOR** are cumulative; multiple runs with OPCODE='POLR' *will* result in invalid feed parameters in the **AN** table or incorrect right minus left phase offsets applied to the **CL** table. In particular, **PCAL** and **CLCOR** with **POLR** are a matched pair: there can be only *one* application of **POLR** per run of **PCAL**. If **POLR** is applied twice accidentally, then **PCAL** must be run again, followed by another **POLR**.

Step 5: Use **LISTR** to verify the polarization corrections:

> TASK 'LISTR' C_R	
> SOURCE '*cal*1' , '*cal*2' , ... C_R	to list the calibrators to be checked.
> TIMERANG 0 C_R	to display all times.
> ANTENNAS *list of antennas* C_R	to list the antennas to use.
> UVRANGE *uvmin uvmax* C_R	to set *uv* limits, if appropriate.
> BIF 1 C_R	to list the first ('AC") IF.
> FREQID *n* C_R	to use the current **FQ** value (*n*).
> DOCALIB TRUE C_R	to list with calibration applied.
> DOPOL TRUE C_R	to correct for feed polarization and Faraday rotation.
> GAINUSE 3 C_R	to use **CL** table written by **CLCOR**.
> OPTYPE 'MATX' C_R	to get matrix listing.
> STOKES 'POLC' C_R	to display cross-polarized data.
> DPARM 1,0 C_R	to display phases.
> INP C_R	to review the inputs.
> GO C_R	to run the program when inputs set correctly.
> BIF 2 C_R	to list the second ("BD") IF.
> WAIT ; GO C_R	to wait for the first job and then list other IF.

Note well: all of this calibration process must be done with only one **FQ** at a time. **PCAL** with FQID = 2 will over-write solutions done for any other **FQID**.

The phases produced should be consistent. Significant deviations of the phase may indicate that further editing is needed or that residual atmospheric phase errors are still present. If this display appears okay, then the polarization corrections may be applied in **SPLIT** (see below) by specifying DOPOL = 1 when applying the calibration to produce single-source files.

4.7. 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.

4.7.1. Reading the data

If your data are on a VLA archive tape then they should be read into *AIPS* using **FILLM**, as described in § 4.1.1. **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. 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.*

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. To do this, use the task **AVSPC**:

> TASK 'AVSPC' C_R	
> INDI *n* ; GETN *m* C_R	to specify line data set.
> OUTDI *i* ; OUTCL 'CH 0' C_R	to specify output "channel-0" data set disk and class.
> CHANSEL 10, 30, 1, 31, 55, 2 C_R	for example, to average every channel between 10 and 30 and every other channel between 31 and 55.
> GO C_R	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.

4.7.2. Editing the data

You should follow the steps outlined in § 4.B 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' C_R	
> INDI *n* ; GETN *m* C_R	to specify channel-0 data set.
> OUTDI *i* ; GETO *j* C_R	to specify the line data set.
> INEXT 'FG' C_R	to copy the **FG** table.
> INVER 1 C_R	to copy table 1.
> NCOUNT 1 C_R	to copy only one table.
> OUTVER 1 C_R	to copy it to output table 1
> INP C_R	to review the inputs.
> GO C_R	to run the program when inputs set correctly.

Specifying the "**ALL-CH**" setting in **TVFLG** and specifying BCHAN 1 ; ECHAN 0 C_R 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** to examine and, perhaps, to edit their data. This task is very similar to **TVFLG** described in §4.4.2, but **SPFLG** displays spectral channels 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' C_R	
> INDI *n* ; GETN *m* C_R	to specify channel-0 data set.
> OUTDI *i* ; GETO *j* C_R	to specify the line data set.
> INEXT 'FG' C_R	to copy the **FG** table.
> INVER 1 C_R	to copy table 1.
> OUTVER 1 C_R	to copy it to output table 1.
> BCOUNT 1 ; ECOUNT 0 C_R	to copy from the beginning to the end.
> OPTYPE 'COPY' C_R	to do a simple copy appending the input table to the output table.
> TIMER 0 C_R	to copy all times.
> INP C_R	to review the inputs.
> GO C_R	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 channels alone to a special "channel-n" data set.

4.7.3. Bandpass calibration

The task **BPASS** is designed to take visibility data from specified calibrator(s) to determine the antenna-based complex bandpass functions. It does this in a manner analogous to self-calibration in that the data are divided by a source model or the so-called "channel 0" before the antenna gains are determined as a function of frequency. These are written to a BandPass (**BP**) table. The bandpass calibration is the first operation that should be performed on the line data. So long as one uses the mode in which the data are divided by the so-called "channel 0," it is not necessary to calibrate the data before estimating the bandpasses.

> TASK 'BPASS' C_R	
> INDI *i* ; GETN *j* C_R	to specify line data set.
> CALSOUR '*cal1*' , '*cal2*' , ... C_R	to specify bandpass calibrators.

which can be sent to the printer (individually) by **LWPLA**. You might want to use a larger value of **NCOUNT** to reduce the number of executions of **LWPLA** and pieces of paper.

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, and most cpu-intensive, mode (DOBAND 3) is to interpolate in time between the antenna bandpasses and generate the correction from the interpolated data. This mode has been disabled within *AIPS* due to technical problems, and may, therefore, not be available.

4.7.4. Amplitude and phase calibration

The channel-0 data set should be calibrated as described above for continuum data (§§ 4.4 and 4.5). 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 2 ⏎	to copy table 2 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.

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 '*source*1' , ' ' ⏎	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 TRUE ⏎	to apply the antenna gain calibration.
> GAINUSE 2 ⏎	to use **CL** table 2.
> DOBAND TRUE ⏎	to apply the bandpass calibration.
> BPVER 1 ⏎	to use **BP** table 1.
> FREQID 1 ⏎	to use only one **FQ** value.
> APARM 0 ⏎	to do scalar averaging of amplitudes and self-scale the plots.
> SMOOTH 1 , 0 ⏎	to apply Hanning smoothing in the spectral domain.
> INP ⏎	to review the inputs.
> GO ⏎	to run the program when inputs set correctly.
> GO LWPLA ⏎	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.

4.8. Solar data calibration

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 on the VLA archive tape. 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 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. This is accomplished through two tasks. **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 calibartion parameters in the **CL** table.

4.8.1. Reading solar data from a VLA archive tape

To load a solar *uv*-data file to disk from a VLA archive tape follow the general instructions given above (§§ 4.1.1 and 4.7.1) with the following addition:

`> VLAMODE 'S ' ⏎` to indicate solar mode observing.

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 ⏎` 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 very large disk files which make all subsequent programs take a long time to run. 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 (§ 4.8.2) and then apply the system-temperature corrections (§ 4.8.3). Finally, apply the saved/copied **SN** table to the **CL** table 2 of each using **CLCAL**.

4.8.2. Assessment of the nominal sensitivities using SNPLT and LISTR

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. There are two tools available for this purpose: **SNPLT**, which plots the nominal sensitivities in graphical form and **LISTR** or **PRTAB**, which allow one to inspect the values directly.

> TASK 'SNPLT' ; INP C_R	to review the inputs needed.
> IND *m* ; GETN *n* C_R	to specify the input *uv* file.
> INEXT 'TY' C_R	to plot data from **CL** extension table.
> INVERS 0 C_R	to use the highest version number.
> SOURCES 'SUN' , ' ' C_R	to plot solar source only.
> TIMERANG 0 C_R	to select all times.
> ANTENNAS 5 11 12 18 C_R	to select only CAL-equipped antennas; this sample list for K, U, or C band.
> PIXRANGE 0 C_R	to self-scale each plot.
> NCOUNT 4 C_R	to do 4 plots on a page.
> XINC 1 C_R	to plot every **XINC**th point.
> OPTYPE 'TSYS' C_R	to plot nominal sensitivities.
> INP C_R	to review the inputs.
> GO C_R	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 name, **SOURCES**, and **ANTENNAS** adverbs are set as above):

> TASK 'LISTR' ; INP C_R	to review the inputs needed.
> OPTYPE 'GAIN' C_R	to list quantities in a calibration file.
> INEXT 'TY' C_R	to select the sensitivities.
> TIMER *d1 h1 m1 s1 d2 h2 m2 s2* C_R	to select by suspect time range.
> DOCRT -1 C_R	to route output to the printer.
> DPARM 10 0 C_R	to list nominal sensitivities.
> INP C_R	to review the inputs.
> GO C_R	to run the program when you're satisfied with inputs.

4.8.3. Applying the system-temperature correction with SOLCL

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. If you are using a version of *AIPS* older than **15JUL94**, you must copy **CL** table version 1 into **CL** table version 2 before running **SOLCL** (which does the copy for you in later releases). 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 ↵ to review the inputs.

> GO ↵ 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 § 4.8.1.

4.9. Completing the initial calibration

When you are satisfied with the initial calibration (pre self-calibration) of your data set, you should back up your full multi-source data set on magnetic tape. Then you can apply the calibration to the data for each program source, creating a separate single-source *uv* data set for each. These data sets are used with the imaging and self-calibration tasks to be described in the following chapters. For the impatient, there is one imaging task which reads the multi-source data set directly, applying any calibration,

4.9.1. Writing multi-source data to FITS tape with FITTP

The recommended way out of *AIPS* for multi-source *uv* data is to use **FITTP** to write a FITS-format tape. This will preserve the data and all associated calibration and editing tables in a machine-independent form. Consult § 3.9 about magnetic tapes in *AIPS*. That section tells you to mount your tape on the hardware device and then to do a software mount in **AIPS**. For example,

> INTAP *n* ↵ to specify which tape drive to use.

> DENSITY 6250 ↵ to set the density to 6250-bpi, if needed.

> MOUNT ↵ to mount the tape in software.

This step used to be optional for some operating systems. However, in recent versions of *AIPS*, it is required on all operating systems.

To write the data to tape:

> TASK 'FITTP' ↵

> IND *m* ; GETN *n* ↵ to specify the multi-source data set.

> DOEOT TRUE ↵ to write at the end of tape — if there are other data files on the tape you wish to preserve.

> OUTTA INTAP ↵ to write to tape just mounted.

> DOSTOKE FALSE ↵ to leave the data in input Stokes form.

> DOTABLE TRUE ↵ to write associated tables.

> DONEWTAB TRUE ↵ to use the latest format wrinkles — if the version of *AIPS* with which you will read the tape is recent enough. Read the help file for the latest details.

> FORMAT 3 ↵ to use IEEE floating format for data.

> BLOCKING 10 ↵ to use blocked FITS for tape efficiency.

> INP ↵ to review the inputs.

> GO ↵ to run the program when inputs set correctly.

Most people use 8mm Exabyte or 4mm DAT tapes today. These have very large capacities. However, if you must still use half-inch reel tapes, you will find that many data sets (particularly spectral line) may

be too large to fit on one 6250 bpi tape even with **BLOCKING = 10** and **FORMAT = 1** (16-bit integer). Since it is not possible to write multi-volume FITS tapes, it is recommended that you back up the single-source data sets formed after applying the calibration tables in **SPLIT** (see § 4.9.2). Alternatively, since all of the calibration information is contained in the extension tables, you may copy these to a dummy *uv* file with task **TASAV** and write this new file to tape with **FITTP**.

Be sure to run task **PRTTP** to make sure that the data were written succesfully on your tape *before* you delete your multi-source *uv* data set!

4.9.2. Creating single-source data files with SPLIT

When you are happy with the calibration 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' C_R	
> SOURCE '*sou1*' , '*sou2*' , ... C_R	to select sources, ' ' means all.
> TIMERANG 0 C_R	to keep all times.
> BIF 1 ; EIF 2 C_R	to keep both IFs
> FREQID 1 C_R	to set the one **FQ** value to use.
> DOCALIB TRUE C_R	to apply calibration.
> GAINUSE 0 C_R	to use the highest numbered **CL** 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 FALSE C_R	to write visibilities in uncompressed 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 files produced by this process should be completely calibrated and edited and ready to be imaged or further processed as described in later chapters.

4.9.3. Making images from multi-source data with HORUS

HORUS can be used to make images from multi-source data files. Since these data are not sorted into visibility-grid ("XY") order, large image sizes may require multiple passes through the data. 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 **HORUS** is:

> TASK 'HORUS' C_R	
> SOURCE '*sou1*' , '*sou2*', ... C_R	to choose sources to image, ' ' means all.
> TIMERANG 0 C_R	to use data from all times.
> OPTYPE 'SUM' C_R	to select continuum (combine channels) mode or **'LINE'** (or blank) for multi-channel mode.

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 **IBLED** is described at the end of the chapter. You may find it more useful than **TVFLG** (§ 4.4.2) for removing data at this stage in the processing. This chapter has been re-written for the **15JUL95** release of *AIPS* and significant portions of it do not apply to previous releases. In particular, task **IMAGR** was new and **SCMAP** became useful in that release. Tasks **MX**, **HORUS**, *et al.*, which are now obsolete, are no longer described.

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 ↵. The **15JUL95** versions of these lists are also given in Chapter 13 below. Basic data calibration is discussed in Chapter 4, editing is discussed in §§ 4.4 and 8.1, and imaging and self-calibration are also discussed in § 8.4 for spectral-line data and in § 9.10 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 tapes or disk files. 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

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. Type:

> TASK 'PRTTP' ; INP ↵	to list the required inputs on the terminal.
> INTAPE *m* ↵	to specify the tape drive number (*m*).
> NFILES 0 ↵	to print information for all the files.
> DOCRT -1 ↵	to print output on your system printer.
> PRTLEV 0 ↵	to select the level of reporting.
> GO ↵	to run the program.

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 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. 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**.

Once **UVSRT** has finished, check that a *uv* data base 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. Almost all *AIPS* installations, including the NRAO systems, have "private" catalog files, in which your *uv* files will have contiguous catalog numbers starting from 1 when you first write *uv* data to disk. See also § 3.3.3.

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**, 355 (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 Cotton and 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**; and the fourth 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*. § 5.3.6 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.

images, smaller values are better for smaller images of compact objects, and rather larger values may be good for extended objects. 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.

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 **TELL**s.) 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 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 parameter that must be set to restart a Clean is **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 a new file of residual visibilities. An image can be re-convolved by setting NITER = the sum of the **BCOMP**s 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. **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 **FITTP**.

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**, and **IN2DISK** parameters, but a new one is created each time you run **IMAGR**; *i.e.*, **IN2SEQ** must be 0. This file is useful if you suspect that there are bad samples in your data. Use **LISTR** (§ 4.4.1) **UVFND** (§ 6.2.1), **PRTUV** (§ 6.2.1), **UVPLT** (§ 6.3.1) or even **TVFLG** (§ 4.4.2) 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. Be sure to delete old ones with **ZAP**.

5.3.2. Multiple fields in IMAGR

IMAGR can also deconvolve components from up to 16 fields of view simultaneously, taking correct account of the w term at each field center. 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 sources 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.

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 $^{C}_{R}$	to make images of n fields.
> IMSIZE i,j $^{C}_{R}$	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.
> FLDSIZ $i_1, j_1, i_2, j_2, i_3, j_3, \ldots$ $^{C}_{R}$	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.)
> RASHIFT $x_1, x_2, x_3, \ldots$ $^{C}_{R}$	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 $y_1, y_2, y_3, \ldots$ $^{C}_{R}$	to specify the y shift of each field center from the tangent point; $y_n > 0$ shifts the map center to the North (up).

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 roughly **RASHIFT** divided by $\cos(\delta)$.

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. 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. 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. Even then, there is one other "gotcha." In the restore step, Clean only restores components to the fields in which they were found. 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.

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, bottommost 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* : *filename*' ↵ — to specify the name of a text file listing the Cleaning windows. Blank means no file.

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 500 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* character is not a number is taken as a comment, *i.e.*, the field number must be left-justified in the line (leading zeros, not blanks, are allowed). **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.

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.

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 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^{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.

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 left column options are:

OFFZOOM	to turn off any zoom magnification
OFFTRAN	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**
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**
REBOX	to reset the current Clean boxes and create more as in **AIPS**

The explain file includes more detailed information on these options or you can get on-line help with TV button **D**.

The right hand column offers the options

CONTINUE CLEAN	to resume Cleaning now rather than wait for the time out period.
STOP CLEANING	to stop the Clean at this point, restore the components to the residual images, write them on disk, and exit.
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**.

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.

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.

We encourage use of DOTV TRUE $\mathcal{C}_R$ when you are Cleaning an image, especially for the first time. 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. 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.

5.3.4.1. 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 **CPARM(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. The primary beam is assumed to be that of a uniformly illuminated disk of diameter **CPARM(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.

5.3.4.2. 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 **CPARM(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.

5.3.4.3. 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 **CPARM(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.

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 from the data, producing a residual visibility data set. 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 § 9.1, 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.

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.

the task. 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.)

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 16 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 $\subset_R$.

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 $\subset_R$, EXPLAIN APVC $\subset_R$ 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 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!

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!

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, do not set **NCOMP** in **CALIB** so high that any 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 (§ 5.3.5). 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** (§ 5.3.5) 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 **SOLINT**s 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 will 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.*

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 § 4.4 before investing large amounts of time in editing even at this stage.

There are facilities in **CALIB**, **CLIP**, and **CORER** 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 **UVFND** is also recommended for printing out suspicious portions of the data base; 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 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 **APARM(5) = 1** in **CLIP**.

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. The task **UVSUB** will subtract the

IBLED has three main steps:

1. The data are selected and calibrated using all of the usual calibration adverbs and options. Then they are averaged over spectral channels **BCHAN** through **ECHAN**, over IFs **BIF** through **EIF** (if **DPARM(3)**> 0), and over **DPARM(5)** seconds of time. The resultant visibilities are written to a "master file" which is cataloged temporarily or (semi) permanently on disk **IN2DISK** with class **IBLEDR**. Each visibility record in this file consists of a weight, vector-averaged amplitude, vector-averaged phase, and "decorrelation" index (ratio of vector-averaged to scalar-averaged amplitude). This master file can be used in one or, if **DOCAT**> 0, more than one session of step 2 below to prepare flag commands which are stored in an **FC** table attached to the master file.
2. You then interact with your data, in one or more sessions, arranging the TV display and then selecting which data are to be flagged. The data are plotted with time along the X axis plotting one point per TV pixel and skipping 20 pixels between scans. Therefore, the time axis is quite non-linear. Tick marks are placed at integer hour points along the X axis and the end points labeled to provide you with some idea of the times. The selected parameter (amplitude, phase, decorrelation coefficient) is plotted linearly on the y axis. All interactive editing operations provide detailed displays of the currently selected time and visibility. The main editing area may not be large enough to contain all the data. In this case, a smaller "frame" of data is selected and you may alter the selected frame to gain access to all of your data. A small plot at the top of the screen shows all of the data, potentially with a fairly crowded X (time) axis.
3. When you have done enough, you instruct **IBLED** to apply the flag commands to the data. A flag table is written for multi-source files and may be written for single-source files. If there is not already a flag table on the single-source file and you set **FLAGVER** = **0**, the *uv* data themselves are flagged. This must be done when exiting if you have not cataloged the master file (*i.e.*, **DOCAT**$\leq$ 0) or when you are fully satisfied with your flag commands. When the flags are applied to the input data set, the master file is deleted.

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 first column contains options to alter the display of the data and the choice of which data are flagged ultimately. The second column has 7 interactive modes for selecting data to be flagged, 2 operations on the **FC** table itself, and 10 operations which cause the display to be changed.

When the menu is displayed, one or more lines of status information are displayed at the bottom of the screen. The line that is always present shows the type of data displayed and which Stokes and whether one or all sources, IFs, and channels will be flagged by the next flag commands. If all baselines to one or both of the displayed antennas will be flagged and/or if the points are currently plotted with error bars, a second line reflecting this fact will be displayed. If the Stokes type, the first or second IF, or the error bar type will change on the next **LOAD**, then another line is displayed saying "**NEXT LOAD SHOWS**" these new things. Finally, if some data are omitted from the plot due to user-selected plot scaling, then another line appears warning of this fact. The plot of the data is labeled along the vertical axis by normal labeled tick marks and the units, while the horizontal axis has ticks at integer hours and time strings at both ends of the plotted data. The line at the top of the plot identifies the data currently displayed.

The left-hand menu section begins with 3 operations to alter the TV display zoom and TV transfer functions. The zoom is useful for greater accuracy in selecting data to be flagged, but the enhancements

have little use here. Then there are 3 options to set the range of amplitude, phase, and decorrelations which are actually plotted. These default to the full range in the current data, and can be set back to default by entering `0 0`. The next two options ask you to enter the desired IF number and Stokes type for the next display, but do not actually change the display. Use any of the redisplay options in the second menu (`LOAD` is simplest) to update the display. The next option allows you to specify a second IF number. When that is not 0 and not the same as the first IF, `IBLED` plots the amplitude ratio or phase difference of the first IF and the second IF. The next 5 options switch the binary states of several flags: the first controls the plotting of the full data set in the small plot at the top, the second controls the vertical bars plotted about the data, switching between error estimates and a small constant, and the rest control whether one or all spectral channels, IFs, and sources are flagged by the following flag commands. The next option allows you to set the Stokes which will be flagged by the following flag commands. And the final 2 options allow you to switch between flagging only the displayed baseline and flagging all antennas to one or both of those displayed. The last two menu items actually reflect the station names for the two antennas and change as you change baselines.

This menu looks like:

OFF ZOOM	to turn off any zoom magnification.
OFF ENHANCEMENT	to turn off black & white and pseudo-color enhancements.
TVFIDDLE	to do interactive zoom and enhancement.
ENTER AMP RANGE	to type in the intensity range to be used for plotting amplitudes (Jy or ratio).
ENTER PHS RANGE	to type in the phase range to be used for plotting phases (degrees).
ENTER DCR RANGE	to type in the decorrelation range to be used to plot decorrelation (0-1 or ratio).
ENTER IF	to enter on the terminal the desired IF to be displayed next.
ENTER STOKES	to enter on the terminal the Stokes type to be displayed next.
ENTER 2ND IF	to enter on the terminal the IF to divide into the first IF to display ratios of data. A zero turns off this option.
SHOW TOP PLOT	to toggle on/off the display of the complete data for the current baseline along the top of the screen.
PLOT ERROR BARS	to toggle between plotting error bars or a small constant bar — works only with fully calibrated data weights.
SWTCH 1-CH FLAG	to toggle between flagging all spectral channels and only `BCHAN` through `ECHAN`.
SWTCH 1-IF FLAG	to toggle between flagging all IFs or only the one shown (or `BIF` through `EIF` if averaged).
SWTCH 1-SO FLAG	to toggle between flagging all sources in any time range or flagging only those displayed.
SET STOKES FLAG	to enter on the terminal the 4-character string which will control which correlators (polarizations) are flagged. Note: this parameter applies only to flag commands prepared after it is set. It should be changed whenever a different Stokes is displayed. See also the explain file.
FLAG *station*1	to toggle between flagging all data to this antenna and flagging only data with *station*2.

FLAG *station*2 — to toggle between flagging all data to this antenna and flagging only data with *station*1.

The all-channel flag remains true if the full range of channels in the input data set is averaged in the data being flagged. The all-IF flag remains true if the full range of IFs in the input data set is averaged in the data being flagged.

The right-hand menu section begins with 7 flagging operations: to flag one time at a time, to flag a range of times, to flag an area in the intensity-time plane, to flag all points above some intensity or below some intensity, to flag all points more than x times the rms from the mean, and to flag all points above or below a user-drawn curve in the intensity-time plane. During these operations the current position selected by the cursor is displayed in the upper left hand corner of the TV with source name, time, flux, and (optionally) error. Lines or boxes are also drawn on the TV to indicate the area being selected for flagging. During interactive flagging, button **A** switches between corners or end times except for **FLAG TIME** where it does the flagging and **FLAG INTERACTIV** where it marks a point. Button **B** does the flagging and loops for more except for **FLAG INTERACTIV** where it marks another point on the curve. 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. When a flagging command is generated, any flagged data currently displayed are erased (only from the lower plot) and some number of records are written into the **FC** table. Note that one can flag only those data in the current frame using the intensity type in the current display (plus other IFs, Stokes, etc. as chosen by the flagging control commands). You must visit other frames in order to flag them.

The next two operations in the right-hand menu allow you to list the flag commands already in the **FC** table and to undo any of them. The **UNDO FLAGS** operation prompts you for a list of flags to be undone by number (get these from **LIST FLAGS**) with a zero ending the list. When a flag is undone, all cells in the master file which were first flagged by that command are restored to use, except for those also flagged by some later command. This is done automatically by **IBLED** without the special commands required by **TVFLG**. After an **UNDO FLAGS** operation, the TV is reloaded, potentially with new plot scales.

The next 10 operations in the right-hand menu also cause the display to be updated immediately if needed. The first three select the type of data to be displayed and used in flagging; the next four change the current frame (portion of the current baseline) being displayed and edited; the next two change the current baseline, and the last forces an update of the screen including the top plot in case some of the changes (*e.g.*, flagged points, new IF or Stokes selection, etc.) have not been fully reflected in the displays. The **SELECT FRAME** uses the top plot to select the desired frame interactively and can work only when the top plot is displayed. The last option in this menu selection is to **EXIT**, optionally applying any flagging commands to the input data set.

The right-hand menu looks like:

FLAG TIME — to flag single visibility points.

FLAG TIME RANGE — to flag all data in a range of times

FLAG AREA — to flag a rectangular area in the flux-time plane.

FLAG ABOVE — to set a lower limit and flag all data above it over a selected range of times.

FLAG BELOW — to set an upper limit and flag all data below it over a selected range of times.

FLAG ABOUT MEAN — to set a time range within which the mean and rms are determined and then flag all data outside the range MEAN $\pm x\times$ RMS, where the user is asked to specify x.

FLAG INTERACTIV — to define a piecewise-linear curve and then flag all data above or below it.

LIST FLAGS	to list a selected range of flag commands.
UNDO FLAGS	to remove flags by number from the **FC** table and undo that flagging in the data.
SHOW AMPLITUDE	to reload the TV, displaying amplitudes.
SHOW PHASE	to reload the TV, displaying phases.
SHOW DECORRELAT	to reload the TV, displaying decorrelation index.
SELECT FRAME	to select the visibility set to be displayed interactively from the top plot.
FIRST FRAME	to display the first set of visibilities.
NEXT FRAME	to display the next set of visibilities.
PREVIOUS FRAME	to display the previous set of visibilities.
SELECT BASELINE	to display a new baseline by entering the desired antenna pair on the terminal.
NEXT BASELINE	to display the first set of visibilities from the next baseline.
LOAD	to reload the TV with the current flags and parameters, checking the scale.
EXIT	to resume **AIPS** and, optionally, enter the flags in the data.

Before the flags are entered in the data, **IBLED** asks you whether or not you actually wish to do this. You must respond yes or no. If the master file is not cataloged (**DOCAT** ≤ 0), your flags will be lost if you do not say yes. If the master file is cataloged and you intend to do more flagging on this particular set of data, then you should answer no. Having done so, you can resume later without having to compute a new master file and will have the option to undo previous **IBLED** flag commands. Once the flags are applied to the input data set, they are much harder to undo and the master file is destroyed since it no longer accurately represents the input data.

5.6. Additional recipes

5.6.1. 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.

6. IMPROVING IMAGES — DECONVOLUTION, SELF-CAL, EDITING

This Section deals with the major *AIPS* programs that can improve the quality of dirty images which are not noise-limited but which suffer from dynamic range problems. See Chapters 8—10 and 12, 13 and 16 in *Synthesis Imaging in Radio Astronomy**. for general discussion and guidance about image defects and the strategies for reducing them. Chapters 10 and 16 are particularly useful in this respect.

6.1. Deconvolving images

The most widely used deconvolution method is CLEAN. Images made with **HORUS** or **UVMAP** can be CLEANed using the task **APCLN** described in § 6.1.1. **MX** combines imaging and CLEANing functions in one task. Its CLEAN option uses many of the same inputs as **APCLN**, plus extras that we describe in detail in § 6.1.2. Sections 6.1.3 through 6.1.5 give tips, applicable to both **APCLN** and **MX**, on monitoring the progress and success of a CLEAN and on restarting CLEANs.

Images may also be deconvolved by other methods in *AIPS*. § 6.1.6 mentions several of these and describes the most popular alternative, a Maximum Entropy method embodied in the task **VTESS**.

6.1.1. APCLN

APCLN implements a CLEAN deconvolution of the type devised for array processors by Barry Clark (*Astron. & Astrophys.* **89**, 355 (1980)). (Your computer does not need not to have an array processor or other special vector hardware to run **APCLN**, however.) CLEAN components 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, multiplied by the transform of the beam, transformed back to the image plane, and then subtracted from the dirty image. The rule for deciding when a major iteration should end in order to achieve a desired accuracy is complicated (see the Clark paper). As well as letting you control the usual options associated with CLEANing such as loop gain, boxes, etc., **APCLN** lets you vary the major iteration rule somewhat to suit the requirements of your image. Type DOCRT TRUE; EXPLAIN APCLN ↵ to print out advice on CLEANing.

To run **APCLN**, enter:

> TASK 'APCLN' ; INP ↵	to see what you have to specify.
> INDI *n1* ; GETN *ctn1* ↵	to select the dirty image, where *n1* is the disk on whichit is stored and *ctn1* is its catalog number there.
> IN2D *n2* ; GET2N *ctn2* ↵	to select the dirty beam, where *n2* and *ctn2* are its disk and catalog numbers.

Examples of other inputs to **APCLN** are:

> OUTN '3C138 A C' ↵	to specify the CLEAN image name — default: same as **INNAME**.
> 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 (§ 6.1.4).

* *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)

> OUTC 'ICLN1' ↵	to specify the output file class. It defaults to **'ICLN'** for **IN-CLASS IMAP**, **'QCLN'** for **QMAP**, **'UCLN'** for **UMAP**, and **'VCLN'** for **VMAP**.
> GAIN 0.2 ↵	to set the loop gain parameter, defaults to 0.1. Values in the range 0.1 to 0.25 may be suitable for complicated sources, higher values are okay for simple, point-like sources.
> FLUX 0.003 ↵	to stop CLEANing when peak of residual image falls to 3 mJy.
> NITER 500 ↵	to stop CLEANing when 500 components have been subtracted (default 100).
> BMAJ 1.25 ↵	to set the major-axis FWHM of restoring beam to 1.25 arcsec. **BMAJ = 0** specifies that the beam is to be fitted. Use **BMAJ < 0** if you want the *residual* image, not the CLEAN one, to be stored in the output file.
> BMIN 1.10 ↵	to set the minor-axis half-width of restoring beam to 1.10 arcsec.
> BPA 52.8 ↵	to set the position angle of beam axis to 52°48′ measured counter-clockwise from North (*i.e.*, East from North).
> INVERS 1 ↵	to specify which version of the CLEAN components file(s) is desired. Default is 1.
> NBOXES 1 ↵	to set the number of boxes in which to search for CLEAN components. Must be ≤ 10, default is 1.
> BOX 156, 156, 356, 356 ↵	CLEANing box is 201 × 201 centered on pixel (256, 256). Default 0, 0, 0, 0 specifies inner quarter of image area. Fill **BOX** with more windows if **NBOXES** > 1. Note that **APCLN** cannot CLEAN properly over more than one-quarter of the image area.

You can use the TV cursor to set the CLEANing boxes if the TV display shows either the dirty image or a previous version of the CLEAN image. Type:

> NBOXES *n* ↵	to set the number of boxes, where $n \leq 10$.
> TVBOX ↵	to begin an interactive, graphical setting of the *n* boxes.

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 box array for **APCLN**. Note that the terminal will display some additional instructions. These may be followed to reset any of the previously set corners should you need to do so.

To continue the example parameters:

> DOTV TRUE ↵	to show residual images on TV in interactive **AIPS** and to provide an opportunity to terminate the CLEAN after each major cycle using button D. The residual image is displayed on the TV after each major cycle and the CLEAN image is displayed at the end, if **DOTV** is set to **TRUE** (+1.0). Default = **FALSE** (−1.0) for no display. (See also § 6.1.3.)
> INP ↵	to review what you have specified.

Note that specifying **INNAME** and (usually) **NITER** may be enough to run the program if your dirty image and beam have the same image name (differing only by their classnames, *e.g.*, **IMAP** and **IBEAM**) and if you are content to use the system defaults for the CLEANing parameters. These defaults are reasonably chosen for CLEANing a complicated image.

The **FACTOR** parameter in **APCLN** 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.

When you're satisfied with the inputs listed by **INP**, then:

> GO ↵ to start the CLEAN.

6.1.2. MX

In its CLEANing mode, **MX** finds the individual components in "minor" iterations by the same methods as **APCLN**. However, **MX** then subtracts CLEAN components from the ungridded *uv* data using either a direct Fourier Transform (DFT) or a hybrid DFT/gridded Fast Fourier Transform (gridded-FFT) to compute the model visibility from the CLEAN components. The DFT is more accurate, but is often slower than the gridded-FFT (except in some multiple-field cases — EXPLAIN MX ↵ will print out some text containing, among many other things, details of timing as a function of data set size and number of CLEAN components). You can let **MX** choose which method to use as its CLEANing progresses by using the default (null) value of its input parameter **CMETHOD**; we strongly recommend use of this default unless you are sure which method to specify for your case (again, use EXPLAIN MX ↵ to print out some detailed advice on this).

CLEANing by subtracting component visibilities from the ungridded data does not alias sidelobes into the field of view. **MX** can therefore CLEAN more accurately than **APCLN**, and over a much larger portion of the dirty image. This feature of **MX** is especially valuable for "snapshot" observations in which the dirty beam has high sidelobe levels.

MX can also deconvolve components from up to 16 fields of view simultaneously (provided that they were imaged simultaneously using **MX**'s imaging step), taking correct account of the *w* term at each field center. 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. To take advantage of this property of **MX** to CLEAN high-resolution images using prior knowledge about the field (or information from a low-resolution image made from your own data) you will need the to use the multi-field option at **MX**'s imaging step. An illustrative set of input parameters might be:

> NFIELD 3 ↵ to specify that you want to image 3 fields.

> IMSIZE 128 to specify that the minimum image size for any of the fields should be 128 by 128.

> RASHIFT = -70, 0, 250 ↵

> DECSHIFT = 120, 10, -400 ↵

to specify that the first field should be centered on $(-70'', 120'')$, the second on $(0'', 10'')$ and the third on $(250'', -400'')$ relative to the original tangent point. Note that the true effect of using **RASHIFT** is to shift the image by approximately **RASHIFT** / $cos(\delta)$ where δ is the declination of the original tangent point. You might then specify the CLEANing areas within each field by the parameter:

> FLDSIZ 32, 32, 246, 246, 20, 20 ↵

These inputs would produce three images. The first would be 128 by 128 cells centered on $(-70'', 120'')$; it would be searched for CLEAN components over its center 32 by 32 cells. The second would be 256 by 256 cells centered on $(0'', 10'')$; it would be searched for CLEAN components over its center 246 by 246 cells, leaving a five-cell "guard band" around the CLEANing area to avoid interpreting the region most likely to be contaminated by effects of aliasing and the gridding convolution. The third would be 128 by 128 cells centered on $(250'', -400'')$. It would be searched for CLEAN components over its center 20 by 20 cells. The beam would be 256 by 256 cells centered on $(0, 0)$.

Most of **MX**'s CLEANing inputs are like those described under **APCLN** in § 6.1.1. The following are used exactly as in **APCLN**: **OUTN**, **OUTS**, **OUTCL**, **GAIN**, **FLUX**, **NITER**, **BMAJ**, **BMIN**, **BPA**, and **FACTOR**. **NBOXES** and **BOX** in **MX** work as in **APCLN**, but apply only to the first field defined by the imaging inputs. If **NBOXES** = 0, the CLEAN search area of field number 1 is determined from the first **FLDSIZ** window. If **NBOXES**> 0, **FLDSIZ** is used only to check the size of the image required for field 1. DOTV = -1 in **MX** specifies no residual display on TV channel 1 (as in **APCLN**) but DOTV = n with $1 \leq n \leq 16$ in **MX** specifies TV display of the residuals from field number n.

The following **MX** inputs differ substantively from those of **APCLN**:

> INDI *n* ↵ — to specify the number of the disk on which the original *uv* data set used to make the images resides.

> INNAM '*filename*' ↵ — to specify the filename of the original *uv* data set.

> INCL '*classname*' ↵ — to specify the classname of the original *uv* data set.

> INSEQ *seqn* ↵ — to specify the sequence number of the original *uv* data set.

> IN2DI *n* ↵ — to specify the number of the disk on which resides a *uv* work file in which **MX** stores the current *uv* data with the current list of CLEAN components subtracted. The file is created at the imaging step with default name the same as the input *uv* data set and default class **.UVWORK**. It is modified as **MX**'s CLEAN progresses.

> IN2N '*filename*' ↵ — to specify the filename of the *uv* work file; default is **INNAM**.

> IN2CL '*classname*' ↵ — to specify the classname of the *uv* work file; default is **UVWORK**.

> IN2SEQ *sqn* ↵ — to specify its sequence number; default is highest.

The above inputs may of course be set using:

> INDI *n1* ; GETN *ctn1* ↵

> IN2DI *n2* ; GET2N *ctn2* ↵

as usual, where (*n1*, *ctn1*) and (*n2*, *ctn2*) select the disk numbers and catalog slot numbers of the appropriate files. Alternatively, if you have previously run **MX** with the same parameters, TASK 'MX' ; TGET ↵ will restore them.

Note that the *uv* workfile will exist, and you should specify its parameters, even if you have only made a dirty image or images with **MX** previously. If you do not specify the workfile parameters, **MX** will repeat all steps of making the dirty image, which wastes time.

Note also that **MX** will remake the beam file whenever it is restarted, as the beam it uses for CLEANing need not have the same size as the image. (During CLEANing, it uses the smallest beam it thinks it can get away with). We therefore recommend that you delete or rename the beam file from any previous run of **MX** whenever you restart analysis of the same data set with **MX**, to prevent confusion later. (Renaming the full-size beam made when **MX** is used without CLEANing may be useful if you subsequently want to use it for another type of deconvolution, such as **VTESS**, which needs such a beam file).

To specify the default component subtraction method, type:

> CMETHOD ' ' ↵ — to allow **MX** to use DFT or gridded-FFT component subtraction at each major cycle, depending on which is faster. We strongly recommend use of this default unless you are very sure that you wish to force DFT or gridded-FFT subtraction at all iterations. 'DFT' forces DFT subtraction; 'GRID' forces gridded-FFT subtraction.

Before running **MX**, use:

the previous CLEAN image since the CLEAN component list is associated with this image file (as its **CC** extension). When restarting a CLEAN with **MX**, you must also spcify **IN2N**, **IN2CL** and **IN2SEQ** explicitly to identify the *uv* workfile. An image can be re-convolved by setting NITER = BITER (in **APCLN** only) and specifying the desired (new) CLEAN beam.

Both **APCLN** and **MX** write over the CLEAN image file(s) as they proceed 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 **FITTP**.

6.1.5. Manipulating CLEAN components

To print the list of CLEAN components associated with a CLEAN image, use:

> INDI *n* ; GETN *ctn* ↵	to select the CLEAN image, where *n* and *ctn* select its disk and catalog numbers.
> BITER *n1* ; NITER *n2* ; XINC *n3* ↵	to list CLEAN components from *n1* to (*n1* + *n2* - 1) with increment *n3*.
> DOCRT *m* ↵	to route the list to your terminal of width *m* characters.
> DOCRT FALSE ↵	to route the list to the line printer.
> GO PRTCC ↵	to execute the task.

The list of CLEAN components associated with a CLEAN image can be compressed by:

> 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 CCMRG ↵	to execute the task.

The algorithm used by **APCLN** and **MX** 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 which can reduce greatly the size of the list and, hence, the time required for model computations in tasks such as **CALIB** (§ 6.2) and **UVSUB**.

You can plot the list of CLEAN components associated with a CLEAN image in various ways with **TAPLT**. 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 QMSPL ↵	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. There should seldom be a need to edit CLEAN component files. 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.

6.1.6. Alternative deconvolution methods

The subject of image deconvolution has been widely studied and many methods have been proposed for tackling it. CLEAN is renowned for often yielding images that contain many artificial beam-sized lumps,

determine the appropriate *uv*-limits for the gain solution by referring to the hard copy of the visibility function you made at step 2. **CCMRG** may be used to reduce the number of components in the model, which will improve the speed of **CALIB**. However, unless you were going to include all components, this operation alters the model which is used to compute the gains.

6. Use **UVSRT** to sort the data from "XY" (pre-imaging) order back to "TB" order (unless you preserved the TB sort for self-calibration purposes or have made your image with **HORUS** from either a single-source or multi-source "TB" sorted database).

7. Plan your **CALIB** inputs; see the inputs list and the section on "Choosing **CALIB** inputs" that follows below (§ 6.2.3).

8. Use **CALIB** to calculate the gain corrections, to produce a new TB-sorted data set, and to catalog the gain corrections as an extension to the **input** *uv* data file.

9. Use **SNPLT** on the input data file, followed by **TKPL**, to review the gain corrections before proceeding further. (You may want to take hard copy of this output for future reference also — use the TEK hard-copy switch, or run **PRTPL** or **QMSPL** on the plot file extension of the old *uv* data file that was written by **SNPLT**.) Also use (**GNPLT** + **TKPL**) to plot the extrema of the gains.

10. 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.

11. If the corrections were believable, run **UVSRT** to produce a new version of the *uv* data in XY order for imaging if you are using (**UVMAP** + **APCLN**). You should consider deleting one or more of the *uv* data sets from the previous iteration at this point to save disk space.

12. 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 5 through 11 unless you feel you can make an informed change to the **CALIB** input parameters at step 7. Task **UVDIF** may help you to decide whether there have been significant changes to your data due to the previous iteration of **CALIB**.

6.2.2. Inputs to 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". (**CALIB** no longer includes global fringe-fitting *i.e.*, generating solutions for antenna-based solutions for phase, phase rate and group delay — task **FRING** now handles this operation.) 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 with listed with **LISTR**.

The following input parameters are used by **CALIB**:

> INDI *n1* ; GETN *ctn1* ↵	to select the 'TB' sorted *uv* data base.
> IN2D *n2* ; GET2N *ctn2* ↵	to select the CLEAN image to use.
> NMAPS q ↵	select the number of map CC files to use for the model.
> NCOMP = $n_1, n_2, \ldots$; INVERS m ↵	to cut off the model at the n_i^{th} CLEAN component in the m_i^{th} CC file(s) associated with this image.
> SMODEL S, x, y, m ↵	don't use a clean component model — use a source model (type m) of S Jy located at x, y arcsec with respect to the pointing centre. Generally, m = 0 *i.e.*, a point model.
> SUBARRAY = s ↵	select the appropriate subarray — **SUBARRAY = 0** implies all subarrays.
> UVRANGE = x_1, x_2 ↵	minimum and maximum baseline (in kilowavelengths) to be given full weight.
> WTUV = x_3 ↵	weight for baselines outside the **UVRANGE(1)** → **UVRANGE(2)** range. (**WTUV = 0** is interpreted as zero weight.)
> REFANT = x_4 ↵	reference antenna (choose a known good one for best results).
> SOLMODE 'A&P' ↵	to solve for amplitude and phase corrections.
> SOLMODE 'P' ↵	to solve for phase weighted by amplitude.
> SOLMODE 'P!A' ↵	to solve for phase ignoring amplitude.
> SOLTYP ' ' ↵	normal (non-linear) least squares solution.
> SOLTYP 'L1' ↵	L1 solution - useful for databases with lots of outlying points *e.g.*, MERLIN, VLBI databases. This attempts to approximate the antenna gains by minimising the sum of the moduli of the residuals. Note that the 'L1' mode will cause the program to run more slowly than for the normal least squares.
> SOLTYP 'GCON' ↵	least squares with a gain constraint — this requires the use of **GAINERR** and **SOLCON**.
> GAINERR $g_1, g_2, g_3, \ldots$	estimates of the standard deviations of the modulus of the gains for each antenna (in order). Accurate values of these are necessary.
> ANTWT $w_1, w_2, w_3, \ldots$	additional antenna weights (in order) to be used in generating the solutions. The defaults (= 0) imply weights of unity.
> APARM(1) = x_5	minimum number of antennas required for a solution — the default (= 0) implies 6.
> APARM(2) = x_6	if > 0 then the data has already been divided by the model.
> APARM(3) = 0 ↵	to solve for RR and LL separately.
> APARM(3) = 1 ↵	to average RR and LL correlators before solving.
> APARM(4) = x_7	if > 0 then average all frequencies in each IF before determining a solution.
> APARM(5) = x_8	if > 0 make a combined solution for both IFs — not recommended for polarisation observations.
> APARM(6) = x_9	set print level flag — set this to 0 (none) or 1 (some information). These give manageable levels of useful information. **APARM(6) = 2** is *not* recommended.

> APARM(7) = x_{10}	set the minimum acceptable S/N ratio. The default uses a S/N ratio of 5.
> SOLINT = x_{11} C_R	to set the length of the solution interval (in minutes).
> APARM(10) = 0 C_R	to scale the gain corrections, when **APARM(7) = 1**, by the mean modulus of all gains to keep the flux density scale from drifting.
> APARM(10) = 1 C_R	to request no gain normalization.
> CPARM(1) = 0 C_R	don't average over *all* frequencies in generating a solution.
> CPARM(1) = 1 C_R	average over *all* frequencies.
> CPARM(2) = 1 C_R	constrain the *modulus of mean gain* to be unity.
> CPARM(3) = y_1 C_R	print the values of amplitude errors which are greater than y_1 %.
> CPARM(4) = y_2 C_R	print the values of closure phase errors which are greater than y_2 degrees.
> CPARM(5) = 1 C_R	form scalar average of amplitudes before forming solution.

Other parameters are defaulted sensibly — type EXPLAIN CALIB C_R 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!

6.2.3. Choosing CALIB inputs

In many cases, only a few **CALIB** input parameters 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, do not set **NCOMP** in **CALIB** so high that any 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. 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)**.

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)** kilowavelengths (read this off a plot file output from **UVPLT**).

SOLINT (or **APARM(9)** in **ASCAL**) 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

components (using the same criteria as **UVMAP**) before checking that the amplitudes 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 **APARM(5) = 1** in **CLIP**.

TVFLG and **IBLED** are TV-based, interactive editors. **TVFLG** is most suitable for datasets with large numbers of baselines *e.g.*,the VLA but it can be used usefully for VLBI data — for example, *global* 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 § 4 of the *COOKBOOK — Calibrating VLA data in AIPS*. **IBLED** has a different philosophy; it plots one baseline at a time. This is obviously more useful for small arrays — *e.g.*, VLBI, MERLIN, the Australia Telescope. This program is described below in § 6.5.

CLIP is a very useful task 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. The task **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* dataset 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 **CLIP** away evidence for real extended structure near the center of your *uv* plane! Before re-imaging, you must of course run **UVSUB** again after doing the flagging or clipping, to add the transform of the CLEAN components back into the data (using the input FACTOR = −1.0).

Note that **MX**'s workfile is also a *uv* dataset from which the current CLEAN component model has been subtracted. It may also be used with **UVPLT** to help you to diagnose problems.

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 readback verbs like **CURVALUE** and **IMPOS** 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 **APCLN** and **MX** to put out a residual image by setting **BMAJ** <0.)

6.4.1. 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° 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.

6.5.2 Practical notes

The display of the global window can be suppressed to speed up the display. Typically, you should invoke the global window the first time that you look at a particular baseline. If you require to edit the data extensively throughout the whole timerange, you may wish to turn off the global window by toggling the **SHOW ALL DATA** switch and select each frame in turn with **NEXT FRAME**.

The time axis of the global window is not completely regular in general. Gaps in the data, for example, scan boundaries, will be displayed as gaps of 20 pixels in this window. Note, too, that this will mean that data observed in two sessions several hours or days apart will appear adjacent to each other in the global window. This may give rise to *apparent* discontinuties in the visibility function. This may seem rather awkward but, since the global display is used for frame selection, it is important to display the data selected as usefully as possible. If your data were obtained in several sessions with large gaps of time in between, it may be better to select sub-sections of your data with the **TIMERANGE** adverb and to edit these separately.

Currently, the aspect ratio of the data frame display is different from that of the global display. Occasionally, this can create some confusion with the selected frame looking somewhat different from it's depiction in the global window.

On some TV displays, many of the commands in the menus are somewhat abbreviated. This is a little unfortunate but is forced upon the display by the limited size of TV screens generally.

This is a new program and is under development currently. If you wish enhancements to the current program or have suggestions for it's improvement, contact the *AIPS* group by way of the GRIPES system or by **aipsmail** and let us know.

% setenv MYLOGICAL *myarea* ↵

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.

Then, once inside *AIPS* , tell **IMLOD**:

> INFILE 'MYLOGICAL:IMAGE.DAT' ↵ to read in the FITS disk file *myarea:IMAGE.DAT*.

7.2. Loading an image from your *AIPS* catalog to the TV

To select an image file from your *AIPS* catalog for display you must assign several input parameters to the values carried by the chosen file. If you know the disk and catalog numbers of the file from an MCAT listing, you can initialize all of the name parameters at once with:

> INDI *n* ; GETN *ctn* ↵ where *n* and *ctn* select the required image.

Alternatively, you may specify the parameters individually.

The procedure **TVALL** bundles several important TV display functions into one verb and is therefore a good tool for a "first look" at the simpler *AIPS* TV display options. Use HELP TVGEN ↵, HELP TVCOLOR ↵ and HELP TVINTER ↵, or see § 15 of this *COOKBOOK*, to review other TV display verbs once you have experimented with **TVALL**, however.

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 'NE' ↵ negative linear.

> FUNC 'NG' ↵ inverse logarithmic.

> 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, logarithmic, and negative linear 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.

Your local *AIPS* installation may have several TVs available for image display. You may have selected a display device when you started up your copy of *AIPS* long before you began to display images. If no image appears after executing **TVALL**, make sure that you are looking at the TV device that you selected at start-up. (There may be a puzzled *AIPS* user looking at your image somewhere else). 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 to have been loaded and you are sure that your are looking at the correct (or the only) TV display, check which image channel (**TVCHAN**) is specified by your inputs. Many TV display systems have several image channels (*e.g.*, there are 4 on NRAO's I^{2}S Model 70 systems). Try:

> TVON ↵ to turn on the channel selected by your current **TVCHAN**.

If this still does not display your image, call the *AIPS* Manager for your site. If a labeled wedge does not appear on the display, you should also call your system Manager, as the display may need adjustment.

> TVPSEUDO ↵

and then follow the instructions listed on your terminal.

A rich zoo of color coding is available. 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. Remember that the TV image may not respond instantaneously to cursor motions; move the trackball or mouse slowly. Aficionados of certain Dutch color contouring schemes or of the early VLA *IMPS* "spectrum colors" will find these options in **TVPSEUDO** using button **C**.

7.2.2. Transfer functions

To fiddle the intensity transfer function of the display without reusing **TVALL** type:

> TVTRAN ↵

and then follow the instructions listed on the terminal.

If you use **TVTRAN** in a heavily loaded or slow computer, the TV display may not respond instantaneously to requested modifications. It may be necessary to move the trackball or mouse slowly. Also, notice that the transfer function is graphed in the top right corner of the image only when the zoom is used at lowest magnification.

7.3. Reading out image data using the TV cursor

There are several facilities for reading out intensity and position information from displayed images using the TV cursor:

> IMPOS ↵	reads out the two coordinate values (*e.g.*, RA and Dec) from the cursor position when any button is depressed.
> IMXY ; IMVAL ↵	reads out the image intensity and the two coordinate values (*e.g.*, RA and Dec) from the cursor position and when any button is depressed.
> CURV ↵	continuously reads out pixel coordinates and the pixel intensity in user-recognizable units at the position selected by the TV cursor. The pixel parameters are displayed in the upper left corner of the image.
> 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.
> 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.

> TVCORN 0	to restore the defaults
> TVCH 1; TVLOD ↵	to load the first image on plane 1
> GETN 2 ↵	to select the second image
> TVCH 2; TVLOD ↵	to load the second image on plane 2
> TVCH 12; TVBLINK ↵	to blink planes 1 and 2

Instructions for controlling the rate of blinking and the transfer functions appear on your terminal while **TVBLINK** is running. You may use the trackball or mouse position and the buttons to achieve the most appropriate display.

When using either of the above image-comparison tools, it may be helpful to annotate one or both of the images to distinguish them easily. The verb **TVANOT** lets you annotate a displayed image by embedding one or more strings of up to 16 characters either horizontally or vertically in the asociated image memory. The position of the bottom left corner of each string can be selected either with the TV cursor (if **TVCORN** = **0,0**) or by preloading a value of **TVCORN** before invoking **TVANOT**. **TVANOT** may also be used to write strings in the graphics (labeling) planes for "custom labeling". Type HELP TVANOT ↵ to see all the options.

7.5. Multi-plane images

On I^{2}S Model 70 displays, *AIPS* supports several powerful tools for displaying images derived from multi-plane data.

TV3COLOR is a verb that displays TV channel 1 in red, TV channel 2 in green and TV channel 3 in blue, and disables all other image channels. The pixel ranges, transfer functions, etc. of the three TV channels may be set independently of each other before or after executing **TV3COLOR**. Because both the intensity and perceived color at each pixel depend differently on the summation of the three channels, **TV3COLOR** is a powerful tool for displaying and interpreting multispectral data. Common examples of its use 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. **TV3COLOR** can also be used to color-code different types of depolarization effects from multi-frequency polarimetry.

TVHUEINT is a verb that produces a composite display in which the intensity is set by an image loaded into one channel while the hue is derived from an image loaded into another. The transfer functions used for the intensity and hue, and the roles of the two channels, may then be adjusted with the trackball. The following example sets up a display coding the intensity from a single-frequency radio continuum image and the hue from a two-frequency depolarization image:

> INDI n; GETN m ↵	Intensity image from disk n, slot m
> TVCH 1; FUNCTYP 'LG'; TVLOD ↵	Load in channel 1 logarithmically
> INDI p; GETN q ↵	Depolarization from disk p, slot q
> TVCH 2; FUNCTYP ' '; TVLOD ↵	Load in channel 2 linearly
> TVCH 12; TVHUE ↵	Make display

Instructions for adjusting the transfer functions of the display channels are given on the user terminal while **TVHUEINT** is running. **TVHUEINT** displays are especially powerful when you need to compare complex spectral, polarimetric or kinematic patterns with equally complex (*e.g.*, filamentary, spiral) intensity distributions. Common examples of continuum displays use the intensity for Stokes I or $\sqrt{Q^2+U^2}$ data and the hue for spectral or polarimetric data. Line observers commonly use the integrated line strength as the intensity image and an average line velocity as the hue image, to display kinematic information.

The verb **TVROAM** may be used to load a single image that would overflow one plane into 2 or 4 memory planes so that it can be "roamed" without the loss of information that occurs if you use **TXINC** or **TYINC** > 1. Use HELP TVROAM ↵ for details.

7.6. Displaying image cubes

There are many situations in which you may wish to display three dimensional images, commonly called "cubes". The most obvious one is a spectral line image, where the first two dimensions are spatial co-ordinates and the third dimension is the radial velocity range. Another situation in which a three dimensional display is very useful is when you have a series of continuum observations of an object as a function of time, for example, snapshots of the Sun or planets, or radar reflection images of the nearer planets.

To view these sequences of images the first step is to build a cube, using the task **MCUBE**. Ensure that all the images required have the same name and class, with each image in the series identified by its sequence number. As an example, consider the case of 50 images in a time sequence. Construct the cube as follows:

> TASK 'MCUBE'; INP ↵	to review the inputs.
> INNA 'object name ↵	to select the source.
> INCL 'IMAP' ↵	to select the input class.
> INSEQ 1;IN2SEQ 50;IN3SEQ 1 ↵	to select first and last images in the series and the increment between images.
> AXREF 1; AX2REF 50 ↵	to define the third axis of the cube.
> NPOINTS 50 ↵	set the total number of points on the third axis.
> OUTNA " ↵	use the default **OUTNAME**.
> OUTCL 'XYTCUB' ↵	specify the **OUTCLASS** as an X, Y, and T(ime) cube.
> GO ↵	to run MCUBE.

After **MCUBE** has ended, view the cube with **TVMOVIE**. You can choose sub-portions of each plane and restrict the range of planes by specifying values for the adverbs **TBLC** and **TTRC**. The number of images and the maximum image size per plane which can be loaded are dependant on your local TV display and the way in which it is set up. Often, you can run TVMOVIE by using the defaults:

> INDISK n; GETN n ↵	to select the cube.
> INP TVMOVIE ↵	to check the inputs.
> TVMOVIE ↵	to run TVMOVIE.

On the terminal screen you will see a set of instructions which allow you to change the speed and direction of your movie, and to adjust the brightness and contrast of the displayed images. The verb **REMOVIE** will restart the display after **TVMOVIE** has ended provided all image planes in the TV display still contain the original TVMOVIE frames. Individual **TVMOVIE** frames, each containing 4, 16 or 64 subframes typically, can be inspected by typing OFFZOOM ↵ after **TVMOVIE** has been stopped. The TV image planes can be shown one at a time by selecting the appropriate TV channel, *e.g.*, TVON 2; TVOFF 1 ↵.

> GO QMSPL ↵	to display the plot file on a QMS laser printer.
> GO LWPLA ↵	to display the plot file on a PostScript laser printer.

Systems having other types of display devices may have other tasks to put the results on those devices. The above tasks will select the plot file with the largest version number if INVER 0. This is generally what is desired.

8.2. Contour plots with polarization vectors (PCNTR)

PCNTR plots polarization vectors on top of contours. You must first make a polarized-intensity image and a polarization position-angle image from the Q and U images (see § 8.1.1 below). The inputs to **PCNTR** can be reviewed by:

> TASK 'PCNTR' ; INP ↵

Use:

> INDI 0 ; MCAT ↵	to review the contents of your image catalog for image names and classes.
> 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 polarized intensity image.
> IN3DI *n3* ; GET3N *ctn3* ↵	where *n3* and *ctn3* select the position-angle 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 (image 1) intensity is less than *mm* in the units of image 1.
> FACTOR *xx* ↵	to set the length of a vector of 1 (in units of total polarization) to *xx* cell widths.
> INP ↵	to review your inputs and remind you of others. Most are similar to those in **CNTR** and sensibly defaulted.
> GO PCNTR ↵	to generate the plot file, which can then be routed to output devices via **TKPL**, **TVPL**, **QMSPL** or **PRTPL**.

8.3. Plotting two images (GREYS)

The combination of laser printers and their *AIPS* driver task **QMSPL** makes the plots produced by **GREYS** particularly useful. **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. A sample set of inputs could be:

> TASK 'GREYS' ; inp ↵	to review the inputs.
> INDISK *n1* ; GETN *ctn1* ↵	to select the grey-scale image.
> 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.

then set the TV cursor to the desired beginning point for the slice, press any trackball 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.

When **SLICE** has terminated (watch the *AIPS* monitor), the file may be plotted on the Tektronix 4012 (graphics) terminal using the verb **TKSLICE**:

> INP TKSLICE ↵ to review what you can specify.

> 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.

Then:

> TKSLICE ↵ to plot the slice on the TEK screen.

Note: several slices may be put on one TEK plot. Use TKASLICE ↵ for the additional ones.

Slice files may be converted into plot files by:

> GO SL2PL ↵

The resulting plot files may then be output by:

> GO QMSPL ↵ to display the plot file on the QMS laser printer.

> GO PRTPL ↵ to display the plot file on the printer/plotter.

to obtain a better-quality plot than can be obtained with the **HARD COPY** button on the graphics terminal. In addition:

> GO TKPL ↵ to display the plot file on the graphics terminal.

> GO TVPL ↵ to display the plot file on a TV graphics plane.

A one-dimensional fit of up to four Gaussian components to slice data can be calculated using the task **SLFIT** (see § 8.3.3). Relevant displays of the data, the fit, and the residuals are also available both as verbs (like **TKSLICE**) and as options in the task **SL2PL**. See HELP SL1D or the listing in § 14 for complete lists.

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 § 8.4 above, but use INEXT 'SL' ↵ in place of INEXT 'PL' ↵.

8.6. Other one-dimensional plots

The "slices" described above may, of course, be specified to be single rows of your image (*i.e.*, **BLC(2) = TRC(2)**). There are several tasks, however, which plot rows more directly. The simplest of these is **PLROW** which 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 subimage from (100,100) to (300,300).
> YINC 3 ↵	to plot only every 3rd 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 QMSPL ↵	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**. **PROFL** makes a 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.

Two other row-plotting tasks, **PLCUB** and **XPLOT**, are designed primarily for spectral-line and other data "cubes" (see § 9). **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 2nd and 3rd axes of the cube. **XPLOT**, unlike the tasks described previously, does not create plot files. Instead, it is an interactive task which plots selected rows on the graphics terminal. Rows are selected not only with the usual inputs **BLC**, **TRC**, **YINC**, and **ZINC**, but also with inputs designed to limit the rows displayed to those with peak intensities falling within the flux range of interest.

8.7. Other displays

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.

Plus signs may be drawn on the plots produced by **CNTR**, **PCNTR**, **GREYS**, and several other tasks. In these tasks, the parameter which controls the plotting is **STFACTOR**, a scale factor for the plus signs. 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 **RUN** files.

All plots are drawn with labeled tick marks although these may be suppressed with the **LTYPE** parameter. For plots having signficantly non-linear coordinate axes, *e.g.*, wide-field images, it is useful to draw a full, non-linear coordinate grid rather than just short lines at the edges of the plot. Tasks **CNTR**, **PCNTR**, **GREYS**, and **PROFL** and the verb **TVLABEL** offer this option; enter DOCIRC TRUE ↵.

8.8. Dicomed copies of images

At the time of this edition of the *COOKBOOK*, there is no *AIPS*-standard method for recording images on Dicomed film recorders. However, *ad hoc* methods exist both at the AOC and in Charlottesville for doing

9. ANALYZING IMAGES

In order to obtain useful astronomical information from the data, software exists for the analysis of images, combining of images, estimating errors, *et al.* Only a few of the programs are described here; the others should be self-explanatory using the HELP and INPUTS files for the tasks listed in § 15. A complete list of software in *AIPS* for the analysis of images may also be obtained at your terminal by typing HELP ANALYSIS ↵.

9.1. Combining 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 derive the polarization intensity and angle from the Q and U Stokes parameter images.

9.1.1. Polarized intensity and position angle images

To compute a polarized intensity image, enter:

> TASK 'COMB' ; INP ↵	to review the required inputs.
> INDI 0 ; MCAT ↵	to help you find the catalog numbers of the Q and U images that you want to combine.
> INDI *n1* ; GETN *ctn1* ↵	where *n1* and *ctn1* select the Q image.
> IN2D *n2* ; GET2N *ctn2* ↵	where *n2* and *ctn2* select the U image.
> OUTN '*xxxx*' ↵	where *xxxx* is your choice of outfile name for the polarized intensity image.
> OUTC '*ccc*' ↵	where *ccc* is your choice of classname for 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.

The *AIPS* monitor should note **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*' ↵	where *yyyyy* is your choice of outfile name for the polarization angle image.
> OUTC '*ddd*' ↵	where *ddd* is your choice of class name for 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.

The *AIPS* monitor should note again **TASK COMB BEGINS** followed by a listing of the **POL. ANGLE** algorithm. Once both **COMB** tasks have terminated with **COMB: APPEARS TO END SUCCESSFULLY** messages on the message monitor, you should find the requested images in your catalog:

> MCAT ↵	to list the images in your catalog.

9.2. Image statistics and flux integration (IMEAN, IMSTAT, TVSTAT, BLSUM)

The task **IMEAN** is used to determine the statistics of the image over a specified rectangular 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* ↵	where *n* and *ctn* select the disk and catalog numbers of the relevant image file.
> 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.
> GO ↵	to run the task.

The statistics will appear on the *AIPS* monitor. 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 § 8.9, type one of:

> GO TKPL ↵	to display histogram on the Tektronix.
> GO PRTPL ↵	to display histogram on the printer/plotter.
> GO QMSPL ↵	to display histogram on the laser printer.

The verbs **TVSTAT** and **IMSTAT** provide similar functions to **IMEAN** without the histogram option. 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**. **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" (see § 10.4) in almost any transposition. However, numerous continuum applications also exist (*e.g.*, polarization, comparison across frequency). Type EXPLAIN BLSUM ↵ for a description of the operation.

9.3. Fitting of images

There are three programs which estimate the position and intensity of a component on a two-dimensional image. The simple and fast method is the verb **MAXFIT**. This 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. 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.

9.3.1. Parabolic fit to maximum (MAXFIT)

MAXFIT's speed makes it useful for simple regions. Type:

> HELP MAXFIT ↵ to get a good explanation of the algorithm.

These 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 and hit any of the buttons behind the trackball. The fit will then appear on your terminal.

9.3.2. Two-dimensional Gaussian fitting (IMFIT)

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* ↵	where *n* and *ctn* select the disk and catalog numbers of the required image.
> 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 FWHP of 6 by 4 pixels with the major axis at position angle 20 deg.
> DOWID FALSE ↵	to hold all of the widths constant in the fitting process (if required).
> INP ↵	to review inputs.
> GO ↵	to run the task.

To save execution time, 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. **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 and to estimate the errors. Comparison of the results of **IMFIT** and **JMFIT** will sometimes be instructive. It is wise to treat the results of **MAXFIT**, **IMFIT** and **JMFIT** with considerable caution, particularly the estimates of the errors in the component widths after deconvolution.

Use RUN INPFIT ↵ (see § 14.6) 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 subimage window with the TV cursor (it uses verb **TVWINDOW**) and then to point the TV cursor at the peaks of each of the Gaussians, hitting any button when the cursor is correctly placed. The inputs **GMAX**, **GPOS**, **BLC**, and **TRC** are set in this way.

9.4. 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 and VLBI data reduction are described in §§ 10 and 11, respectively.

Chapter 11 of *Synthesis Imaging in Radio Astronomy** covers the topic of image analysis in more detail.

9.4.1. 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, 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 *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. Most of them require very large internal buffers and hence, are available only on virtual-memory computer systems.

The simplest geometric conversion is a regridding of the data with new axis increments and dimensions with no change in the type of projection or coordinates. Two tasks, **GEOM** and **LGEOM**, perform this basic function and also allow rotation of the image. One use of these tasks is to obtain smoother displays by regridding a subimage on a finer grid. To rotate and blow up the inner portion of a 512^2 image, enter:

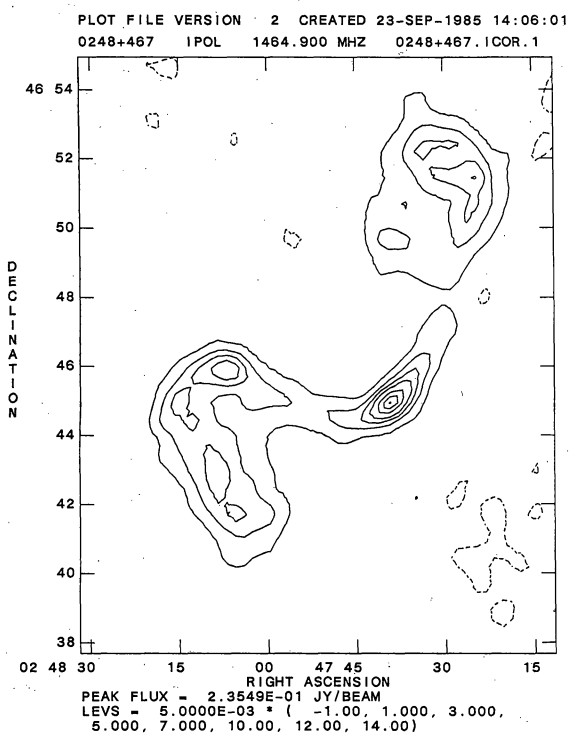
> TASK 'LGEOM' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to select the image.
> BLC 150 ; TRC 350 ↵	to select only the inner portion of the image area.
> IMSIZE 800 ↵	to get an 800^2 output image. This will allow the subimage to be blown up by a factor of 3 and rotated without having 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° CCW (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. Type EXPLAIN LGEOM ↵ for more information and advice. **GEOM** is a small-buffer version of **LGEOM**. As a result, it works on non-virtual-memory computers, but is very limited in the amount of rotation it can do on larger images. These tasks should be used for rotation only if the pixels are square.

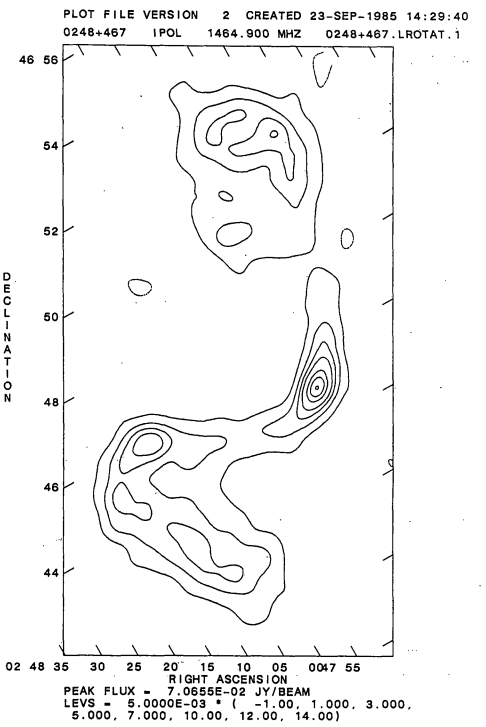
---

* *Synthesis Imaging in Radio Astronomy*, A collection of Lectures form 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)

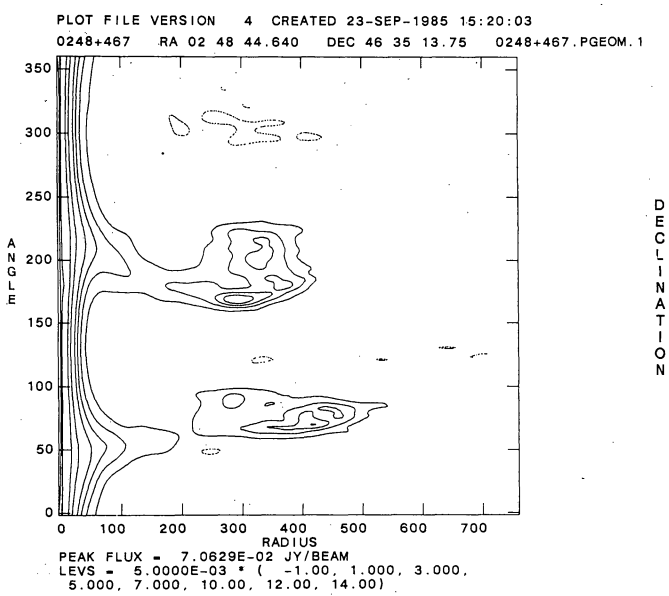
8.4.4. Examples



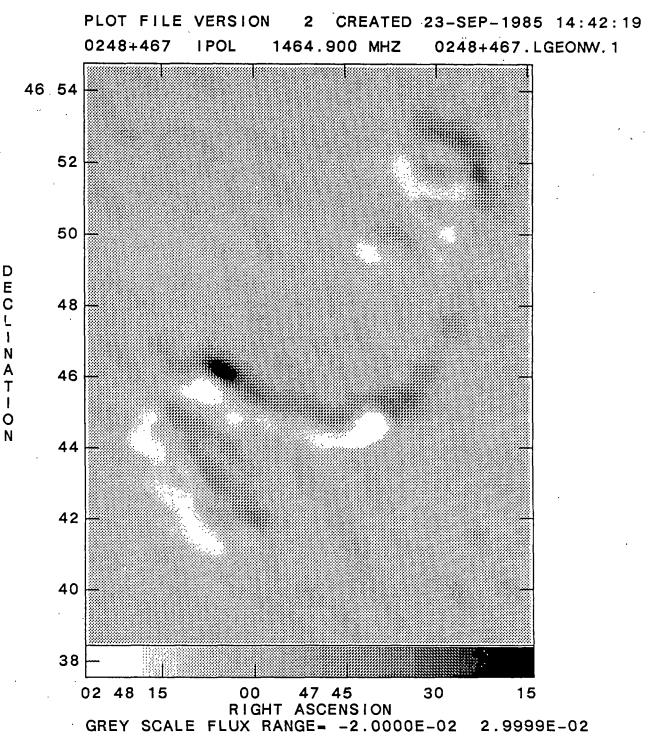
input image



LGEOM with rotation and interpolation



PGEOM



NINER: **'NW '** derivative

3. Use **CALIB** to generate complex gain solutions. (Note that **CALIB** is a powerful task with many options. Do not use it blindly. Read the **EXPLAIN** file and perhaps also consult an expert before using it if you are unsure of how best to steer the program.)
4. Generate a new calibration (CL) table by applying the *self-calibration* solutions to the best earlier CL table with **CLCAL**.
5. Use **HORUS** (applying the new CL table) to generate a new map. Inspect the new map carefully and decide whether the image has improved or not. If no significant improvement can be seen, then you may not have sufficient signal-to-noise to use *self-calibration*. Alternatively, your data may be corrupted by interference or by spurious data. Chapter 10 of *Synthesis Imaging in Radio Astronomy* may help you decide if some data editing is required. If the image has improved noticeably, ask yourself if such an improvement would be noticed in the (line minus continuum) image *i.e.*, would the "dynamic range" improvement of the (line minus continuum) image be above the noise?
6. If so, copy over the new CL table to the line database (if the solution was generated from Channel 0 data) and use this CL table in future steps *e.g.*, **POSSM**, **HORUS**, **SPLIT**.
7. Use **HORUS** to make a data "cube" of images from the line database. (It may be necessary to use **MX** for wide-field imaging and cleaning since **HORUS** is limited to images of 1024×1024 on a Convex. In this case, you will have to run **SPLIT** applying the required CL table to write the calibrated target source data to a "single-source" file. This file will require sorting to 'XY' order prior to mapping and cleaning with **MX**.)
8. Use **ALTDEF** to update the velocity in the file header, if required, and proceed to § 10.1.3.

10.1.2. Self-calibrating data from elsewhere

If your data have come from some other calibration package, then follow the prescription below.

Starting from calibrated *uv* data *from elsewhere*,

1. Use **UVLOD** to load data from FITS tape (see § 10.2).
2. Use **IMHEAD** to check the headers to see which frequency channels are in the file.

or, starting from images,

1. Use **IMLOD** to load images from FITS tape and then go to step 9.
2. Use **MCUBE** to assemble the images into a "cube" then go to § 10.1.3.

Do you need self-calibration? Decide on this by first processing the broad-band continuum channel or a line channel with a very strong signal:

3. Use **HORUS** on a channel with a strong line, or use channel 0.
4. Use **APCLN** to CLEAN it.
5. Use **CALIB** to self-calibrate the TB-sorted *uv* data.
6. Repeat steps 3 & 5 to iterate to an optimal solution.

To image channels 8 to 15 and make one beam (channel 8):

> NMAPS 8 C_R	to make 8 images.
> CHANNEL 8 C_R	to start at the eighth frequency in the file.
> GO UVMAP C_R	to run **UVMAP**.

As these illustrations show, the meaning of **CHANNEL** in **UVMAP** is the frequency pixel in the *uv* data file, at which one wants to start making a total of **NMAPS** images. The examples also use the global nature of **AIPS**' inputs; those, such as **INNAME**, which don't change, don't have to be entered for each execution of **UVMAP**.

To image channels 8 through 23 using **MX**:

> INNA 'N315' C_R	to select the source.
> INSEQ 2 C_R	to select the line data file.
> STOKES 'I' C_R	to make intensity images.
> NPOINTS 1 C_R	do not average channels.
> CHANNEL 0 C_R	do not restart.
> BCHAN 8 C_R	first channel to image.
> ECHAN 23 C_R	last channel to image.
> CHINC 1 C_R	channel increment.
> GO MX C_R	to run **MX**.

This makes images and beams of channels 8 through 15.

10.4. Building a cube

If the spectral line maps were made with **UVMAP** or were loaded as single frequency images from tape (with **IMLOD**), it is necessary first to use **MCUBE**.

The task **MCUBE** is used to put a series of 2- or 3-dimensional images at different frequencies into a 3-dimensional image ("cube") cataloged on disk. The cube can be transposed to create images with velocity on one axis and RA or DEC as the other axis of each plane. Many spectral-line functions make sense only on cubes. Once the images have been put into a cube, you can inspect all the data with one command, **TVMOVIE**. And, if you want to perform the same operation on many images, it is most convenient to do it on a cube instead.

To put 31 frequency channel images into one cube, type:

> TASK 'MCUBE' ; INP C_R	to review the inputs.
> INNA 'N315' C_R	to select the source.
> INCL 'IMAP' C_R	to select input image class for all images.
> INSEQ 1 ; IN2SE 31 ; IN3SE 1 C_R	to set the first and last image and the step in the task's loop over input sequence number.
> AXREF 1 ; AX2REF 31 C_R	to set the pixel coordinate of the first and last image on the third axis in the cube.
> NPOINTS 31 C_R	to set the total number of points on the third axis.
> OUTN ' ' C_R	to use the default **OUTNAME**.
> OUTCL 'LMVCUB' C_R	to specify, in **OUTCLASS**, the order of axes.
> GO C_R	to run **MCUBE**.

> OFFZOOM ↵ to show one of the memory planes with a number (16 in our example) of images.

To look at a different TV channel with the remaining images, type:

> TVON 2 ; TVOFF 1 ↵ depending on which one you were looking at.

Be aware that the order of planes is reversed in even numbered planes.

In general, if you want to use any of the other display programs, you must specify which plane in the cube you want to see. So, *e.g.*, if you want to do **TVALL** on channel 16, which is on pixel 16 at 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 tasks.

10.7. Subtracting the continuum

The continuum can be subtracted in a variety of ways. You can subtract the continuum *either* in the map plane *or* in the *uv* plane. Which technique is best depends on the particular circumstances. If you require high dynamic range, then you should do the continuum subtraction in the *uv* plane.

10.7.1. Subtraction in the map plane

XBASL may offer the best method, but the standard way is still to combine a group of line-free dirty channel images and subtract that from the cube. Let us assume that there is no line emission in channels 1 through 10 and 22 through 31. We first make two images of channels 1–10 and of 22–31 with **SQASH**. Type:

> TASK 'SQASH' ; INP ↵ to review the inputs.

> INNA 'N315' ; INCL 'LMVCUB' ↵ to select the data cube.

> BLC 0 0 1; TLC 0 0 10 ↵ to select the first and last image in the loop.

> BDROP 3 ↵ to indicate that the summing is along the third (or frequency) axis.

> GO ↵ to run **SQASH**.

To do the other set of images:

> BLC(3) 22 ; TRC(3) 31 ↵

> GO ↵

With **COMB**, these images can be averaged by typing:

> TASK 'COMB' ; INP ↵ to review the inputs.

> INNA 'N315' ; INCL 'SQASH' ↵ to select the first image.

> INSE 1 ↵

> IN2NA 'N315' ; IN2CL 'SQASH' ↵ to select the second image.

> IN2SE 2 ↵

> OPCODE 'SUM' ↵ to take $\mathbf{APARM}(1) \times \mathbf{MAP}_1 + \mathbf{APARM}(2) \times \mathbf{MAP}_2$.

> APARM 0.05, 0.05, 0 ↵ to average the images.

> OUTN 'N315CON' ↵ to select an output file name.

```
> GO CR                                          to run COMB.
```

and then subtracted from the cube by entering:

```
> TASK 'COMB' ; INP CR                           to review the inputs.
> INNA 'N315' ; INCL 'LMVCUB' CR                 to select the cube.
> INSE 1 CR
> IN2NA 'N315CON' ; IN2CL 'COMB' CR              to select the continuum.
> IN2SE 1 CR
> OUTN 'N315L-C' ; OUTCL 'LMVCUB' CR             to give it an output name.
> OPCODE 'SUM' CR
> APARM 1, -1 CR                                 to take MAP1 − MAP2.
> GO CR                                          to run COMB.
```

10.7.2. Subtraction in the *uv* plane

In general, subtraction in the map plane will work fine; however, in some cases, *e.g.*, if you have used very broad bandwidth or if you have a very strong continuum source, you might want to do the subtraction in the *uv* plane. There are two ways to do this. In the first method, you initially generate an image of the continuum emission. One way is to use **SQASH** and **COMB** to combine the beams and the dirty maps of the line-free channel images and then use **MX** to CLEAN the mean continuum dirty image with the mean continuum beam. Alternatively, you could use the channel selection parameters in **MX** to clean the appropriately selected channels to give a continuum image. Another way is to generate a continuum database from the spectral line data with **AVSPC** (this has much more flexibilty in the selection of the channels than has **MX**) and then CLEAN the continuum data with **MX**. Once you have formed a CLEAN image from the continuum data, use **UVSUB** to subtract the CLEAN components from *all* of the *uv* data. This method works best when the continuum can be represented well by discrete components.

A new task, **UVBAS**, is available which will attempt to perform a a spectral baseline subtraction. It first forms a continuum database from the average over two ranges of spectral line channels. This continuum is then subtracted from the spectral line database in the *uv* plane. This task appears to be most successful in problems where the total bandwidth is small and *extended* continuum emission is present.

10.8. Decomposing the cube into separate images

Most programs work on cubes. However, you may find it convenient, on occasion, to work with single-channel images. To separate the images from the cube, use task **SUBIM**; type:

```
> TASK 'SUBIM' ; INP CR                          to review the inputs.
> INNA 'N315L-C' ; INCL 'LMVCUB' ; INSE 1 CR     to select the cube.
> OUTN 'N315L-C' ; OUTCL 'IMAP' CR               to give it an output name and class.
> BLC 0 ; TRC 0 CR                               to select full planes.
> FOR J=10:20 ; BLC(3)=J ; TRC(3)=J ; PRIN J ; OUTSE J ; GO ; WAIT ; END
```

This runs the program 11 times, taking planes 10 to 20 and creating separate images for them. To put the images back into the cube after they have been modified, see § 10.4.

> INDI *n* ; GETN *ctn* C_R to select the **RGBMP** output.

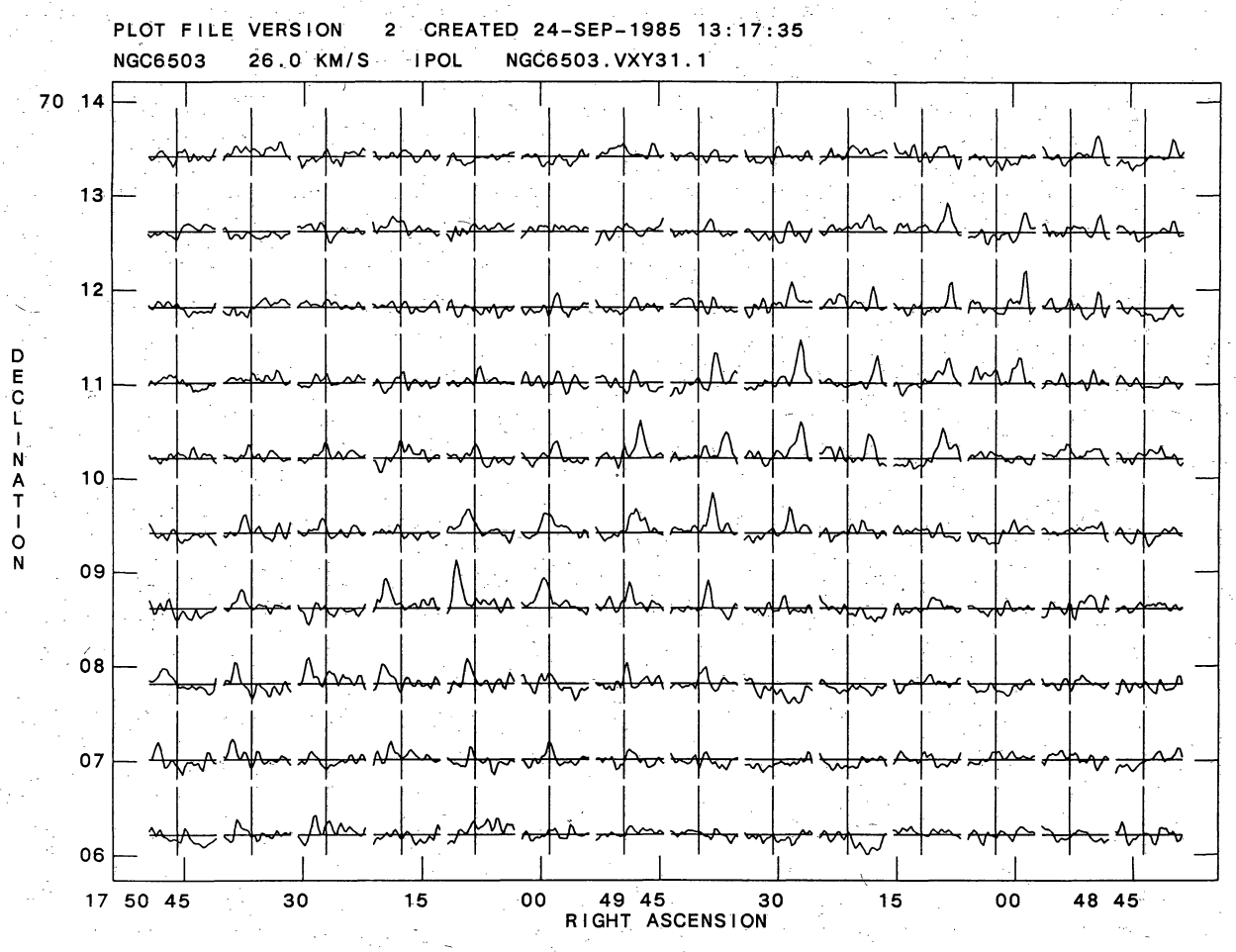
> TV3C ; FOR TVCH = 1 TO 3 ; TBLC(3) = TVCH ; TVLO ; END C_R

If you prefer to fit Gaussians instead of calculating moments, the program **XGAUS** can be used. It is a good idea to use **XPLOT** first to look at (a sample of) the profiles, before you do any Gaussian fitting. In most cases, it is perhaps preferable to use the interactive mode, so that you can see what is happening, but be aware that it might be rather time-consuming. The experimental task **NNLSQ** performs a constrained, non-linear deconvolution of the spectra in a VLM cube.

GAL fits models of galaxy rotation to images of the predominant velocity (*e.g.*, the first moment images written by **XMOM**, **XGAUS**, or **MOMNT**).

You can plot all your channel images in one plot on the plotter. Star positions and beams can be included. All this is done with the task **KNTR**.

10.11. Sample display from PLCUB



9. REDUCING VLBI DATA IN *AIPS*

This chapter describes the reduction of VLBA or other VLBI data in *AIPS*. A step-by-step recipe is presented. We also include some background information concerning the structure of VLBI data sets and 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 *Very Long Baseline Interferometry and the VLBA* (ASP Conference Series, No. 82, 1995; order information is available from the Astronomical Society of the Pacific, 390 Ashton Avenue, San Francisco, CA 94112-1722).

A list of programs of particular interest for VLBI may be found in Chapter 13 or displayed on your terminal by typing ABOUT VLBI ↵ or APROPOS VLBI ↵. Remember, the best and most complete information available on all *AIPS* verbs and tasks may be found in their **EXPLAIN** files.

Most types of VLBI data, once read into *AIPS*, appear very similar in structure as far as the user is concerned. We shall concentrate on describing the reduction path for VLBA data, but most operations also apply to MkIII and MkII data. Where appropriate, we shall draw the reader's attention to any differences. In particular, § 9.12 will deal with reading data from a MkIII correlator into *AIPS* and the steps necessary to prepare such data for calibration.

Some of the VLBI related tasks require the ability to read files resident outside of *AIPS*. In order to communicate to *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.

9.1. Loading and inspecting data

This section describes the loading and inspection of VLBI data, with particular emphasis on data correlated at the VLBA correlator. This includes a discussion of **FITLD** in § 9.1.1, correlator corrections in § 9.1.2, and general data examination methods in § 9.1.3.

9.1.1 Loading VLBA data using FITLD

Data generated by the Socorro VLBA correlator are loaded from DAT (or Exabyte) tape 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. 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.

Typical inputs to **FITLD** would be;

> TASK 'FITLD' ; INP ↵	to review the inputs.
> INTAPE *n* ↵	to specify the input tape number.
> INFILE ' ' ↵	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 ' ' C_R	to load any type of file found.
> NCOUNT 20 C_R	to load 20 tape files.
> DOUVCOMP 1 C_R	to save disk space by writing compressed data.
> DOCONCAT 1 C_R	to concatenate files with same data structure into one disk file.
> DIGICOR 1 C_R	enable VLBA digital corrections.
> DELCORR 1 C_R	enable VLBA delay decorrelation corrections.
> GO C_R	to run the program.

Because the data files tend to be very large, you should usually write compressed data (**DOUVCOMP=1**). This takes about 1/3 of the space of uncompressed data, but causes information about the weights of individual spectral channels and IF channels to be lost. 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 (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**.

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 (see § 9.13). Note that, at this point, the data written by **FITLD** are sorted correctly and are immediately ready for further processing. The sorting, concatenating, and merging required for MkIII/MkII data (see § 9.12.2) are not necessary. **FITLD** can also be used to load archived *AIPS* data previously written to tape using **FITTP**, as described in § 5.1.2. In this case the VLBA-specific adverbs, such as those enabling digital and delay corrections, are not active.

9.1.2 Correlator corrections

Corrections for correlator scaling and quantization are performed in **FITLD** under control of adverb **DIGICOR**. It is recommended that these corrections be enabled (i.e. **DIGICOR=1**). However, 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.

Adverb **DELCORR** enables amplitude corrections for known delay decorrelation losses in the VLBA correlator, as described in *AIPS* Memo. 90. Setting **DELCORR=1** will create a correlator parameter frequency (**CQ**) table for each file written by **FITLD**. 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 the **CQ** table is missing. For older **FITLD** files the **CQ** table can be generated using task **FXVLB**. This must be run before any changes in the frequency structure of the file.

The VLBA correlator may label the polarization of the output data incorrectly under certain limited circumstances. This is something that has been corrected over time but it is well to check the polarization labeling at this point. The cases where mis-labeling may occur are: i) dual parallel-hand correlation (i.e. RR and LL only), and: ii) single-polarization LL correlation. Full polarization correlation data (i.e. RR, LL, RL, LR) have generally been labeled correctly.

In case (i), the effect is to double the number of AIPS IF channels and label them all as RR. This can be corrected using interim task **FXPOL**, which will correct the mislabeling within *AIPS*. Calibration tables

> 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.
> NCOUNT 4 ↵	to plot 4 baselines/page.
> GO ↵	to run the program.

An example **VPLOT** output appears at the start of § 9.6. 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.

Another task which can display data as a function of time is **IBLED**. 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. Tasks **TVFLG** and **SPFLG** are also useful in this way.

Supplied with your VLBI data will be a number of important tables used for calibration and many more are generated as data calibration proceeds. Section 9.13 summarizes the contents of each of these tables. One of the most important tables is the default (*i.e.*, version 1) calibration or **CL** table. This table often contains important information such as the results of phase-cal calibration (for data from MkIII correlators) or the geometrical model used by the correlator. To protect this valuable table, it is wise to copy it to version 2 using the task **TACOP**. If for some reason you delete all **CL** tables, a default **CL** table can be recomputed using task **INDXR**. Note that the task **TASAV** can be used to copy all the tables 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**.

9.2. Data editing

Before proceeding to calibrate, you should first flag any obviously bad data. Task **UVFLG** is capable of reading a text file listing periods of known errors, and flagging the corresponding data. For VLBA stations such a text file can be generated from the calibration file deposited after your observation in the appropriate directory on the NRAO computer "aspen." You can obtain the logs by running ftp to **aspen.aoc.nrao.edu** (or **[146.88.5.8]**) and logging in as **vlbiobs** using a password which you must obtain from NRAO staff. The log information is stored in subdirectories corresponding to the month and year of your observation and will be contained in a file called *xxxx***cal.vlba** (where *xxxx* is your experiment code). Using an editor, extract the editing information from the rest of the data in this file (*i.e.*, system temperatures, weather information, etc.).

> INEXT 'CL' ; INVER 2 CR	to choose **CL** table version 2.
> DOTV 1 CR	to plot on the TV screen; otherwise, create plot extension files.
> NCOUNT 5 CR	to plot 5 antennas per page.
> GO CR	to run the program.

SNPLT can also be used to plot quantities from other tables generated by the calibration process including the contents of **SN** ("solution tables"), **TY** ("system temperature tables"), and **PC** ("phase-cal tables").

As the calibration process proceeds, both amplitude and phase corrections are incorporated into the **CL** tables. Unlike VLA data, which are roughly amplitude calibrated (in Janskys) when loaded, VLBI correlator output is in terms of dimensionless "correlation coefficients." Therefore, 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 § 9.6.

Note that, unlike VLA users, VLBI users normally do not attempt to calibrate the absolute phase of the data using external calibrator sources. VLBI users just calibrate out the phase derivatives with respect to time and frequency. The absolute phases are normally left uncalibrated and "self-calibration" methods are then used to generate images (see § 9.10). The alternative of absolute phase calibration using an external calibrator, known as "phase-referencing" in VLBI circles, is much more difficult in VLBI than for the VLA. It is possible to do such calibrations, but their description is beyond the scope of the *CookBook*.

Optimum fringe-fitting results are obtained if amplitudes are calibrated first, since, in this case, the data can be weighted optimally. In addition if the data are amplitude calibrated, the quoted SNRs will be correct; this may not be the case if uncalibrated data are used. For these reasons, we describe the process of amplitude calibration first in § 9.4. Then, in §§ 9.5 and 9.6, 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 the sensitivities on all baselines are roughly the same), the order of the two calibration steps may be reversed.

9.4. Amplitude calibration

This section describes amplitude calibration methods for VLBI data. In the **15JUL95** and later versions of *AIPS* two new amplitude calibration tasks, **ANTAB** and **APCAL**, have been added which supersede **ANCAL**. The new tasks offer greater flexibility in amplitude calibration, including corrections for atmospheric opacity. **ANCAL** remains part of the *AIPS* distribution, but **ANTAB** and **APCAL** are the recommended route for amplitude calibration.

VLBA data, particularly if observed in two bit quantization, require a preliminary amplitude correction for digital sampler bias. This is discussed in § 9.4.1. The use of a priori T_{sys} and antenna gain information is described in § 9.4.2, which includes a discussion of both **ANCAL** and the new amplitude calibration tasks **ANTAB** and **APCAL**.

9.4.1 Digital sampler bias

The voltage threshold levels in the digital samplers at the VLBA antennas may differ from their optimum theoretical values, and this may vary across the network as a whole. This sampler bias, which is only

significant in two bit quantization, may introduce an antenna-based amplitude offset. In full polarization observations this may appear as an amplitude offset between **RR** and **LL**. Typically this is $5-10\%$ but values as high as 20% have been observed. This can be corrected using the measured level in the autocorrelation spectra using task **ACCOR**. 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 ↵	select all data.
> SOLINT 60 ↵	solution interval for sampler corrections (min).
> SNVER 0 ↵	write a new SN table.
> GO ↵	to run the program.

The correction factors are expected to be fairly stable over time but it is well to examine the solution (**SN**) table using **SNPLT** for any bad points or inconsistent values.

For one-bit quantization, no significant sampler bias correction expected. In this case if the **ACCOR** gain factors deviate from unity this may point to overall scaling or b-factor errors. In this respect it is recommended that **ACCOR** be run on one bit data as a consistency check.

The **SN** table can be edited and smoothed using task **SNSMO**. The sampler bias corrections are then applied using **CLCAL**. Inputs required for this step are:

> TASK 'CLCAL' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> CALSOUR ' ' ↵	use all solutions produced by **ACCOR**.
> SOURCE ' ' ↵	apply corrections to all sources.
> OPCODE 'CALI' ↵	to apply **SN** tables to a **CL** table.
> INTERPOL ' ' ↵	to use linear vector interpolation with no **SN** table smoothing.
> SMOTYPE ' '; INTPARM 0 ↵	to clear interpolation and smoothing parameters.
> SNVER 1 ; GAINVER 2 ; GAINUSE 3 ↵	to specify which **SN** table(s) and which input and output **CL** tables are used.
> REFANT 0 ; BADDISK 0 ↵	amplitude calibration being applied; no explicit reference antenna required.
> INP ↵	to check the inputs.
> GO ↵	to run the task.

9.4.2 A priori amplitude calibration

As discussed in § 9.3 this section assumes that **CL.1** has been copied to **CL.2** at the outset of calibration using **TACOP**. This is a recommended precaution to preserve the initial **CL** table created by **FITLD**. Task **TASAV** can also be used to create an extra backup copy.

This section describes the calibration of VLBA data using a priori T_{sys} and antenna gain information. This can be achieved using tasks **ANCAL** or **ANTAB** and **APCAL**, which are described in § 9.4.2.1 and § 9.4.2.2 respectively. An initial discussion concerning the common aspects of preparing the external amplitude calibration file is given below.

Eventually all amplitude calibration information will be available in tables written by the VLBA correlator. However, at present this information must be provided in an external text file. To create the required

in **FITLD**. Also in the control group, the parameters **BIF**, **EIF** and **POLNO** can be used as an alternative to **INDEX** as a limited "global" method of assigning T_{sys} values to *AIPS* IF channels.

Be certain to enter T_{sys} values at the end of scans immediately before source changes in order to avoid interpolation problems for sources of greatly differing flux density.

Having created the input text file, run **ANCAL**, *e.g.*,

> TASK 'ANCAL' ; INP $\subset_R$	to review the inputs.
> INDISK *n* ; GETN *ctn* $\subset_R$	to specify the input file.
> INFILE 'MYVLB:BC25CAL.VLBA' $\subset_R$	to specify the text file.
> GAINVER 2 $\subset_R$	to update **CL** table 2.
> GO $\subset_R$	to run the program.

If **ANCAL** fails to run first time, check the syntax of your text file carefully. Also look for negative or undefined ("NaN") values within the T_{sys} blocks. Note that, as well as modifying **CL** table 2, each time **ANCAL** is run, a new **TY** table is also appended to include the conversion factors from correlation coefficients to Janskys. These **TY** tables will contain the antenna system temperatures in Kelvins. If your **CL** table is incorrect due to an error in the text file, you should use **EXTDEST** to delete both the **CL** and **TY** tables (highest versions *only*). Then re-run **TACOP** to regenerate the **CL** table and, after correcting the error, re-run **ANCAL**. A useful check after running **ANCAL** is to run **SNPLT** to plot the amplitude correction factors in the new **CL** table and the system temperatures in the **TY** table.

Note that, in the future, the VLBA correlator will write system temperature information directly into a **TY** table and antenna gains into a **GC** table, removing the need for **ANCAL** to read an external text file.

9.4.2.2 Amplitude calibration using ANTAB and APCAL

These tasks, available in **15JUL95** and later versions of *AIPS*, are the recommended path for a priori amplitude calibration. **ANTAB** offers greater flexibility in reading the calibration text file, allows IF-dependent and tabulated antenna gains and can be used to append to existing *AIPS* tables, as will be written by the VLBA correlator in future.

ANTAB reads the external calibration file and fills the *AIPS* system temperature (**TY**) and gain curve (**GC**) tables. **APCAL** takes these tables as input and generates an amplitude solution (**SN**) table, allowing an optional solution for atmospheric opacity. The table is applied using **CLCAL**, at which point smoothing and primary amplitude calibrators can be selected.

The user is advised to read the **ANTAB** help file carefully. In common with **ANCAL**, the INDEX keyword is used to assign the tabulated T_{sys} data to individual *AIPS* IF channels and polarizations. Greater flexibility in the INDEX keyword is permitted as described in the help file. The CONTROL group at the head of the calibration file is used only to specify a default index mapping. If the IF channel order in the calibration file and the *uv* file are identical it is not required. The default b-factor and 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, where flux densities can be inserted using **SETJY** if necessary. The GAIN records have a more flexible format than **ANCAL** and IF-dependent and tabulated gains are allowed.

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 ignore calibration data for which there are no corresponding *uv* data.

task **PCCOR** to generate a **SN** table which corrects for the single-band instrumental phase and delay offsets. A short calibrator scan must be specified which is used by **PCCOR** to resolve any 2π phase ambiguities in the pulse-calibration data. Typical inputs to **PCCOR** are:

> TASK 'PCCOR' ; INP C_R	to review the inputs.
> INDISK *n* ; GETN *ctn* C_R	to specify the input file.
> TIMERANG 1 2 15 0 1 2 20 0 C_R	short scan on a calibrator.
> SNVER 2 C_R	write solutions to **SN.2**.
> INVER 1 C_R	input **PC** table version.
> REFANT 5 C_R	specify reference antenna.
> SUBARRAY 1 ; FREQID 1 C_R	set subarray and freqid.
> CALSOUR '3C345',' ' C_R	set calibrator source name.
> GO C_R	to run the program.

The resulting solution table is applied using **CLCAL**. Examine the corrected data using **POSSM** to determine if the instrumental phase and delay offsets between the IF channels have been removed correctly.

In order to check whether your data from a MkIII correlator contains phase-cal data, use **SNPLT** on **CL** table version 1 to plot the phases. If the values are non-zero, phase-cal data are present. To check that the 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 gradient between channels and small gradients within each channel (which should be approximately the same from IF channel to IF channel, but not necessarily the same as the gradient between channels). If these conditions hold for all baselines, you can proceed directly to § 9.6.

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 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 C_R	to review the inputs.
> INDISK *n* ; GETN *ctn* C_R	to specify the input file.
> GAINVER 2 C_R	to specify which **CL** table to modify.
> OPCODE 'PCAL' ; CLCORPRM 0 C_R	to set phase-cals to zero.
> INP C_R	to check the inputs.
> GO C_R	to run the task.

Now 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 § 9.6. Note that it is simplest to choose a single short section of data using **TIMERANGE** and to set **SOLINT** equal to this interval 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 C_R	to review the inputs.
> INDISK *n* ; GETN *ctn* C_R	to specify the input file.
> CALSOUR '0954+658 ' C_R	to select a strong calibrator source.

> TIMERANGE 0,16,0,0,0,16,2,0 C_R	to select a single short scan.
> DOCALIB 1 ; GAINUSE 2 C_R	to apply the amplitude calibration from **CL** table 2.
> FLAGVER 0 C_R	to apply most recent flag table.
> SMODEL 0 C_R	to use the null (point-source at the origin) source model.
> REFANT 1 C_R	to specify a reference antenna that will give fringes to most other antennas.
> SOLINT 2 C_R	to set the solution interval in minutes; do not exceed the atmospheric coherence time.
> APARM 2 , 0 , 0 , 0 , 0 , 1 , 0 C_R	to require at least 2 antennas, to print some additional information and, with **APARM(5) = 0**, to solve for the rate, single-band delay and phase of each IF separately.
> DPARM 1 , 0 , 0 , 2 , 0 C_R	to use only 1 baseline in the initial (coarse) fringe search, to search the Nyquist window in delay defined by the frequency spacing, to search the Nyquist window in rate defined by the integration time, to specify the integration time in seconds, and to do normal least-squares solutions, frequency averaging on output, and re-referencing to a common antenna, respectively. Use **DTSUM** to determine **DPARM(4)**.
> SNVER 1 C_R	to write solutions into SN table 1.
> ANTWT 0 C_R	to apply no additional weights to the antennas before doing the solutions.
> INP C_R	to check the inputs.
> GO C_R	to do the fit.

Since we want to apply a time constant phase offset to each IF channel, we must first set the residual rates in the output **SN** table to zero before creating a new **CL** table. We use the task **SNCOR** to do this:

> TASK 'SNCOR' ; INP C_R	to review the inputs.
> INDISK *n* ; GETN *ctn* C_R	to specify the input file.
> SOURCE ' '; TIMER 0 C_R	to do all sources over all times.
> BIF 1 ; EIF 0 C_R	to modify all IFs.
> SNVER 1 C_R	to change **SN** table 1.
> OPCODE 'ZRAT' C_R	to specify the function to zero rates.
> PHASPRM 0 ; SNCORPRM 0 C_R	to zero all parameters for safety's sake.
> INP C_R	to check the inputs.
> GO C_R	to set the rates to zero.

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 and **ANTENNAS** to this antenna plus all of the antennas *not* found in the first run. Again the rates must be set to zero for the output **SN** table using **SNCOR**.

The phase solutions in the **SN** table(s) are interpolated onto a calibration or **CL** table using task **CLCAL**.

> TASK 'CLCAL' ; INP C_R	to review the inputs.
> INDISK *n* ; GETN *ctn* C_R	to specify the input file.
> CALSOUR '0954+658' C_R	to use solution found on the specified calibrator only.
> SOURCE ' ' C_R	to apply them to all sources.
> OPCODE 'CALI' C_R	to apply **SN** tables to a **CL** table.

> INTERPOL 'AMBG' ⏎	to interpolate **SN** solutions onto the **CL** table using **SN** solutions immediately before and after each **CL** entry with the residual fringe-rate information to resolve any phase ambiguities.
> SMOTYPE ' ' ⏎	to do no smoothing; we recommend using **SNSMO** to smooth VLBI data.
> SNVER 2 ⏎	to select **SN** table 2.
> GAINVER 3 ⏎	to specify the input **CL** table to which solutions will be applied.
> GAINUSE 4 ⏎	to specify the output **CL** table containing updated calibration information.
> GO ⏎	to run the program.

Note that, when running **FRING**, we first applied **CL** table 3 before deriving the **SN** table solutions. Therefore it is important to set **GAINVER=3**, so that the output table **GAINUSE=4** contains the full calibration solution, *i.e.*, the original calibration in **CL** table 3 plus the incremental change as derived by **FRING**. Specify the version of the **SN** table to use by setting **SNVER** explicitly. Note that, if **SNVER=0**, *all* **SN** tables are combined and applied, not just the highest version number. This can be the source of serious errors. If two **SN** tables contain two similar attempts at finding fringe-fit solutions and **SNVER=0** then, effectively, **CLCAL** will apply the solutions twice!

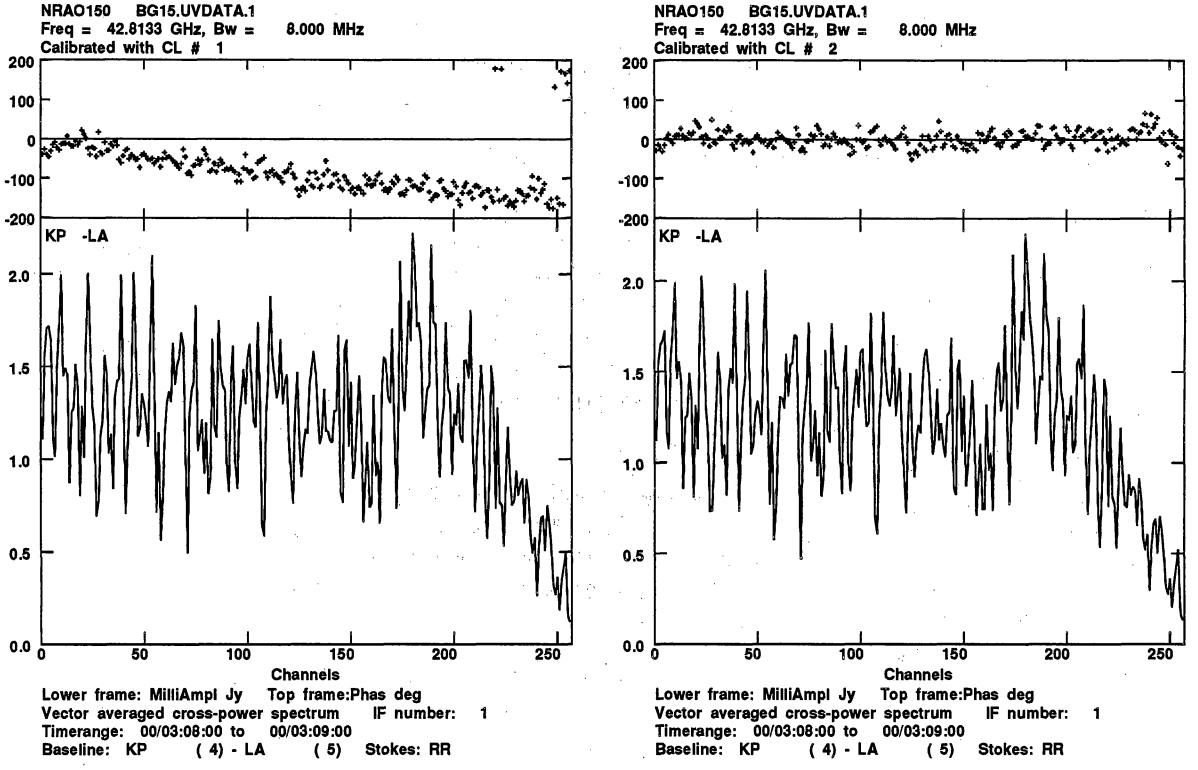
The parameter **INTERPOL** allows the user to choose between several different methods of phase interpolation. Use EXPLAIN CLCAL ⏎ to view all the options. With good quality data, the **'AMBG'** option should work well. Note, however, that this option uses the **SN** solutions immediately before and 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 **SOURCE** and **CALSOUR** to the name of the desired source. Note that it is not necessary to create a new **CL** table each time **CLCAL** is run; in the example given above, **GAINVER=3** and **GAINUSE=4** would not be changed and the **CL** solution for each source would be updated in **CL** table 4. 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 single **SN** table entry closest in time for that source is used without any interpolation. This is not as good as doing multiple runs with the **INTERPOL = 'AMBG'** option because there can potentially be jumps in phase at points equidistant from two **SN** table entries.

Other interpolation options include **INTERPOL = 'BOX'**, in which the **SN** table is smoothed with a boxcar function before being linearly interpolated onto the **CL** table. The boxcar smoothing times for delay, rate etc. are specified in parameter **INTPARM**. However it is recommended that **SN** smoothing be done prior to **CLCAL** using task **SNSMO**.

The **INTERPOL = 'POLY'** option is useful if there are bad **SN** solutions remaining in the data. This option fits a polynomial to the rate solutions and then integrates this polynomial to determine the phase corrections to be entered into the **CL** table.

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 § 9.1. 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 is shown in the plots on the next page.



left: A **POSSM** plot of the uncalibrated spectrum of NRAO150 at 43 GHZ on the baseline Kitt Peak to Los Alamos. The plot only shows the spectrum for IF 1 in order 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.

9.6.2. 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, namely while enforcing closure constraints, **BLING** makes independent fits to each baseline for delays and rates, creating a **BS** table of baseline-based solutions. Then the task **BLAPP** converts these solutions to station-based quantities and, by default, updates or generates a new **CL** table. This **CL** table can then be applied to the data using **SPLIT** as described in § 9.8. Note that, unlike using **FRING**, there is no need to run **CLCAL** to update the **CL** table. However, you can run **BLAPP** with **OPCODE 'SOLV'**, if you wish, to generate only an **SN** table which can then be interpolated using **CLCAL**.

In most cases, the global fringe fitting described in § 9.5.1 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. Amongst the advantages of the baseline based fringe fitting are:

1. **BLING** may be more robust in the presence of a complex source structure (*e.g.*, 3C84 at 8.4 GHz). While, in such cases, **FRING** requires an accurate source model, **BLING** provides relatively good solutions with a point-source default.

2. While **FRING** will only look initially for fringes to two antennas and then gives up, **BLING** searches for fringes on all baselines; then **BLAPP** finds a route through these solutions from each antenna to the reference antenna. For some combinations of source structure and *uv*-coverage, this may be a superior approach.
3. Fringe solutions can be found for cross-polarized fringes without editing the *uv* header.
4. As presently implemented, for a given number of IF and spectral channels, **BLING** can solve for longer scans than can **FRING**.
5. **BLING** has the option of adjustable, non-zero centered fringe-search windows, which can be controlled from an external file. This option will be especially important in fringe fitting orbital VLBI data.

The following suggestions should help you make the best use of the new **BLING**.

1. Run **BLING** on a machine that does not penalize double-precision arithmetic (*e.g.*, an IBM RS/6000) whenever possible; **BLING** makes heavy use of double precision during the chi-squared fit and takes a relatively large performance hit on machines where double-precision arithmetic is slower than single-precision (*e.g.*, SPARCs).
2. Apply *a priori* amplitude calibrations before running **BLING**. If you don't do this, the data weights will not reflect the expected noise in the data and **BLING** will fail spectacularly (in some cases it may crash).
3. Don't turn on the fringe acceleration search (**DPARM(7)** to **DPARM(9)**). This is a special option for space VLBI and will merely slow **BLING** down and degrade the quality of the solutions for ground-based arrays.

Further information about **BLING** and **BLAPP** is available in *AIPS* Memo 89 and the relevant explain files.

9.7 Calibrating baseline-based errors and the bandpass

In order to achieve the highest possible dynamic range, it may be necessary to calibrate the bandpass response function and/or to solve for baseline-based non-closing phase and amplitude errors. The dynamic range level at which these types of calibration are required is not yet well known for VLBA data, but is probably at the level of a the few thousand to one. For many experiments, you may skip this calibration initially and return to it later if you suspect your images are limited by these effects.

Bandpass calibration is achieved using the task **BPASS** (see § 9.11.1) using either the auto-correlation or cross-correlation data. The output is a **BP** or bandpass table. 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**. It is suggested that integration over a variable bandpass function is one of the most significant source of non-closing errors in 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 Array Memo No 697A) after applying bandpass calibration. Other dynamic range limiting effects also exist such as lack of polarization purity. 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. In order 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 C_R	to review the inputs.
> INDISK *n1* ; GETN *ctn1* C_R	to select the multi-source visibility data as the input file.
> IN2DISK *n2* ; GET2N *ctn2* C_R	to select your best image as the input model file.
> SOURCE 'DA193' C_R	to select your calibration source.
> DOCALIB 1 ; GAINUSE 4 C_R	to apply previous calibration from **CL** table 4.
> BLVER 1 C_R	to create **BL** table version 1.
> SOLINT 1440 C_R	to determine one complex gain correction per baseline for the whole observation.
> GO C_R	to run the program.

9.8. Applying calibration and time averaging

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 C_R	to review the inputs.
> INDISK *n* ; GETN *ctn* C_R	to specify the input file.
> SOURCE ' ' C_R	to write all sources.
> DOCALIB 1 ; GAINUSE 4 C_R	to apply calibration, specifying the version of **CL** table to use.
> DOBAND 1; BPVER 1 C_R	to apply **BP** table 1 if present.
> BLVER 1 C_R	to apply **BL** table 1 if present.
> APARM(1) 1 C_R	to have all spectral channels within each IF averaged.
> APARM(2) 2 C_R	to have an amplitude correction made for the correlator integration time (in seconds)
> GO C_R	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)=2** in **SPLIT** or use task **AVSPC**:

> TASK 'AVSPC' ; INP C_R	to review the inputs.
> INDISK *n* ; GETN *ctn* C_R	to specify the input file.
> OUTDISK *n* ; OUTNAM ' ', OUTCLA ' '	to specify the output file.
> AVOPTION 'AVIF' C_R	to average over IFs.
> BIF 0 ; EIF 0 C_R	to select all IF channels
> GO C_R	to run the program.

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**, or **UVPLT**, and list them with **LISTR**, **PRTUV**, or **UVPRT**. **POSSM** is now no longer useful since you have averaged your data in frequency space.

It is now convenient to average the data up to a 30–60 seconds or so using **UVAVG**, both to reduce the bulk of the data and to increase the signal-to-noise for subsequent iterations of the self-calibration/mapping 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

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, then it is only necessary to delete the latest **CL** table containing amplitude calibration and rerun **ANCAL** 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 § 9.4. 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**.

9.10. Self-calibration and imaging

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 generally preferable to **UVMAP**, **HORUS** and **APCLN** and to **MX** and **WFCLN** for VLBI data.

The main difference between the processing of VLBI and VLA data is that, initially, absolute phases of VLBI data are un-calibrated (unless phase-referencing is used). 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 the procedure **HYB** which executes a whole cycle of hybrid mapping (*i.e.*, an **MX** plus **CALIB**). It also plots images and allows editing of Clean component files prior to self-calibration. Type HELP HYB $\subset_R$ for more information.

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. Others use the new *AIPS* task **SCMAP**, while still others follow older **HYB** path. 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* which is available on the World-Wide Web. Here we make a few suggestions on how to control **CALIB** and **IMAGR**. Again, please note that this task is new in **15JUL95** *AIPS* and is preferable to all previous imaging tasks (except perhaps the revised **SCMAP**).

9.10.1. CALIB

1. Start by correcting antenna phases only, *i.e.*, use SOLMODE = 'P' $\subset_R$. 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 interpolated 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 (≤ 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 may be flagged.
2. On subsequent iterations, use the Clean image as produced by **IMAGR** as your input model. Unlike VLA images, 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 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.

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 which was used to produce that Clean model. It is probably advantageous to go back to the original data set periodically. Amongst other things, this can prevent the telescope amplitudes from "wandering" (see below).

4. If your array contains antennas which have a wide range of sensitivities, *e.g.*, the VLBA plus the phased VLA and/or the Bonn 100m, 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 baselines weight 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** 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.

7. When you are finally ready to solve for amplitude corrections, you should first apply all previous phase calibration. Then run **CALIB** for amplitude, initially setting the solution interval (**SOLINT**) to several hours. 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 C_R to constrain the mean amplitude solutions over all antennas to be one. You can also set SOLMODE='GCON' C_R 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.

9.10.2. IMAGR

1. Before using **IMAGR**, print out and read the **EXPLAIN** file. It is a powerful and complicated task 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' $\subset_R$) weighting in **IMAGR**. These images will represent the extended structure of the source better. If the highest resolution is required, try uniform (UVWTFN 'UN' $\subset_R$) 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 Bonn or the phased VLA), then it may be advantageous to lower 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 these sensitive antennas. One way of doing this is to use UVWTFN = 'UV' $\subset_R$ in **IMAGR**. This option takes the fourth root of the input weights before applying uniform weighting. Another, more 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 **IMAGR** supports circular as well as rectangular windows. Alternatively, **CCEDT** can be used to edit the Clean components after **IMAGR**, but before the next **CALIB**. **IMAGR** and **SCMAP** allow this to be done interactively at the start of each major Clean cycle including the first.
4. Use DOTV = 1 $\subset_R$ 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.

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.

9.11. Spectral-line VLBI data

This section deals with the special problems associated with calibrating and imaging spectral-line VLBI data within *AIPS*. Most of the procedures to be followed are very similar to those used in the continuum data reduction described above. This section will, therefore, describe the principal differences and will not list all the task parameters in gory detail. instead, it will emphasize areas of particular concern to spectral-line users.

Although continuum VLBI data sets normally contain spectral channels, a spectral-line database will normally contain many more channels, typically 128 or 256 per IF for VLBA, and probably 112 for Mark III. Spectral-line data from the VLBA correlator should be read using **FITLD** as described above for continuum data. **MK3IN** should be used for data from MkIII correlators (see § 9.12).

9.11.1. Calibrating spectral-line VLBI data

The data should be calibrated by following the recipe outlined below:

1. Determine the bandpass calibration. Most often this will be done using "self" (auto-correlation) spectra on the continuum calibrators. However, a full complex bandpass can be determined if the data quality is good enough.
2. Run **CVEL** to ensure that all data are at the same velocity. For observing convenience, the data were probably all taken at the same frequency. Therefore, a Doppler correction has to be performed. It is important to apply the bandpass correction at this stage since the data are given source- and time-dependent frequency shifts by **CVEL**.
3. Determine the telescope-based amplitude calibration factors. These are estimated from the self-spectra on the program source.
4. Obtain the residual delay and gross residual fringe-rate corrections from the continuum calibrators.
5. Apply these corrections to the data and determine the fine residual fringe-rate corrections from a spectral channel (or group of channels).
6. Apply all the calibration determined in the previous steps and write a fully calibrated data set.
7. Self-calibrate and image a reference feature in the spectrum.
8. Phase-reference the rest of the spectrum to the reference feature by applying the complex gains determined from the self-calibration.
9. Image the whole source.

The bandpass correction functions are determined using **BPASS**. An example of the inputs to produce bandpass spectra from the self-spectra would be:

> TASK 'BPASS' ; INP C_R	to review the inputs.
> INDISK *n1* ; GETN *ctn1* C_R	to select the multi-source visibility data as the input file.
> CALSOUR 'BLLAC','DA193' C_R	to specify the continuum source(s) which were observed for the purpose of bandpass calibration.
> BCHAN 0 ; ECHAN 0 C_R	to use the whole spectrum.
> DOCALIB -1 C_R	to not apply any calibration.
> SOLINT 0 C_R	to average data over whole scans before determining the bandpass.
> SMOOTH 1, 0, 0 C_R	to Hanning-smooth the spectrum.
> BPASSPRM 1, 0 C_R	to use the self-spectra.
> GO C_R	to run the program.

This will produce a **BP** table containing the antenna-based bandpass functions to be applied to the data. Since only the self-spectra were used, the phase response of the bandpasses is not determined. If you wish to correct for phase errors across the band, then you must first fringe fit the calibrator data (see below), then set **DOCALIB** to 1 and **BPASSPRM(1)** to 0, and run the task. However, you should check your results very carefully. The **BP** tables can be plotted with **POSSM** or printed with **PRTAB**.

The next step is to perform the Doppler correction on your data. Normally, when observing, you will have kept the frequency constant throughout the run for ease of observing and also because most modern video-converters (BBCs) can only be set to a precision of 10kHz. Therefore, although your data will have

> TIMERANG 0 22 0 0 0 22 30 0 CR — 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 CR — to pass only self-spectra.

> GO CR — 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 amplitude gain correction factors into an **SN** table.

> TASK 'ACFIT' ; INP CR — to review the inputs.

> INDISK *n* ; GETN *ctn* CR — to specify the input file.

> IN2DISK *n* ; GET2N *ctn* CR — to specify the template file.

> CALSOUR 'OH127.8 ' CR — to select the source to use for calibration.

> DOCALIB -1 CR — to avoid applying any previous calibration.

> DOBAND -1 CR — to skip the bandpass correction since it was done when **CVEL** was run.

> SOLINT *n* CR — to average the self-spectra over *n* minutes (*e.g.*, 10) before doing the least-squares fit.

> REFANT 1 CR — to select the desired reference antenna from the template file.

> BCHAN 50 ; ECHAN 70 CR — to set the range of spectral channels over which the fit is performed.

> APARM 0, 0, 50, 0, 0.72 CR — 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.

> BPARM 80, 120 CR — to set up to 5 pairs of start and stop channels to use in determining the polynomial spectral baseline to be removed from the program source. The order of the polynomial is specified in **APARM(1)**.

> CPARM 80, 120 CR — to set up to 5 pairs of start and stop channels to use in determining the polynomial spectral baseline to be removed from the template source. The order of the polynomial is specified in **APARM(2)**.

> XPARM 45.1, 48.0, 50.1, 49.5 CR — 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.

> SNVER 1 CR — to set the **SN** table into which the solutions are written.

> GO CR — 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**:

> TASK 'SNSMO' ; INP C_R	to review the inputs.
> INDISK *n* ; GETN *ctn* C_R	to specify the input file.
> SOURCE ' ' C_R	to smooth all sources in the **SN** table.
> BIF 1 ; EIF 0 C_R	to smooth all IFs.
> INTERPOL 'BOX' C_R	to smooth the data with a boxcar function.
> BPARM 0.5, 0 C_R	to smooth amplitudes (only) for 30 minutes.
> CPARM 0 C_R	to do no clipping of amplitude solutions.
> SMOTYPE 'AMPL' C_R	to smooth only amplitudes.
> SNVER 1 C_R	to choose which **SN** version is smoothed, with the smoothed values over-writing the original values.
> GO C_R	to run the program.

Then run **CLCAL** to apply them to the calibration table:

> TASK 'CLCAL' ; INP C_R	to review the inputs.
> INDISK *n* ; GETN *ctn* C_R	to specify the input file.
> SOURCE ' ' C_R	to calibrate all sources.
> CALSOUR ' ' C_R	to use solutions from all sources when applying.
> OPCODE 'CALI' C_R	to smooth and apply gain solutions will to the **CL** table.
> INTERPOL ' ' C_R	to use linear (vector) interpolation of the **SN** table with no smoothing.
> INTPARM 0 C_R	to set smoothing time; not used here.
> SNVER 1 C_R	to select the **SN** table containing solutions to be interpolated.
> GAINVER 1 C_R	to select the **CL** version to which solutions are to be applied.
> GAINUSE 2 C_R	to select the output **CL** version, containing updated calibration information.
> GO C_R	to run the program.

9.11.3. Residual delay and fringe-rate calibration

The determination of the delay and fringe-rate calibration is a two-step process for spectral line VLBI data. The residual delay cannot be estimated from the line source itself because, due to the very nature of the source, the delay is a rapidly varying function of frequency. Therefore, we must estimate first the residual delay and fringe rate for each antenna from the continuum calibrators observed for this purpose.

Due to the large numbers of baselines and spectral channels now available in VLBA observations, you are often restricted to solution intervals which are unnecessarily short for continuum calibrators. This can be overcome be 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. One then performs the fringe fitting described below, with a much more reasonable solution interval, and runs **TACOP** to copy the **SN** table back to the original data file.

> TASK 'FRING' ; INP C_R	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 amplitude calibration.
> GAINUSE 2	to do this from **CL** table 2.
> 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 1 C_R	to select a reference antenna (see continuum discussion).
> SOLINT 6 C_R	to set the solution interval in minutes. It may be greater than 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 SNR acceptable. *Warning:* data with lower SNR will be flagged as bad.
> 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 250kHz-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 2 C_R	to write solutions in a new **SN** table.
> GO C_R	to do the fit.

Use **TACOP** to copy the **SN** table back to the line data set and then run **CLCAL** in order to apply the delay and fringe-rate solutions to all sources, as is described in the § 9.6.1.

The second step in this process is to correct the spectral-line data for any further residual fringe rates. The fringe rates determined from the continuum calibrator may not be those applicable to the line source since the line source may well be in a different part of the sky and the antennas will be looking through a different atmosphere. It is also likely that the position of the line source will not be known accurately, thereby introducing an additional residual fringe rate. You should select a suitable line channel (or group of channels) to use for this fringe-fitting step by examining the spectra produced with **POSSM**. Then rerun **FRING**:

> 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 3	to use **CL** table 3, the latest one generated.
> FLAGVER 0 C_R	to apply the most recent flag table.
> SNVER 3 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 1 CR	to use the same reference antenna as in the previous run of **CALIB**.
> SOLINT 6 CR	to set the solution interval in minutes. It may be greater than the coherence time.
> APARM(6) 1 CR	to get useful, but limited printout.
> APARM(7) 9 CR	to avoid false detections by setting the minimum SNR acceptable. *Warning:* data with lower SNR will be flagged as bad.
> DPARM(1) 1 CR	to use one-baseline combination in initial, coarse fringe search (FFT).
> DPARM(2) -1 CR	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 CR	to search for fringes over the full Nyquist fringe-rate window since we don't know where the fringes are.
> DPARM(4)= 1 CR	to tell **FRING** what the correlator integration time was.
> DPARM(5) 0 CR	to do the least-squares solution.
> GO CR	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. These should be applied to the data with **SPLIT**, writing single-source files (see § 9.8). Then what remains to be done is the self-calibration step. This is performed in the same manner as for continuum data, although the complex gain solutions are determined using just one spectral channel and then applied to all channels before making the image cube. See § 9.2 for more information on this procedure.

9.12 Processing data from a MkIII correlator

9.12.1 Loading the data

Data from a MkIII correlator, such as that in Bonn, Germany or Haystack, Massachusetts, can also be read into *AIPS*. In order 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 data into *AIPS* and preparing it for further processing is much more cumbersome than the equivalent process for VLBA data This just reflects the manner in which data are generated on a baseline-based correlator with a limited number of playback drives.

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 **A***tttt*. 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 more **'ANY'** entries than stations in the data set or these extra stations will not be loaded. The parameters in this text file are:

STATIONS='NRAO','VLA','OVRO','FDVS','MPI'	station names.
NO_POL=2	the number of polarization correlations (*e.g.*, RR, LL, RL and LR), the default is 1.
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.
/	**keyin** style delimiter.

Then, from inside **AIPS**, mount the tape (§ 3.9) and run **MK3IN**:

> TASK 'MK3IN' ; INP C_R	to review the inputs.
> INFILE 'MYVLB:PARAM.LIS' C_R	to define the text control file.
> IN2FILE 'MYVLB:AFILE.LIS C_R	to point to a file containing a list of scans to be loaded as produced by **AFILE**
> INTAPE 4 C_R	to specify the tape drive number.
> OUTNA 'EXP 86-34' C_R	to select the output file name.
> OUTCL 'MK3IN' C_R	to select the default output class name.
> REFDATE '12/11/89' C_R	to tell **MK3IN** the start date of the observations — get this right or you may get negative times.
> DOUVCOMP 1 C_R	to write data on disk in compressed format.
> APARM 1, 0 C_R	to set the time increment in the **CL** table entries in minutes (to 1 here).

> 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.

9.12.2. Sorting, concatenating and merging *uv* data files

The *AIPS* data files created by **MK3IN** will be in an arbitrary sort order. Use **UVSRT** 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 name.
> SORT 'TB' ↵	to sort to time-baseline order.
> GO ↵	to make the sorted *uv* file.

If you did not set **DOCONCAT=1** when running **MK3IN** and several files were loaded from tape for one observation, use **DBCON** to concatenate them together. Both of the input files must have the same reference day number and have identical antenna numbers. That is, the antennas extension (**AN**) files with each input *uv* data file must be the same. 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 name.
> DOARRAY 1 ↵	to force **DBCON** to mark the output data records as being in the same sub-array. Both of the input files must have the same reference day and have identical antennas files.
> GO ↵	to concatenate the two files.

MkIII VLBI correlators usually produce redundantly correlated data. Merge your **UVSRT** or **DBCON** output 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 name.
> 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 ↵	to set the input table version.
> OUTVER 1 ↵	to set the output table version to the input.
> 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.

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**:

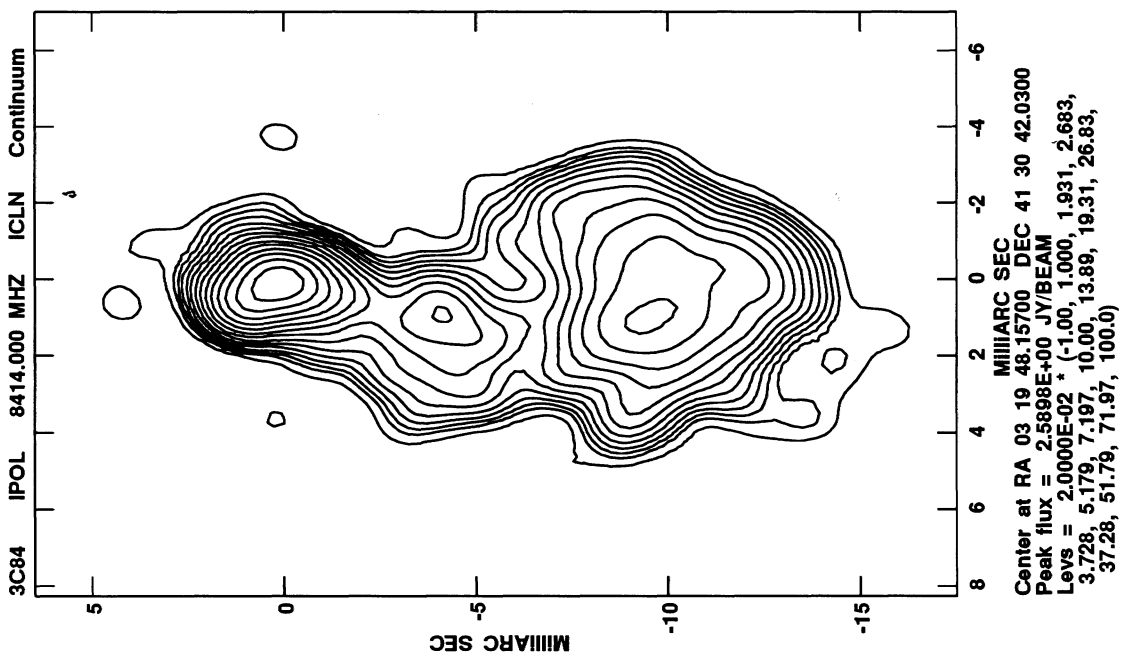
> TASK 'SBCOR' ; INP ↵	to review the inputs.
> INDISK *n1* ; GETN *ctn* ↵	to specify the input file.
> OUTNA INNA ; OUTCL 'SBCOR' ↵	to specify the output file name.
> BCHAN 1 ; INP ↵	to specify the lowest channel of lower sideband.
> ECHAN 4 ; INP ↵	to specify the highest channel of lower sideband.
> APARM(1) 0 ; INP ↵	to apply the default phase offset (*i.e.*, −130°.)
> ANTENNAS *n1* , *n2* , *n3* ,... ; INP ↵	to specify the VLBA antenna numbers; use **PRTAN** to identify VLBA antennas.
> GO ↵	to run the program.

Once your data are all in one file, all redundant correlations and **CL** table entries have been removed, and any sideband corrections applied, you should 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 $\leq$ 10-minute time gaps within scans, to limit scans to $\leq$ 30 minutes, and to not create a new **CL** table.
> GO ↵	to run the program.

Now return to §§ 9.2 through 9.11 to process your data.

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.
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.
TY	System temperatures 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.
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.



The first scientific results reported from the VLBA correlator in the continuum. We thank R. C. Walker for these data. More details are available via the World-Wide Web.

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 which are consistent with the inputs to **ALLDEST**.

And to clear all your messages and compress your 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 SYSTEM MANAGER. Old data and messages belonging to any user (including you) may be deleted by the verb **TIMDEST**. The definition of "old" is set by your local *AIPS* Manager, who must be consulted about the rules for invoking **TIMDEST**.

A list of the software associated with deleting various files can be obtained at your terminal by typing:

> HELP DELETE ↵

This listing is also given in § 15.

12.3. Sending comments to the *AIPS* programmers

Comments, suggestions and bug reports about any facet of *AIPS* can be entered into a file by typing:

> GRIPE ↵

while in **AIPS**. Simply follow the directions and record your comment. On a weekly basis, "gripes" are collected from all NRAO computers and an acknowledgement is sent to each individual "griper." More detailed replies to the gripes are generated by the *AIPS* programming staff and are mailed to each griper later. Normally, a full reply will be given at the end of the current *AIPS* release cycle (*i.e.*, quarterly), but significant modifications and additions to *AIPS*, if warranted in response to gripes, may take longer to implement. In such cases, follow-up replies are also generated and sent to the gripers. The gripes and responses will become public (at NRAO), so write them clearly and in good taste. Complete copies of the gripes and responses are kept in binders in the *AIPS* Caiges at Charlottesville and the Array Operations Centre.

Gripes can be registered outside of **AIPS** at monitor level by typing (in VAX/VMS when logged in as **AIPS**):

$GRIPE *version* ↵ — to specify *version* **OLD**, **NEW** or **TST**.

and following directions. The program invoked by this procedure, **GRIPR**, implements the same code as **AIPS**, but its control structure is much simpler. Try:

> HELP ↵

after the GRIPE for some information about the program's capabilities. The command to invoke this program may be different at some installations, particularly non-VAX/VMS ones.

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 *AIPS* programmers' enthusiasm.

Gripes may also be e-mailed to the *AIPS* group at **aipsmail@nrao.edu** or **aipsmail@nrao.bitnet** or **6654::aipsmail** or . . .**!uunet!nrao1!aipsmail**.

12.4. Exiting

To exit from `AIPS` type:

> EXIT ↵

Please clean up any papers, tapes, etc. in the area around your terminal before you go.

12.5.1. Bananes rôties

1. Preheat oven to 375°.
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.

12.5.2. 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.

13. PANIC SECTION

D O N ' T P A N I C ! ! ! !

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 Appendix Z for the details appropriate to your particular computer system. Type HELP PANIC ↵ to review recent information on bugs, disasters and misfortunes within *AIPS* that have known solutions. Some well-known possibilities follow.

13.1. My printout is semi-infinite!!

To abort a job which is currently printing out, do the following:

1. Turn off power to the printer to conserve paper.
2. Type SPY ↵ to see if the offending program is still executing. If so, ABORT *taskname* ↵ will kill it.
3. Get into monitor mode (prompt **$** on VAXes; prompt % on UNIX machines) either on your terminal or another one.
4. Abort the current print job. See Appendix Z for the details for your machine (*e.g.*, § Z.1.7.1).
5. Turn the power back on for the printer.

13.2. The TEK screen will not print hard copy!!

1. Find the device which makes the hard copies. Is it turned on?
2. Check to see if there is paper in the device.
3. Power it down, then power it up again. Try the **HARD COPY** button on the TEK again.
4. If it still fails, call for hardware repair.

13.3. My data catalog has vanished!!

1. Are you connected to the right *AIPS* computer, if your site has more than one?
2. Are you logged in to the correct *AIPS* version? NRAO *AIPS* systems, in particular, may have separate data areas for the **TST** and **NEW / OLD** releases.
3. Have you set **INDISK** *et al.* correctly before running **CAT**? Type INP CAT ↵ to check. Is **USERID** not set to 0 or your user number?
4. Is a non-system disk mounted? Type FREE ↵ or exit from **AIPS**, and have the operating system tell you which disks are currently running. See Appendix Z for details.

```
> OUTNAME '3C138' CR
> OUTSEQ 5 CR
> GO PRTIM CR
```

to be shortened to:

```
> Inn '3c138' ; INC 'imap' ; 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.)

14.2. 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), your user identification number, 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 9999.

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.

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 such a file and it contains directory information for all images and *uv* data sets for all users on the disk pack. On most systems, each user is assigned their own catalog file on each disk.

Extension files may be associated with any image data file. Each image can have (in principle) up to ten types of extension files and up to 255 "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 values. 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^{th} (first) coordinate increments fastest and is associated with a column in each plane of the image. The j^{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 subarray in up to seven dimensions. Whenever a sub-image is taken from an image, the pixel designation of any image element will usually change.

14.3. AIPS language

AIPS contains a basic set of symbols and keywords which are needed for facility in the language, as well as the symbols needed by the application code. A list of the basic symbols is given in § 15 and may be listed on your terminal by typing:

```
> HELP POPSYM ↵
```

Here are some simple examples of uses of the **AIPS** language:

`> TYPE (2 + 5 * 6) ↵`	32 is written on the terminal.
`> TYPE 'X =', ATAN (1.0) ↵`	`X = 45` is written on the terminal.
`> TYPE 'MAPNAME ', INNAME, INCLASS, INSEQ ↵`	`MAPNAME 3C138 IMAP 1` is written on the terminal.

14.4. Processing loops

The do-loop capability in **AIPS** uses the pseudoverbs **FOR**, **TO**, and **BY** for repetitive operations. Such loops are primarily intended for use in "procedures" (see § 14.5). If a 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 'IMAP' ↵`	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 ↵
```

14.5. 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 procedures in **RUN** files (§ 14.6) 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:

with a convenient set of tasks, such as loading an image, sorting the data, and making and cleaning the image. All facilities in **AIPS** such as **GET**, **SAVE**, and **TGET** can be used in **RUN** files.

14.7. 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 § 14.12 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 ; SUM = 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 ↵                                    to mark end of FOR loop.
:  AVG = SUM / N ↵                               to get average value.
:  IF N > 1.5 THEN ↵                             to check if N > 1.
:       RMS = SQRT((SUM2 - N*AVG*AVG) / (N * (N-1))) ↵   to calculate rms if N > 1.
:       END ↵                                    to mark end of IF clause.
:  TYPE 'AVG=',AVG,'RMS=',RMS,'AT PIXEL',PIXXY ↵
```

```
: 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.

14.8. Character handling in *POPS*

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.

POPS provides several string handling expressions. These are:

POPS symbol	use	operation
!!	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

14.9. 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

```
< INSEQ 1 ; OUTN INN ; OUTCL INCL CR
< OUTSEQ = 0 ; SORT = 'XY' CR
< GO CR                          (Batch AIPS always waits for a task to finish before continuing.)
< RUN XXXXX CR                   (RUN files are good to use in a batch job.)
< GO CR
< ENDBATCH CR                    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 CR
```

To edit line *n* in a batch file, type:

```
> BATEDIT n CR
< put text here CR               to replace old line n.
< some more text CR              to insert more commands between old lines n and n+1.
< ENDBATCH CR                    (spelled out in full.)
>                                (Resume normal interactive processing.)
```

As with procedures, 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 CR
```

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 hexadecimal number < F) 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**.

To submit a batch job from outside the **AIPS** interactive program, log in to the **AIPS** area and then type:

```
$ BATER version CR               to specify version OLD, NEW or TST.
```

and generate the batch work file as shown above. It can be submitted without ever going into **AIPS**. Note that a different version of this command syntax may be required on your non-VAX/VMS computer.

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, however, that **SAVE** / **GET** files may become obsolete with new *AIPS* releases.

At present, batch jobs are run after a short delay, on a first-come, first-served basis. However, batch jobs which use an array processor are forbidden in batch queue number 1 (if there is > 1 batch queue) and may be excluded at some times of day by your *AIPS* Manager in the other queues. After your job has been submitted successfully, type:

```
> QUEUES CR                      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. 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.

14.11. Remote use of *AIPS*

14.11.1. Starting up remote AIPS

AIPS has a built-in service for remote users who will forgo the use of a TV. To request the remote user mode, log in to the *AIPS* account on your computer, and start up **AIPS** with one of AIPS TST REMOTE $^{C}_{R}$, AIPS NEW REMOTE $^{C}_{R}$ or AIPS OLD REMOTE $^{C}_{R}$. *AIPS* will then skip the TV selection step and will assume that your terminal supports a Tektronix 4010 emulator.

14.11.2. Remote printout: OUTPRINT

AIPS does not support remote printers explicitly. You have two options for capturing *AIPS* printer output, however. One is to route it to disk on the *AIPS* machine using the **OUTPRINT** parameter of an *AIPS* task or verb, and later copy the disk file to your own location. The other, which is less satisfactory but does not require you to exit from *AIPS*, is to use the screen capture mode of your terminal or communications program to record *AIPS* messages or "printout" with **DOCRT** set to 1. For a current list of *AIPS* verbs and tasks that support the **OUTPRINT** feature at your *AIPS* installation, type HELP OUTPRINT /CR.

14.11.3. Remote graphics: TKPL and TXPL

If you have started up **AIPS** in **REMOTE** mode as described in § 14.11.1, the graphics output from **TKPL** will be routed to your terminal as if it is a stand-alone Tektronix graphics device. Do not use any **TK...** task in *AIPS* unless your terminal can diagnose the onset of Tektronix graphics instructions and switch to graphics mode appropriately!

Many terminal and communications packages that claim to have a Tektronix 4010 emulation mode do not implement a strict emulation. Because of this, you may find that **TKPL** graphics appear correctly on your terminal but are then over-written by *AIPS* messages. This is unfortunate, but in the absence of a firm emulation standard, it is difficult or impossible to write generic *AIPS* code that will prevent it.

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 **UVPL** (to guide self-calibration or to diagnose interference) or examining calibration solution plots from **SNPLT** or **GNPLT**. **TXPL** can also usefully interpret simple contour plots or even grey scales(!) for a remote user. It is always 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).

The *AIPS* task **QMSPL** also has an option to write its output to a disk file using the **OUTFILE** parameter. Remote users with wide-bandwidth links may find it practical to transfer these disk files (which can be very large for complicated plots) to their local QMS system for plotting.

14.11.4. Remote data entry and extraction

Remote users should be aware of the FITS disk capabilities of the *AIPS* tasks **FITTP**, **UVLOD** and **IMLOD**. These provide ways to get data out of, and into, remote *AIPS* systems without using tape drives. **FITTP** can write both images and *uv* data to non-*AIPS* disk files in FITS format using its **OUTFILE** parameter. **UVLOD** can read FITS disk *uv* data files from outside *AIPS* and **IMLOD** can read FITS disk image files from outside *AIPS*, using the **INFILE** parameter. The details of both approaches are somewhat system-specific, but see Appendix Z.1.8.1 of this *COOKBOOK* for an illustrative example from the NRAO Convex systems. At present, remote-to-local and local-to-remote FITS disk file transfers must be done outside *AIPS* with standard network software such as ftp.

14.12. Adding your own tasks to *AIPS*

This Section is a brief guide for the average 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.

14.12.1. What kinds of task can I write?

The simplest way to write an *AIPS* task is to modify one of the four template tasks, **TAFFY**, **CANDY**, **FUDGE** 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 § 14.12.5.

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 any 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* data base, modifies this data base point by point, and then creates and writes this modified output *uv* data base 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.

UVFIL - This task is similar to **FUDGE** except that the input *uv* data may be 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 best to start with **COMB** itself than with one

of the templates. However, more than trivial changes to major tasks will require a careful look at the code, which is generally well documented and segmented.

Changes to tasks that use the TV and other graphics devices, imaging and deconvolution software should not be attempted unless the changes are almost trivial. It is not advisable to modify verbs, because the entire **AIPS** program must be relinked when this is done. This should not be attempted unless the local *AIPS* manager agrees with the changes.

14.12.2. How do I get started?

Unless you have certain privileges in *AIPS*, it may be more convenient to generate and link new code in your own user directory rather than in your installation's designated local *AIPS* directory. Check with your local *AIPS* Manager about this. Decide which template task or other *AIPS* task you want to modify, then follow these instructions after logging into your private area.

Under VMS in a VAX:

Set up logical assignments with

```
$ @DISK:[AIPS]LOGIN.PRG CR          You may have to specify the DISK
```

Copy compilation and linking libraries with

```
$ COPY SYSVMS:QYPGNOTOPT.OPT LOCAL.OPT CR
```

Find out which *AIPS* subdirectory the target task is in with

```
$ DIR APLPGM:*.FOR CR
```

or

```
$ DIR APGNOT:*.FOR CR
```

Copy the original code to your area and rename it to NTASK.FOR:

```
$ COPY APLPGM:TASK.FOR NTASK.FOR CR
$ COPY APGNOT:TASK.FOR NTASK.FOR CR
```

as appropriate and where NTASK is your new task name

```
$ COPY HLPFIL:TASK.HLP NTASK.HLP CR  Copy INPUTS/HELP file for task.
```

Under UNIX in a CONVEX:

```
% $CDTST (or $CDNEW) CR              Set up logical assignments.
% LIBS $APGNOT > APGNOT.OPT CR       Copy compilation and linking libraries.
% LIBS $APLPGM > APLPGM.OPT CR
% ls $APLPGM/*.FOR CR
```

or

```
% ls $APGNOT/*.FOR CR                To find where your task is located.
% cp $APLPGM/TASK.FOR NTASK.FOR CR   Copy code for TASK into your area and rename.
% cp $APGNOT/TASK.FOR NTASK.FOR CR   Alternate for above.
% cp $HLPFIL/TASK.HLP NTASK.HLP CR   Copy INPUTS and HELP file for task.
% setenv MYAIPS 'pwd' CR             Define MYAIPS (must be uppercase) as your version name for
                                     AIPS. This must be done in UNIX systems in order to avoid
                                     AIPS upper/lower case problems.
```

15. 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 **15JUL94** 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.

15.1. 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
```

ANTENNAS	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
ASPMM	Plot scaling parameter - arc seconds per millimeter on plot
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
BASELINE	specifies which antenna pairs are to be selected/deselected
BATFLINE	specifies starting line in a batch work file
BATQUE	specifies the desired batch queue
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 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
BPA	gives position angle of major axis of beam or component
BPARM	general numeric array adverb used too many places
BPVER	specifies the version of the bandpass table to be applied
CALCODE	specifies the type of calibrator to be selected
CALSOUR	specifies source names to be included in calibration
CCBOX	specifies pixel coordinates of subarrays of an image
CELLSIZE	gives the pixel size in physical coordinates
CHANSEL	Array of start, stop, increment channel numbers to average
CHINC	the increment between selected channels
CLCORPRM	Parameter adverb array for task CLCOR
CLEV	Contour level multiplier in physical units
CLINT	CL table entry interval
CODETYPE	specifies the desired operation type
COLORS	specifies the desired TV colors
COMMENT	64-character comment string
CPARM	general numeric array adverb used many places
CTYPE	specifies type of component
CUTOFF	specifies a limit below or above which the operation ends
DDISK	Deterimins where input DDT data is found
DDTSIZE	Deterimins which type of DDT is RUN.
DECSHIFT	gives Y-coordinate shift of an image center from reference
DENSITY	gives the desired tape density
DETIME	specifies a time interval for an operation (destroy, batch)
DIST	gives a distance - PROFL uses as distance to observer
DOALIGN	specifies how two or more images are aligned in computations
DOALL	specifies if an operation is done once or for all matching

MINPATCH	specifies the minimum size allowed for the center of the beam
NAXIS	Axis number
NBOXES	Number of boxes
NCCBOX	Number of clean component boxes
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
NITER	The number of iterations of a procedure
NMAPS	Number of maps (images) in an operation
NPOINTS	General adverb giving the number of something
OBJECT	The name of an object
OFFSET	General adverb, the offset of something.
OFMFILE	specifies the name of a text file containing OFM values
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.
OUTCLASS	The class of an output file
OUTDISK	The disk number of an output file.
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.
OUTVERS	The output version number of an table or extension file.
PCUT	Cutoff in polarized intensity
PHAT	Prussian hat size
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.
PLEV	Percentage of peak to use for contour levels
PLVER	specifies the version number of a PL extension file
PMODEL	Polarization model parameters
POLPLOT	specifies the desired polarization ratio before plotting.
PRIORITY	Limits prioroty of messages printed
PRNUMBER	POPS number of messages
PRSTART	First record number in a print operation
PRTASK	Task name selected for printed information
PRTIME	Time limit
PRTLEV	Specified the amount of information requested.
QUAL	Source qualifier
RASHIFT	Shift in RA
REASON	The reason for an operation
REFANT	Reference antenna
REMHOST	gives the name of another computer which will provide service
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

TVXY	Pixel position on the TV screen
TXINC	TV X coordinate increment
TYINC	TV Y coordinate increment
TZINC	TV Z coordinate increment
USERID	User number
UVBOX	width of the smoothing box used for uniform weighting
UVCOPPRM	Parameter adverb array for task UVCOP
UVFIXPRM	Parameter adverb array for task UVFIX
UVRANGE	Specify range of projected baselines
UVTAPER	Widths in U and V of gaussian weighting taper function
UVWTFN	Specify weighting function, Uniform or Natural
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
WTUV	Specifies the weight to use for UV data outside UVRANGE
XINC	increment associated with an array of numbers
XPARM	General adverb for up to 10 parameters, may refer to X coord
XTYPE	Specify type of process, often the X axis type of an image
XYRATIO	Ratio of X to Y units per pixel
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

15.2. ANALYSIS

List of verbs, adverbs, tasks in category ANALYSIS

AHIST	Task to convert image intensities by adaptive histogram
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
COMB	combines two images by a variety of mathematical methods
CTYPE	specifies type of component
DOALIGN	specifies how two or more images are aligned in computations
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
DOWIDTH	selects solution for widths of model components
ECOUNT	give the highest count or iteration for some process
FLUX	gives a total intensity value for image/component or to limit
GEOM	regrids images with rotation, shift using interpolation
GMAX	specifies peak values of model components
GPOS	specifies pixel positions of model components
GWIDTH	gives widths of model components
HGEOM	interpolates image to different gridding and/or geometry
HOLGR	Read and process Holography visibility data
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 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
LGEOM	regrids images with rotation, shift using interpolation
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
MINPATCH	specifies the minimum size allowed for the center of the beam
MOMFT	calculates images of moments of a sub-image
MOMNT	calculates images of moments along x-axis (vel, freq, ch)
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
PBCOR	Task to apply the primary beam correction
PRTIM	prints image intensities from an MA catalog entry
QIMVAL	Verb to determine pixel value at specified position
RM	Task to calculate rotation measure and magnetic field
SAD	fits gaussians to portions of an image
SET1DG	Verb to set 1D gaussian fitting initial guesses.
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
STFUN	Task to calculate a structure function image
SUMSQ	Task to sum the squared pixel values of overlapping,
TABGET	returns table entry for specified row, column and subscript.
TABPUT	replaces table entry for specified row, column and subscript.
TK1SET	Verb to reset 1D gaussian fitting initial guess.
TKAGUESS	Verb to re-plot 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
TVBLINK	Verb which blinks 2 TV planes, can do enhancement also
TVCUBE	Verb to load a cube into tv channel(s) & run a movie
TVMAXFIT	displays fit pixel positions and intensity at maxima on TV
UVADC	Fourier transforms and corrects a model and adds to uv data.
UVBOX	width of the smoothing box used for uniform weighting
UVFIT	Fits source models to uv data.
UVMOD	Modify UV database by adding a model or models
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)
XGAUS	Fits 1-dimensional Gaussians to images
XMOM	Fits one-dimensional moments to each row of an image

15.3. AP

List of verbs, adverbs, tasks in category AP

APCLN	Deconvolves images with CLEAN algorithm
APGS	deconvolves image with Gerchberg-Saxton algorithm
APVC	Deconvolves images with van Cittert algorithm
ASCAL	Computes antenna-based gains based on source model (self-cal)
BLING	fringe fit residual rate and delay on individual baselines
BPASS	computes spectral bandpass correction table
CALIB	determines antenna calibration: complex gain
COMAP	Procedure to MAP and Self-Calibrate a UVDATA set
COMAP_NA	Procedure to MAP and Self-Calibrate a UVDATA set
COMAP_UV	Procedure to MAP and Self-Calibrate a UVDATA set
CONVL	convolves an image with a gaussian or another image
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
GRIDR	makes an image from single-dish data
GUARD	portion of UV plane to receive no data in gridding
HORUS	makes images from unsorted UV data, applying any calibration
IM2UV	converts an image to a visibility data set
MAPIT	Procedure to MAP and Self-Calibrate a UVDATA set
MAPIT_NA	Procedure to MAP and Self-Calibrate a UVDATA set
MAPIT_UV	Procedure to MAP and Self-Calibrate a UVDATA set
MAXPIXEL	maximum pixels searched for components in Clark CLEAN
MX	makes images and deconvolves using UV data directly.
NOBAT	Task to lock lower priority users out of the AP
OHGEO	Geometric interpolation with correction for 3-D effects
RSTOR	Restores a CC file to a map with a gaussian beam.
SCMAP	Imaging plus self calibration loop
UVADC	Fourier transforms and corrects a model and adds to uv data.
UVMAP	makes images from calibrated UV data.
WFCLN	Wide field and/or widefrequency CLEANing/imaging task.

15.4. ASTROMET

List of verbs, adverbs, tasks in category ASTROMETRY

HF2SV	convert HF tables from FRING/MBDLY to form used by SOLVE

15.5. 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.
************************************************************
List of verbs, adverbs, tasks in category BATCH

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
UNQUE      remove a given job from the job queue
```

15.7. CATALOG

List of verbs, adverbs, tasks in category CATALOG

ABACKUP	VMS procedure to back up data on tape
ADDBEAM	Inserts clean beam parameters in image header
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
CATALOG	list one or more entries in the user's data directory
CELGAL	switches header between celestial and galactic coordinates
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
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
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
DOOUTPUT	selects whether output image or whatever is saved / discarded
EGETNAME	fills in input name adverbs by catalog slot number, w error
EPOSWTCH	Switches between B1950 and J2000 coordinates in header
EXTDEST	deletes one or more extension files
EXTLIST	lists detailed information about contents of extension files
GEOM	regrids images with rotation, shift using interpolation
GET2NAME	fills 2nd input image name parameters by catalog slot number
GET3NAME	fills 3rd 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
HGEOM	interpolates image to different gridding and/or geometry
HINOTE	adds user-generated lines to the history extension file
HITEXT	writes lines from history extension file to text file
IMHEADER	displays the image header contents to terminal, message file
IMPOS	displays celestial coordinates selected by the TV cursor
IMVAL	returns image intensity 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
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
MCAT	displays images in the user's catalog directory
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 table or extension file.
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.
QHEADER	Verb to summarize the image header: positions at center
QUAL	Source qualifier
REASON	The reason for an operation
RECAT	Verb to compress the entries in a catalog file
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
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
UCAT	list a user's UV and scratch files on one or more data areas
USERID	User number
ZAP	Delete a catalog entry and its extension files

15.8. COORDINA

List of verbs, adverbs, tasks in category COORDINATES

ALTDEF	Sets frequency vs velocity relationship into image header
ALTSWTCH	Switches between frequency and velocity in image header
EPOSWTCH	Switches between B1950 and J2000 coordinates in header
NAXIS	Axis number
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
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
XTYPE	Specify type of process, often the X axis type of an image
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

15.9. EDITING

List of verbs, adverbs, tasks in category EDITING

IBLED	Interactive BaseLine based visibility EDitor
SPFLG	interactive flagging of UV data in channel-TB using the TV
TABED	Task to edit tables
TABEX	Edit Ancillary tables on UV or Image data
TAFLG	Flags data in a Table extension file
TVFLG	interactive flagging of UV data using the TV
UVFLG	Flags UV-data

15.10. EXT-APPL

List of verbs, adverbs, tasks in category EXT-APPL

GETTHEAD	returns keyword and other values value from a table header
PUTTHEAD	inserts a given value into a table keyword/value pair
TABGET	returns table entry for specified row, column and subscript.
TABPUT	replaces table entry for specified row, column and subscript.

15.11. FITS

List of verbs, adverbs, tasks in category FITS

FITLD	reads tape to load FITS images or FITS UV files to disk
HIEND	First record number in a print operation
HISTART	First record number in a print operation
TCOPY	Tape to tape copy with some disk FITS support

MDISK	Deterimins where input DDT data is 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 procedure
NPOINTS	General adverb giving the number of something
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.
OUTFILE	specifies name of output disk file, not in regular catalog
OUTVERS	The output version number of an table or extension file.
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
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
PRTMSG	prints selected contents of the user's message file
QUAL	Source qualifier
REASON	The reason for an operation
ROTATE	Specifies a rotation
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
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
TCODE	Deterimins which type of DDT is RUN.
TDISK	Deterimins where output DDT data is placed
TELL	Send parameters to tasks that know to read them on the fly
TIMERANG	Specifies a timerange
TMASK	Deterimins which tasks are executed when a DDT is RUN.
TMODE	Deterimins which input is used when a DDT is RUN.
TNAMF	Deterimins which files are input to DDT.
VLAC	verifies correctness of continuum calibration software
VLACSAVE	verifies correctness of continuum calibration

MAPIT_UV	Procedure to MAP and Self-Calibrate a UVDATA set
MAXPIXEL	maximum pixels searched for components in Clark CLEAN
MODVF	task to create a warped velocity field
MWFLT	applies linear & non-linear filters to images
MX	makes images and deconvolves using UV data directly.
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
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 pattern.
PBCOR	Task to apply the primary beam correction
PGEOM	Task to transform an image into polar coordinates.
PHASE	Baseline Phase coherence measurement
PHAT	Prussian hat size
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.
PRTCC	prints the contents of a Clean Components extension file.
PUTVALUE	Verb to store a pixel value at specified position
REGRD	Regrids an image from one co-ordinate frame to another
REMAG	Task to replace magic blanks with a user specified value
RSTOR	Restores a CC file to a map with a gaussian beam.
SCMAP	Imaging plus self calibration loop
SDCLN	deconvolves image by Clark and then "SDI" cleaning methods
SHADW	Generates the "shadowed" representation of an image
SHIFT	specifies a position shift
SKEW	Specifies a skew angle
SMODEL	Source model
SNCUT	Specifies minimum signal-to-noise ratio
SPECR	Spectral regridding task for UV data
STARPOS	Text file name
STEER	Task which deconvolves the David Steer way.
STESS	Task which finds sensitivity in mosaicing
STFACTOR	scales star display or SDI CLEANing process
SUBIM	Task to select a subimage from upto a 7-dim. image
SUMIM	Task to sum overlapping, sequentially-numbered images
TKBOX	Procedure to set a Clean box 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
UBAVG	Baseline dependent time averaging of uv data
UTESS	deconvolves images by maximizing emptiness
UVIMG	Grid UV data into an "image"
UVMAP	makes images from calibrated UV data.
UVSUB	Subtracts/divides a model from/into a uv data base
VTESS	Deconvolves sets of images by the Maximum Entropy Method
WFCLN	Wide field and/or widefrequency CLEANing/imaging task.

15.22. OOP

List of verbs, adverbs, tasks in category OOP

CCEDT	Select CC components in BOXes and above mininum flux.
CCSEL	Select signifigant CC components
FRCAL	Faraday rotation self calibration task
MAPBM	Map VLA beam polarization
MBDLY	Fits multiband delays from IF phases, updates SN table
MULIF	Change number of IFs in output
OHGEO	Geometric interpolation with correction for 3-D effects
PASTE	Pastes a selected subimage of one image into another.
UV2MS	Append single source file to multisource file.
VLABP	VLA antenna beam polarization correction for snapshot images
WFCLN	Wide field and/or widefrequency CLEANing/imaging task.

15.23. OPTICAL

List of verbs, adverbs, tasks in category OPTICAL

IMFLT	fits and removes a background intensity plane from an image

15.24. PARAFORM

List of verbs, adverbs, tasks in category PARAFORM

FUDGE	modifies UV data with user's algorithm: paraform task
NEWTASK	Information about installing a new 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

15.25. PLOT

List of verbs, adverbs, tasks in category PLOT

ASPMM	Plot scaling parameter - arc seconds per millimeter on plot
BDROP	gives number of pooints dropped at the beginning
CANPL	translates a plot file to a Canon printer/plotter
CCNTR	generate a contour plot file from an image
CLEV	Contour level multiplier in physical units
CLPLT	plots closure phase and model from CC file
CNTR	generate a contour plot file or TV plot from an image
DFTPL	plots DFT of a UV data set at arbitrary point versus time
DIST	gives a distance - PROFL uses as distance to observer
DOALIGN	specifies how two or more images are aligned in computations
DOCELL	selects units of cells over angular unit
DOCENTER	selects a single, centered page or multiple pages of plots
DOCIRCLE	select a "circular" display (i.e. trace coordinates, ...)
DOCONT	selects a display of contour lines
DOCRT	selects printer display or CRT display (giving width)

DOHIST	selects a histogram display
DOHMS	selects sexagesimal (hours-mins-secs) display format
DOMODEL	selects display of model function
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
EXTLIST	lists detailed information about contents of extension files
FACTOR	scales some display or CLEANing process
FRPLT	Task to plot fringe rate spectra
FUNCTYPE	specifies type of intensity transfer function
GAPLT	plots GAin table (ASCAL) by antenna, several per page
GNPLT	plots extrema of gain solutions from ASCAL
GREYS	plots images as contours over multi-level grey
GSTAR	Task to read a Guide Star (UK) table and create an ST table.
ICUT	specifies a cutoff level in units of the image
IMEAN	displays the mean & extrema and plots histogram of an image
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	generate a contour plot file from an image w multiple panels
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 a plot file to a PostScript printer (LaserWriter?)
MFPRT	prints MF tables in a format needed by modelling software
PCNTR	Task to generate plot file for contour plus pol. vectors
PCUT	Cutoff in polarized intensity
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)
PLCUB	Task to plot intensity vs x panels on grid of y,z pixels
PLEV	Percentage of peak to use for contour levels
PLROW	Plot intensity of a series of rows with an offset.
PLVER	specifies the version number of a PL extension file
POLPLOT	specifies the desired polarization ratio before plotting.
PROFL	Generates plot file for a profile display.
PRTAB	prints any table-format extension file
PRTIM	prints image intensities from an MA catalog entry
PRTPL	Task to send a plot file to the line printer
QMSPL	Task to send a plot file to the QMS printer/plotter
SL2PL	Task to convert a Slice File to a Plot File
SNPLT	Plots selected contents of SN, TY or CL file
STARS	Task to generate an ST ext. file with star positions
STFND	Task to find stars in an image and generate an ST table.
SYMBOL	General adverb, probably defines a plotting symbol type
TEKSERVER	Information about the message server
TEKSRV	Information about the message server
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
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
TVPL	Display a plot file on the TV
TXPL	Displays a plot (PL) file on a terminal or line printer
UVHGM	Plots statistics of uv data files.
UVPLT	plots data from a UV data base
UVPRM	plots data from a UV data base
VBPLT	plots uv data and model from CC file
XBASL	Fits and subtracts nth-order baselines from cube (x axis)
XGAUS	Fits 1-dimensional Gaussians to images
XPLOT	Plots image rows one at a time on the graphics screen

15.26. POLARIZA

List of verbs, adverbs, tasks in category POLARIZATION

BDEPO	computes depolarization due to rotation measure gradients
FARAD	add ionospheric Faraday rotation to CL table
MAPBM	Map VLA beam polarization
PCUT	Cutoff in polarized intensity
PMODEL	Polarization model parameters
POLCO	Task to correct polarization maps for Ricean bias
POLPLOT	specifies the desired polarization ratio before plotting.
RM	Task to calculate rotation measure and magnetic field
SWPOL	Swap polarizations in a UV data base
VLABP	VLA antenna beam polarization correction for snapshot images

15.27. POPS

List of verbs, adverbs, tasks in category POPS

ABOUT	displays lists and information on tasks, verbs, adverbs
ABS	returns absolute value of argument
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
ATAN	Returns arc tangent of argument (half-circle)
ATAN2	Returns arc tangent of two arguments (full circle)
BY	gives increment to use in FOR loops in POPS language
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 - not implemented
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
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

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
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
HELP	displays information on tasks, verbs, adverbs
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
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
PASSWORD	Verb to change the current password for the login user
PCAT	Verb to list entries in the user's catalog (no log file).
POPSDAT	lists all POPS symbols, used to create them in MEmory files
POPSYM	Describes the symbols used in POPS
PRINT	Print the value of an expression
PROC	Define a POPS procedure using procedure editor.
PROCEDUR	Define a POPS procedure using procedure editor
PSEUDO	Description of POPS pseudoverbs - obsolete list file
PSEUDOVB	Declares a name to be a symbol of type pseudoverb
READ	Read a value from the users terminal
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
SCRATCH	delete a procedure from the symbol table.
SETDEBUG	Verb to set the debug print and execution level
SGDESTR	Verb-like to destroy named POPS environment save file
SGINDEX	Verb lists SAVE areas by name and time of last SAVE.
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
TAN        Tangent function
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.
THEN       Specified the action if an IF test is true
TIMDEST    Verb to destroy all files which are too old
TO         Specifies upper limit of a FOR loop
TPUT       Verb-like puts adverbs from a task in file for TGETs
TYPE       Type the value of an expression
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
WAITTASK   halt AIPS until specified task is finished
WHILE      Start a conditional statement
```

15.28. PROCEDUR

```
List of verbs, adverbs, tasks in category PROCEDURE

CROSSPOL   Procedure to make complex poln.  images and beam.
CXPOLN     Procedure to make complex poln.  images and beam.
HYB        Procedure to run a hybrid mapping loop
PFT        The Perley-Feigelson Test; see PFTLOAD.RUN, PFTEXEC.RUN
SETXWIN    Procedure to set BLC and TRC with TV cursor
TKNBOXS    Procedure to set Clean boxes 1 - n with the TK cursor
TKWIN      Procedure to set BLC and TRC with Graphics cursor
TVALL      Procedure loads image to TV, shows labeled wedge, enhances
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
VLACALIB   Runs CALIB and LISTR for VLA observation
VLACLCAL   Runs CLCAL and prints the results with LISTR
VLARESET   Reset calibration tables to a virginal state
VLBA       Procedure to read and process VLBA data (Phil Diamond)
```

15.29. PSEUDOVE

```
PSEUDO
Type:    General type of POPS symbol
Use:      Pseudoverbs are magic symbols which cause FORTRAN
       programs to carry out specific actions.    Unlike
       verbs, pseudoverbs are executed as soon as they are
       encountered by the compiler even in compile mode.
       In general, the FORTRAN programs 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 pseudoverb will not be
       executed.
```

15.32. 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.
****
List of verbs, adverbs, tasks in category SPECTRAL

ACFIT	Determine antenna gains from autocorrelations
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
BCHAN	sets the beginning channel number
BLOAT	converts pseudo-continuum to proper line UV data set
BLSUM	sums images over irregular sub-images, displays spectra
BPASS	computes spectral bandpass correction table
BPVER	specifies the version of the bandpass table to be applied
CHANSEL	Array of start, stop, increment channel numbers to average
CHINC	the increment between selected channels
CVEL	shifts spectral-line UV data to a given velocity
ECHAN	define an end for a range of channel numbers
FRPLT	Task to plot fringe rate spectra
HLPSPFLG	Interactive time-channel visibility Editor - internal help
HLPTVHUI	Interactive intensity-hue-saturation display - on-line help
HLPTVRGB	Interactive red-green-blue display - on-line help
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
PLCUB	Task to plot intensity vs x panels on grid of y,z pixels
POSSM	Task to plot total and cross-power spectra.
SMOTH	Task to smooth a subimage from upto a 7-dim. image
SPFLG	interactive flagging of UV data in channel-TB using the TV
SQASH	Task to sum together or average planes in a cube
SYSVEL	Systemic velocity
UV2TB	Converts UV autocorrelation spectra to tables
UVBAS	averages several channels and subtracts from uv data.
UVGLU	Glues UV data frequency blocks back together
UVLSF	least squares fit to channels and subtracts from uv data.
VBGLU	Glues together data from multiple passes thru the VLBA corr.
XBASL	Fits and subtracts nth-order baselines from cube (x axis)
XGAUS	Fits 1-dimensional Gaussians to images
XPLOT	Plots image rows one at a time on the graphics screen

15.33. TABLE

List of verbs, adverbs, tasks in category TABLE

DOTABLE	selects use of table-format for data
HF2SV	convert HF tables from FRING/MBDLY to form used by SOLVE
MFPRT	prints MF tables in a format needed by modelling software
PRTAB	prints any table-format extension file
TABED	Task to edit tables
TABEX	Edit Ancillary tables on UV or Image data

```
TACOP     task to copy tables
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
TAPQL     Convert an AIPS table to a POSTQUEL script
TASAV     Task to copy all extension tables to a dummy uv-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.
```

15.34. TAPE

List of verbs, adverbs, tasks in category TAPE

```
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
DOTABLE   selects use of table-format for data
FILLM     reads VLA on-line/archive format uv data tapes (post Jan 88)
FILLR     reads old VLA on-line-system tapes into AIPS
FITLD     reads tape to load FITS images or FITS UV files 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 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.
OUTTAPE   The output tape drive number.
PRTTP     prints contents of tapes, all supported formats
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
TCOPY     Tape to tape copy with some disk FITS support
TPHEAD    Verb to list image header from FITS or IBM-CV tape
UVLOD     Read export or FITS data from a tape or disk
VLAMODE   VLA observing mode
VLAOBS    Observing program or part of observer's name
```

15.35. TASK

```
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.
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
************************************************************
List of TASKs

ACFIT      Determine antenna gains from autocorrelations
ADDIF      Adds an IF axis to a uv data set
AFILE      sorts and edits MkIII correlator A-file.
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
ANCAL      Places antenna-based Tsys and gain corrections in CL table
APCLN      Deconvolves images with CLEAN algorithm
APGS       deconvolves image with Gerchberg-Saxton algorithm
APVC       Deconvolves images with van Cittert algorithm
ASCAL      Computes antenna-based gains based on source model (self-cal)
ASCOR      Applies ASCAL gain tables to other data sets
AVER       Averages over time UV data sets in 'BT' order
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
BATER      stand-alone program to prepare and submit batch jobs
```

FITTP	writes images / uv data w extensions to tape in FITS format
FRCAL	Faraday rotation self calibration task
FRING	fringe fit data to determine antenna calibration, delay, rate
FRPLT	Task to plot fringe rate spectra
FUDGE	modifies UV data with user's algorithm: paraform task
GAPLT	plots GAin table (ASCAL) by antenna, several per page
GEOM	regrids images with rotation, shift using interpolation
GETJY	determines calibrator flux densities
GLENS	models galaxy gravitational lens acting on 3 component source
GNMRG	merges gain files of ASCAL sort
GNPLT	plots extrema of gain solutions from ASCAL
GREYS	plots images as contours over multi-level grey
GRIDR	makes an image from single-dish data
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.
HF2SV	convert HF tables from FRING/MBDLY to form used by SOLVE
HGEOM	interpolates image to different gridding and/or geometry
HISEQ	task to translate image by histogram equalization
HLPIBLED	Interactive Baseline based visibility Editor - internal help
HLPSPFLG	Interactive time-channel visibility Editor - internal help
HLPTVFLG	Interactive time-baseline visibility Editor - internal help
HLPTVHUI	Interactive intensity-hue-saturation display - on-line help
HLPTVRGB	Interactive red-green-blue display - on-line help
HOLGR	Read and process Holography visibility data
HORUS	makes images from unsorted UV data, applying any calibration
IBLED	Interactive BaseLine based visibility EDitor
IM2UV	converts an image to a visibility data set
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 to load images to disk
IMMOD	adds images of model objects to an image
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	generate a contour plot file from an image w multiple panels
LGEOM	regrids images with rotation, shift using interpolation
LISTR	prints contents of UV data sets and assoc. calibration tables
LTESS	makes mosaic images by linear combination
LWPLA	translates a plot file to a PostScript printer (LaserWriter?)
MANDL	creates an image of a subset of the Mandlebrot Set
MAPBM	Map VLA beam polarization
MAPIT	Procedure to MAP and Self-Calibrate a UVDATA set
MAPIT_MX	MX adverbs not changed by MAPIT
MAPIT_NA	Procedure to MAP and Self-Calibrate a UVDATA set
MAPIT_UV	Procedure to MAP and Self-Calibrate a UVDATA set
MATHS	operates on an image with a choice of mathematical functions
MBDLY	Fits multiband delays from IF phases, updates SN table

SCMAP	Imaging plus self calibration loop
SDCAL	Task to apply single dish calibration
SDCLN	deconvolves image by Clark and then "SDI" cleaning methods
SDTUV	Task to convert SD table files to UV like data.
SELSD	Task to select random position single dish measurments
SETAN	Reads an ANtenna file info from a text file
SETJY	Task to enter source info into source (SU) table.
SHADW	Generates the "shadowed" representation of an image
SHOUV	displays uv data in various ways.
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
SMOTH	Task to smooth a subimage from upto a 7-dim. image
SNCOR	applies user-selected corrections to the calibration SN table
SNPLT	Plots selected contents of SN, TY or CL file
SNSMO	smooths and filters a calibration SN table
SOLCL	adjust gains for solar data according to nominal sensitivity
SPECR	Spectral regridding task for UV data
SPFLG	interactive flagging of UV data in channel-TB using the TV
SPLIT	converts multi-source to single-source UV files w calibration
SQASH	Task to sum together or average planes in a cube
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 and find image coordinates
SUBIM	Task to select a subimage from upto a 7-dim. image
SUMIM	Task to sum overlapping, sequentially-numbered images
SUMSQ	Task to sum the squared pixel values of overlapping,
SWPOL	Swap polarizations in a UV data base
TABED	Task to edit tables
TABEX	Edit Ancillary tables on UV or Image data
TACOP	task to copy tables
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
TAPQL	Convert an AIPS table to a POSTQUEL script
TASAV	Task to copy all extension tables to a dummy uv-file
TASRT	Task to sort extension tables.
TBAVG	Time averages data on 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
TKPL	Task to send a plot file to the TEK
TRANS	Task to transpose a subimage of an up to 7-dim. image
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	Task to load a map with histogram equalization

OFFHUINT	Proc which restores TV functions to normal after TVHUE
OFFPSEUD	Verb which deactivates all pseudo-color displays
OFFROAM	Procedure to clear the TV from a Roam condition
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
OFMFILE	specifies the name of a text file containing OFM values
PCNTR	Task to generate plot file for contour plus pol. 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
RGBMP	Task to create an RGB image from the 3rd dim of an image
ROAM	Roam around an image too large for the display.
ROMODE	Specified roam mode
SETROAM	Verb to set roam image mode, then does roam
SETXWIN	Procedure to set BLC and TRC with TV cursor
TBLC	Gives the bottom left corner of an image to be displayed
TTRC	Specifies the top right corner of a subimage to be displayed
TV3COLOR	Verb to initiate 3-color display using 3 TV channels
TVALL	Procedure loads image to TV, shows labeled wedge, enhances
TVANOT	Verb to load anotation to the TV image or graphics
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)
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
TVDIC	Task to copy a TV screen-image to a Dicomed film recorder.
TVFIDDLE	Verb enhances B/W or color TV image with zooms
TVFLUX	displays coordinates and values selected with the TV cursor
TVHLD	Task to load a map with histogram equalization
TVHUEINT	Verb to make hue/intensity display from 2 TV channels
TVHUI	make TV image from images of intensity, hue, saturation
TVHXF	Task to calculate transfer function based on histogram
TVINIT	Verb to return TV display to a virgin state
TVLABEL	Verb to label the (map) image on the TV
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
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).

SHOUV	displays uv data in various ways.
SMOOTH	Specifies spectral smoothing
SNPLT	Plots selected contents of SN, TY or CL file
SORT	Specified desired sort order
SPECR	Spectral regridding task for UV data
SPFLG	interactive flagging of UV data in channel-TB using the TV
SPLIT	converts multi-source to single-source UV files w calibration
STOKES	Stokes parameter
SWPOL	Swap polarizations in a UV data base
TBAVG	Time averages data on all baselines.
TVFLG	interactive flagging of UV data using the TV
UBAVG	Baseline dependent time averaging of uv data
UCAT	list a user's UV and scratch files on one or more data areas
USUBA	Convert a subset of uv data to a specified subarray
UV1TYPE	Convolving function type 1, pillbox or square wave
UV2MS	Append single source file to multisource 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
UVADC	Fourier transforms and corrects a model and adds to uv data.
UVAVG	Average or merge a sorted (BT, TB) uv database
UVBAS	averages several channels and subtracts from uv data.
UVBOX	width of the smoothing box used for uniform weighting
UVCMP	Convert a UV database to or from compressed format
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.
UVDGP	Copy a UV data file, deleting a portion of it
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
UVGLU	Glues UV data frequency blocks back together
UVHGM	Plots statistics of uv data files.
UVIMG	Grid UV data into an "image"
UVLIN	Fits and removes continuum visibility, also can flag
UVLOD	Read export or FITS data from a tape or disk
UVLSF	least squares fit to channels and subtracts from uv data.
UVMOD	Modify UV database by adding a model or models
UVMTH	Averages one data set and applied it to another.
UVNOU	flags uv samples near the V (U=0) axis to reduce interference
UVPLT	plots data from a UV data base
UVPOL	modifies UV data to make complex image and beam
UVPRM	plots data from a UV data base
UVPRT	prints data from a UV data base with calibration
UVRANGE	Specify range of projected baselines
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

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
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
MAX	returns the maximum of its two arguments
MAXFIT	returns pixel position and image intensity at a maximum
MCAT	displays images in the user's catalog directory
MIN	returns the minimum of its two arguments
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
OFFHUINT	Proc which restores TV functions to normal after TVHUE
OFFPSEUD	Verb which deactivates all pseudo-color displays
OFFROAM	Procedure to clear the TV from a Roam condition
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
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 an 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
OFMTWEAK	interactive modification of current TV OFM lookup tables
OFMZAP	deletes an OFM lookup table save file
PASSWORD	Verb to change the current password for the login user
PCAT	Verb to list entries in the user's catalog (no log file).
PRINT	Print the value of an expression
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
QHEADER	Verb to summarize the image header: positions at center
QIMVAL	Verb to determine pixel value at specified position
QUEUES	Verb to list all submitted jobs in the job queue
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
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.
REWIND	Verb to rewind a tape
ROAM	Roam around an image too large for the display.
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
SCRATCH   delete a procedure from the symbol table.
SCRDEST   Verb to destroy scratch files left by bombed tasks.
SET1DG    Verb to set 1D gaussian fitting initial guesses.
SETDEBUG  Verb to set the debug print and execution level
SETROAM   Verb to set roam image mode, then does roam
SGDESTR   Verb-like to destroy named POPS environment save file
SGINDEX   Verb lists SAVE areas by name and time of last SAVE.
SHOW      Verblike to display the TELL adverbs of a task.
SIN       Compute the sine of a value
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
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
TABGET    returns table entry for specified row, column and subscript.
TABPUT    replaces table entry for specified row, column and subscript.
TAN       Tangent function
TGET      Verb-like gets adverbs from last GO of a task
TGINDEX   Verb lists those tasks for which TGET will work.
THEN      Specified the action if an IF test is true
TIMDEST   Verb to destroy all files which are too old
TK1SET    Verb to reset 1D gaussian fitting initial guess.
TKAGUESS  Verb to re-plot 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
TKBOX     Procedure to set a Clean box with the TK cursor
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
TV3COLOR  Verb to initiate 3-color display using 3 TV channels
TVANOT    Verb to load anotation to the TV image or graphics
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)
TVCUBE    Verb to load a cube into tv channel(s) & run a movie
TVFIDDLE  Verb enhances B/W or color TV image with zooms
TVHUEINT  Verb to make hue/intensity display from 2 TV channels
TVINIT    Verb to return TV display to a virgin state
TVLABEL   Verb to label the (map) image on the TV
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
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
TVROAM	Load up to 4 TV image planes and roam a subset thereof
TVSCROL	Shift position of image on the TV screen
TVSLICE	Set slice endpoints on the TV interactively
TVSPLIT	Compare two TV image planes, showing halves
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
UCAT	list a user's UV and scratch files on one or more data areas
UNQUE	remove a given job from the job queue
VALUE	Convert a string to a numeric value
VERB	Declares a name to be a symbol of type verb
WAITTASK	halt AIPS until specified task is finished
WEDERASE	Load a wedge portion of the TV with zeros
ZAP	Delete a catalog entry and its extension files

15.41. VLA

List of verbs, adverbs, tasks in category VLA

CLCOR	applies user-selected corrections to the calibration CL table
CVEL	shifts spectral-line UV data to a given velocity
DAYFX	Fixes day number problems left by FILLM
FARAD	add ionospheric Faraday rotation to CL table
MAPBM	Map VLA beam polarization
USERLIST	Alphabetic and numeric list of VLA users, points to real list
VLABP	VLA antenna beam polarization correction for snapshot images

15.42. VLBI

List of verbs, adverbs, tasks in category VLBI

ACFIT	Determine antenna gains from autocorrelations
AFILE	sorts and edits MkIII correlator A-file.
ANCAL	Places antenna-based Tsys and gain corrections in CL table
BLING	fringe fit residual rate and delay on individual baselines
CL2HF	Convert CL table to HF table
CLCOR	applies user-selected corrections to the calibration CL table
CLPLT	plots closure phase and model from CC file

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

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* × *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

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 mapmaking 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 thumb-wheel 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

$3n \log_2 n$ 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 $\hat{g}_{\text{approx}}$—i.e., $\hat{g}_{\text{approx}} = \chi_B \hat{f} + \text{error}$. 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

$$f_{n+1} = f_n + \mu\chi_A \cdot (\hat{g}_{\text{approx}} - \chi_B \hat{f}_n)^{\check{}} .$$

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* nonnegativity constraint—at each iteration, any pixel value which would be driven negative is modified to become nonnegative. 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

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]). Nonnegativity 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 subregion (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 $\hat{f} : \mathbf{R}^n \to \mathbf{C}$ 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)

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 **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 $n/2a$, $n = 0, \pm 1, \pm 2, \ldots$, 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 $\hat{f}(m\Delta u, n\Delta v)$, $m, n = 0, \pm 1, \pm 2, \ldots$, 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 *subimage* 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 subimage, 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

$$(*) \qquad \nu f(\eta) = \int_{-1}^{1} e^{ic\eta t}(1-t^2)^{\alpha} f(t)\,dt\,;$$

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 $(*)$ 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

$$\frac{\iint_{\text{map}} |\hat{C}(x,y)|^2 (1-(2x\Delta u)^2)^{\alpha}(1-(2y\Delta v)^2)^{\alpha}\,dx\,dy}{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} |\hat{C}(x,y)|^2 |1-(2x\Delta u)^2|^{\alpha}|1-(2y\Delta v)^2|^{\alpha}\,dx\,dy}$$

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

$$S = \begin{pmatrix} \langle E_x(t)\overline{E}_x(t+\tau)\rangle\widehat{\ } & \langle E_x(t)\overline{E}_y(t+\tau)\rangle\widehat{\ } \\ \langle E_y(t)\overline{E}_x(t+\tau)\rangle\widehat{\ } & \langle E_y(t)\overline{E}_y(t+\tau)\rangle\widehat{\ } \end{pmatrix}.$$

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 nonnegative. 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 mapmaking tasks are able to recognize the form of their input dat- and deal with them appropriately.)

subimage — 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.*

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 piecewise 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 mapmaking 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 steplength. 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 $|u_1| \geq |u_2| \geq \cdots \geq |u_n|$ — 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 inaccessi-

ble 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 radiointerferometric 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.*

Appendix Y. 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.

Y.1. Blocks *versus* Bytes

Conventionally, we are used to working in units of *blocks* for the sizes of files. Unfortunately, this is not a very useful unit since *blocks* mean different things in different places *i.e.*, a *block* is a completely arbitrary unit. On the other hand, a *byte* is a well defined unit; it is eight bits regardless of context. It makes much more sense to measure the sizes of files in terms of *bytes*. You are encouraged to develop this healthier way of thinking; it makes for more meaningful communication when you request disk space! (Just to be awkward, a block on a Convex, SUN or VAX is **now** 1024 bytes!)

To calculate the size of the data files (in bytes, of course), you must first understand just how data is stored on the data disks.

Y.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 so for spectral line databases. All of the calibration routines can accommodate either the compressed or the uncompressed formats. Compressed data files can be identified by the dimension of 1 for the "COMPLEX" 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.

How is this achieved? Essentially, this is done by assigning only one each of the "WEIGHT" and "SCALE" random parameters per visibility record and packing the real and imaginary terms into one REAL value with magic value blanking. This is to be compared with the uncompressed format in which each of the real, imaginary and weight terms are stored in a REAL.

Y.2.1. When NOT to use compressed format

In general, compressed data should be used and this is the default in the task **FILLM**. At worst (single frequency, single IF, single polarisation data), you will not save any disk space. In all other cases, there are respectable savings to be made. However, the use of a packed REAL 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 32000:1, then you must use **uncompressed** data format. A situation where this might occur is where you have a spectrum in which there are maser lines of 1 Jy and > 32000 Jy; in this case, you should not use compressed data. **In general, continuum datasets should be loaded with data compression** since the above spectral dynamic range limit will not be encountered normally.

Y.3. How is data stored on disk?

There are two kinds of file types in *AIPS*. The first are *map* files — this is a generic term for *uv* and image files. The second are regular disk files; these are generally what are known as "extension files" in *AIPS* *e.g.*, PL files or CL tables.

uv data files contain information about the coherence function of a wavefront at random locations and random times and 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 is stored on disk in a manner similar to the way that it would be organised on a FITS "random group" tape. The data is 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 polarisation combination for each frequency/IF. The first part of each logical record contains what are known as "random parameters" *e.g.*, spatial frequency co-ordinates and time. After the random parameters, there is a small, regular array of data.

For an uncompressed, multi-source data set such as might be created by **FILLM**, the random parameter group will contain the following:

UU-L-SIN, VV-L-SIN, WW-L-SIN, TIME1, SOURCE, BASELINE

"UU-L-SIN" *etc.* indicate that the spatial frequency coordinates are computed with a sine projection and are given in terms of wavelengths at the reference frequency. "TIME1" is the label for the time for historical reasons. "BASELINE" and "SOURCE" are self-obvious labels. If you have frequency table identifiers (which is always the case for current data) then there will be an additional random parameter, "FQSEL". For a compressed data base, two additional random parameters will be required — "WEIGHT" and "SCALE".

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". This will have a dimension of 3 for uncompressed data (real, imaginary, weight) and be of dimension 1 for compressed data. The other axes of the regular array are "RA", "DEC", "FREQ" and "STOKES".

Y.4. *uv* database sizes

The size of the database to be loaded to disk with **FILLM** is given by the following formula:

$$\Big[\{(\#\ random\ parameters) + [(dimension\ of\ COMPLEX\ axis) \times (\#\ polns) \times (\#\ freqs) \times (\#\ IFs)]\} \times (\#\ vis) \times 4\Big] \text{ bytes}$$

The number of visibilities is given 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.

So, for example, a 12 hour observation with 30 second integrations, 1 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 § Y.5. If this database had been written in uncompressed format, the *uv* data would have occupied around 62 Mbyte.

Consider another example. IMHE generated the 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
```

How much space does this compressed (`COMPLEX Pixels = 1`) *uv* database occupy? There are 9 random parameters, 4 polarisations, 1 frequency and 2 IFs. The listing also shows that the database contains 105198 visibilities. So, the size of the *uv* part of the database is:

$$\left[\{9 + [1 \times 4 \times 1 \times 2]\} \times 105198 \times 4\right] \text{ bytes}$$

$$= 7.153 \text{ Mbytes}$$

Y.5. How big are the extension tables?

Some extension tables — principally the CL tables — can become rather large.

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 polarisation pairs, with entries every 5 minutes, the CL table will occupy about

$$\left[\{15 + [14 \times 2 \times 4]\} \times 4 \times \frac{(12 \times 60)}{5} \times 27\right] \text{ bytes}$$

$$= 1.975 \text{ Mbytes}$$

Since the other files are noticeably smaller, their sizes can be ignored.

Y.6. How big are my image files?

Since images are regular arrays, the sizes of image files are easier to calculate than for *uv* data files. Generally, the images are stored as floating point numbers *i.e.*, 32 bits per pixel so in bytes the image file size 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 will be increased a little due to the way in which the system allocates space for files.

Y.7. Storing data on tape

Eventually, you will wish to write your images or *uv* databases to tape for archival purposes and for transfer to another machine. How much data can you reasonably expect to fit on a tape?

Current tape drives are *nine track* devices *i.e.*, the data is recorded on eight tracks and the ninth track is used to record *parity bits* for error checking. Standard recording densities are 800, 1600 and 6250 bits per inch per track (*bpi*). Generally, data is recorded at 6250 bpi wherever possible.

Each tape has a reflective metallic marker near the start of the tape to signify *beginning-of-tape* or BOT. The BOT marker is used to allow the tape drive to position the tape at the starting position. Similarly, an EOT (*end-of-tape*) maker is located near the physical end of the tape. This is sensed by the tape drive and control circuitry prevents the device spooling the tape completely off the main spool; this is very useful for self-loading tape drives!

Note that plot (PL) files are not saved to tape with the task **FITTP**.

Y.7.1. Tape sizes

The standard tape length on a regular size reel is 2400 feet long. However, by the use of a thinner tape substrate for the oxide layer (made of Mylar), the regular spool can accomodate up to 3600 feet of tape.

Y.7.2. Blocking density

Data are written on to the tape in *blocks*. In general, the block size is 2880 bytes. After each block, an *inter-record gap* is left before the next block is written. The *tape blocking efficiency* is a measure of the amount of essentially wasted space. It is defined as

$$\frac{\frac{block\ size}{recording\ density}}{[\{\frac{block\ size}{recording\ density}\} + length\ of\ inter\text{-}record\ gap]}$$

The lengths of the inter-record gaps are roughly $\frac{3}{5}$ of an inch at recording densities of 800 and 1600 bpi and $\frac{3}{10}$ of an inch at 6250 bpi. This gives a tape blocking efficiency of around 0.61 for a 6250 bpi tape using 2880 byte blocks.

It is possible to squeeze more data on to tape by reducing the amount of space given over to the inter-record gaps. This is what the *AIPS* adverb **BLOCKING** lets you control in the task **FITTP**. BLOCKING = n assembles the data into block sizes of $n \times 2880$ bytes before writing the data to tape. So, for BLOCKING = *10*, the block sizes are 28,800 bytes long and the tape blocking factor improves to around 0.95 for a 6250 bpi tape. In other words, you can fit more than half again on to the same tape.

In round numbers, the capacity of a standard 2400 feet tape is therefore about 110 Mbytes using BLOCKING = 1 and around 170 Mbyte using BLOCKING = 10 (the maximum). The default value is BLOCKING = 2 and this will allow you to record around 135 Mbyte on the standard tape at 6250 bpi.

Y.7.3. Tape formats

Several formats are available within the task **FITTP** and are controlled through the adverb **FORMAT**. The formats are:

(1) 16 bit integer

(2) 32 bit integer

(3) 32 bit IEEE floating point

For the integer n formats, *AIPS* finds the maximum and minimum value in the image or data file and maps the range on to the integers between $2^{n-1} - 1$ and $-(2^{n-1} - 1)$. (A sign bit is also required.) This is what is happening when *AIPS* reports "First find the scaling parameters" when using the integer formats when the task **FITTP** starts up.

For the 16 bit integer format, the effective "dynamic range" with which images or *uv* datasets can be stored on tape is thus about 65000:1. For many situations, this format is perfectly adequate. If your image or data has a dynamic range of greater than this then you should use one of the 32 bit formats. Which one should you use? For most purposes, it is better to use the 32 bit floating point format. For host computers which support the IEEE 32 bit standard, the acts of reading from and writing to tape then become simple copies; no bit manipulation has to be performed. Consequently, the tape reading and writing tasks are executed much faster. A more minor 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 "quantisation errors". This may be important for software development.

Y.7.4. How *AIPS* writes the data on tape

The preceeding sections do not tell the full story, however. *AIPS* does currently does not write compressed *uv* data to tape in a compressed format. Instead, it effectively expands the data into the uncompressed form first then writes the data on tape (with scaling for the integer formats, if required).

In the conversion, the real and imaginary values that were stored in one packed real 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 these are required no longer. If the data is written in 16 bit format, the 'TIME1' random parameter will be stored in **two** random parameters to maintain the required precision. Consequently, for the 16 bit format, the compressed data is expanded to

$$\frac{\{(\#\ random\ parameters - 1) + [(\#\ pol) \times (\#\ IFs) \times (\#\ frequencies) \times 3]\}}{\{(\#\ random\ parameters) + [(\#\ pol) \times (\#\ IFs) \times (\#\ frequencies)]\}}$$

the original size (where *# random parameters* is the original number in the compressed database). After the expansion, the data is scaled for the integer formats, if required. For the 16 bit format, the size of the database on tape is half of the expanded size since the 32 bit format on disk is mapped to 16 bits on tape. (For the 32 bit formats, the number of random parameters in the expanded database is 2 less than were in the original database.)

Consider the example of the following multi-source spectral line data base stored on disk in compressed format. There were seven channels at 2 IFs, each with 2 polarisations. The random parameters were — UU-L-SIN, VV-L-SIN, WW-L-SIN, TIME1, SOURCE, BASELINE, FREQSEL, WEIGHT and SCALE. The database contained 834031 visibilities. From § Y.4, 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 16 bit integer format, it is first expanded by a factor of

$$\frac{\{(9-1) + [2 \times 2 \times 7 \times 3]\}}{\{9 + [2 \times 2 \times 7]\}}$$

$$= 2.486$$

after which it is effectively reduced by a factor of two by the conversion to the 16 bit format. Consequently, the data will occupy

$$\frac{123 \times 2.486}{2}\ \text{Mbytes}$$

$$= 153\ \text{Mbytes}$$

on tape. In other words, this data base and all the associated extension files are likely to fit on one standard, 6250 bpi tape using BLOCKING = 10 (§ Y.7.2) and FORMAT = 1 (§ Y.7.3).

Y.8. Help! I have a very large database that I can't fit on one tape.

FITTP can not write multi-volume tapes. Some spectral line databases (and perhaps some continuum databases) may be so large that the file can not fit on one tape even with FORMAT = 1 and BLOCKING = 10. What can you do to backup your data? 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, then back these up individually. Another way to solve the problem is to realise 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.

Y.9. An example

Images are stored in floating point as 4 bytes/pixel. This means that a typical image of 512×512 pixels will occupy around 1 Mbyte on disk. Backing up such images to tape in 32 bit IEEE floating point (FORMAT = 3) with BLOCKING = 10 will allow you to fit over 150 such images to a standard 2400 feet tape at 6250 bpi.

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 two Tektronix 4511a color printers, one in Charlottesville and one at the AOC in Socorro. You may display your PostScript file on these printers simply by typing

`$ lpr -Ppscolor` *filename* ↵ where *filename* is the name of your file.

`$ lp -Ppscolor` *filename* ↵ for Solaris systems on Suns

Just before doing this, make sure that the desired type of special Tektronix paper (plain or transparency) is in the cassette mounted in the printer. The paper size is 8.5 × 11 inches, which is the default for *AIPS* tasks **TVCPS** and **LWPLA**. 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. *AIPS* print routines will re-direct PostScript files that actually contain color commands to **pscolor**, but will not re-direct ordinary print jobs to some printer other than **pscolor**. There are some special instructions for the color printer at the AOC in § Z.3.4.

Z.1.2.2. Software to copy your screen

To obtain a color hardcopy of what is on your screen, there are three software options you can choose. These are **TVCPS**, **xv**, and **xgrab**. Having created a PostScript file, you can print it on color printers and film recorders at the NRAO, or copy the file via e-mail or **ftp** to some other site for printing.

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 § 2.5). 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 **pscolor** or the Solitaire. You can, of course, edit the save file (if you are a PostScript wizard) and can insert the file (since it is encapsulated) in another document. See the Charlottesville Workstation Guide for a short chapter on PostScript.

The **xv** program (in version 2) is a Unix utility program available on most systems at the NRAO. (Version 3 of **xv** is significantly improved, but, due to copyright restrictions, is not yet generally available 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. The bell will ring and the cursor will change to a white cross symbol; move it to the top left of the area you want to grab. Then press and hold down the left 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, it will beep twice, and whatever you grabbed appears in the main **xv** window. You can now use the "save" button of the control window to save this as any format you want. Once 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.

The **xv** program can either be started by typing **xv** from the Unix command line, or by choosing the XV option from the programs menu (which is obtained by pressing the middle mouse button on the root window of the X-Windows display).

Finally, the **xgrab** program provides similar functionality to the "grab" feature of **xv**, with fewer output formats but much more control over how the grabbing is done. By default it allows three seconds in between starting the grab and when it actually starts to read the screen; this can be useful for setting things up. Also, it un-maps itself from the window when grabbing so you don't have to worry about getting it out of the way. Unfortunately, there appears to be some problems with its encapsulated PostScript output.

Z.1.2.3. Color film recorders

The NRAO owns two Solitaire film recorders, one in Charlottesville and one at the AOC in Socorro. These recorders support two film sizes, 35mm and 4 × 5 (inches that is). Our normal film type, for which the recorder is very well calibrated, is Ektachrome 100+ Professional. The film recorder has 3 scan resolutions, 2048, 4096 and 8192 lines. Any one of these may be used with any input image. The higher resolution scans take rather longer to do, but will clearly do a better job on larger input images.

Actually, the Solitaire is driven by software called Freedom of Press (**freedom**) from Custom Applications, Inc. which interprets/filters PostScript input files into a form understood by the Solitaire. (The Freedom of Press software has a great many other options as well, with a wide variety of output formats. See the chapter on the Solitaire in the Charlottesville "Sun Workstation Notes" for details.) For PostScript files that need no special processing by **freedom**, you may simply "print" them to special queues set up for the Solitaire. These queues are, of course, actually filtered by **freedom** on the way to the film recorder. The apparent plot size for the Solitaire is 7.33 × 11 inches for 35mm film and 8 × 10 inches for 4 × 5 film. *AIPS* tasks **LWPLA** and **TVCPS** should be given these dimensions in their inputs for plots intended for NRAO's film recorders. These tasks can print directly to the Solitaire queues, if they are among the choices offered when you start **AIPS**, but we recommend that you send the plots to disk text files instead. In this way, you may send the plots to the film recorder in the queue of your choice **after** making sure that the device has the correct film type mounted and ready for use.

The details of how you may have the film type changed, have the film processed, get the resulting pictures/slides back, have the unit cleaned and repaired, and the like differ between the two sites and will be discussed in the appropriate Sections below.

In the hopes of producing both better pictures and a usable NRAO image library, Pat Smiley of the Charlottesville Graphics department has volunteered to coordinate the making of photographs using the Charlottesville film recorder. To avail yourself of this offer, you must (1) copy the picture file to the appropriate disk and (2) notify Pat that you have done this. In particular, to copy the file from a Charlottesville computer:

$ cp *file_name* /net/solitaire/solitaire/ftp/pub/slides ↵

Use anonymous ftp to copy the file from more remote computers:

$ ftp solitaire.cv.nrao.edu ↵	or 192.33.115.79
Connected to solitaire.cv.nrao.edu.	connect message returned
220 solitaire FTP server (SunOS 4.1) ready.	server message returned
Name (solitaire:...): anonymous ↵	reply with anonymous or ftp.
331 Guest login ok, send ident as password.	login message and suggestion.
Password: *your_address* ↵	reply with your account and e-mail address, *e.g.*, **anewman@nrao.edu**; it will not be visible on the screen.
230 Guest login ok, access restrictions apply.	login is successful
ftp> cd pub/slides ↵	change to slides directory

Z.1.4. Solving problems at the NRAO

Below are details specific to the Charlottesville and Socorro systems for handling some of the problems which may arise in *AIPS*.

Z.1.4.1. Booting the workstations

Modern workstations, especially the powerful IBM RS/6000s, 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** (or **lpstat** for Solaris 2.*x*) 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. At this writing, the Charlottesville systems are using a 20-minute delay time.

Finally, check to see if the printout was (a) diverted to the "big" printer (**lp27** 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 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*
$ lpstat -P*ppp* ↵	to show printer *ppp* under Solaris

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 will announce the names of any files that it removes and is 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).

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. Beginning with the `15JUL94` release, 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.

It is not unusual for AIPS jobs to be in the 1 Mbyte or more in length, which will take 5–10 minutes to print. For large text files, it is quite likely that the `ZLPCL2` shell script will divert the job to a "big" printer (in Socorro, `lp27` in room 213). However, graphics files are not subject to such restrictions.

If you plan on generating large or very complex plot files which you intend to print, please select the `psnet` printer at the AOC or the `ps3dup` printer in Charlottesville. Since they are, effectively, on the ethernet, the bandwidth to it is usually an order of magnitude faster than any serial line. You — and others — will have to spend less time waiting for jobs to come out of the printer. If you are submitting jobs which you know are several Mbyte in size, we ask that you wait until after local business hours to avoid tying up the printer.

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.

Z.1.4.6. Tapes won't mount

Occasionally, both local and remote tape mounts may not work successfully. The source of the problem is often your failure to load the tape physically into the device or to wait until the device is ready to read the tape. DATs and Exabytes, in particular, go through lots of clicking and whirring before they are really ready. An error message like

```
AIPS 1:  ZMOUN2:  Couldn't open tape device /dev/nrst0
```

(or some other tape-device name gibberish) is to be expected in this case.

If you attempt to mount a remote tape and get the messages:

```
AIPS 1:  ZMOUNR: UNABLE TO MOUNT REMOTE TAPE DEVICE, ERROR 96
AIPS 1:  AMOUNT: TAPE IS ALREADY MOUNTED BY TPMON
```

it means that your **AIPS** and the tape dæmon that you are using disagree on whether the tape is already mounted in software. The most probable reason for this is that you are attempting to mount someone else's tape (check your inputs and the labels on the device closely) — or that the previous user of the device dismounted the tape from the hardware but neglected to do it from software. In this case, you have two choices: (1) find the culprit and have him do a software dismount, or (2) find an *AIPS* Manager to kill the confused dæmon and restart it. (If you are using tape device n on computer *host_name*, then you need to stop the process called **TPMON***m*, where $m = n+1$ on computer *host_name* and then start it again by running **/AIPS/START_TPSERVERS** on that computer. This should be done by an *AIPS* Manager.)

If you attempt to mount a remote tape and see, instead, the messages:

```
ZVTPO2 connect (INET): Connection refused
AIPS 1:  ZMOUNR: UNABLE TO OPEN SOCKET TO REMOTE MACHINE, ERROR 1
AIPS 1:  ZMOUNT: ERROR 1 RETURNED BY ZMOUN2/ZMOUNR
```

then the tape dæmons are not running on the remote machine. Log into the remote machine as user **aips** and type:

```
/AIPS/START_TPSERVERS
```

After a minute or two, you should see some messages from **STARTPMON** about starting **TPMON** dæmons. Alternatively, you could exit from **AIPS** and start back up again, including **tp=***host_name* on the **aips** command line; see § 2.2.3. If the tape still doesn't mount after doing this, see the *AIPS* Manager.

Z.1.4.7. 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 by typing **FREESPAC** in the **AIPS** session, listing 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 Jon Spargo (at the AOC) or an *AIPS* Manager about enabling your user number.

Then, when you run **AIPS** in that window, the local disks will be the ones attached to **rhesus**. Under most circumstances, the **aips** procedure will be able to figure out which Sun you are actually typing on and use it for the *AIPS* TV, message, and graphics servers (all both executing and displaying on your Sun).

Z.2.2. Using the tape drives in Charlottesville

For a general discussion of magnetic tapes, including the *required* software mount, see § 2.4. The following describes how to deal with the physical tape drives themselves.

Z.2.2.1. Mounting and removing tapes on 9-track drives

If the front door of the tape unit is not open, it may be in use. Ask around before doing anything. When you're certain that it's okay, press the "offline" button, then press the "Unload/Open" button. The door should drop open for you.

Once the door is open, get ready to put your 9-track tape in. Put a write ring in the tape only if you intend to write on the tape during this *AIPS* session. These drives do not support auto-load rims, so all rims must be removed from the tape before loading. Slide the tape spool in sideways with the label facing up until it sits on the central hub (it will feel like it's balanced). Then close the door. After the display indicates that the drive is ready, you can perform the software mount from **AIPS**.

Z.2.2.2. Mounting tapes on Exabyte and DAT drives

Exabyte (8mm) and DAT (4mm) drives have a window or opening through which a mounted tape may be seen. Before touching anything, look in the window or opening to see if there is already a tape in the drive. If there is, ask around to make sure that the tape is no longer in use. Remember that the user of the drive may be in an office as much as two floors away and that Unix does not provide much protection. If you dismount a remote user's tape and mount your own, that user may well write on it, thinking that he is writing on his own tape, without knowing that he is destroying all your data.

On most drives, there will be a single button on the front panel of the device somewhere. When the device becomes available, press this button to open the door. If there was already a tape in the drive, it will be ejected after some whirring and clanking and a few seconds. If a tape is ejected, remove it. Now put your tape in the drive, label facing upwards. On Exabytes, push the door closed gently. For DAT drives, lightly push the tape into the drive until the device "grabs" the tape and pulls it in the rest of the way. 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).

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 access the drive. The first access is, of course, the software **MOUNT** command from inside **AIPS**.

Z.2.3. Color hard copy in Charlottesville

Having created a PostScript file containing color commands or pictures (see §§ Z.1.2.), you can print it either on a Tektronix 4511a color printer known as `pscolor` in the *AIPS* Caige (with plain or transparency "paper") or send it to the Solitaire film recorder. Special Tektronix "paper" is loaded into cassettes, one of which is inserted into the printer. One cassette is used for plain paper and the other for transparency paper. The latter is rather expensive, so be reasonable in its use. Additional boxes of paper may be found inside the cabinet on which the printer rests. When the little green light on the right hand side of the printer is blinking, the printer is either receiving data or computing on data already received. The computer inside the printer is actually rather fast, but a color picture is usually a large data file which takes a while to transmit and display. When the printer starts actually printing, it must move the paper in and out three times, one for each color. When it is done, it will eject the paper fully.

The Solitaire film recorder is in the room across the hall from the *AIPS* Caige. To see details on its use, consult the appropriate chapter of the Charlottesville "Sun Workstation Notes." In those notes, it tells you how to change film and to swap film holders. We suggest that you do *not* do these operations. Instead, have someone from the Graphics Department (George Kessler or Pat Smiley) preferably or a computer technician (Warren Richardson or Gene Runion) to assist you. Cleaning and repairs to the device should be done only by Gene and Warren. We have found that pictures made by the device are very sensitive to dust and particles of paint (from the device itself) and, hence, think it best if the device is handled only by trained personnel.

The Solitaire has an LCD display which usually shows the status of the film recorder. On the left is the film counter, telling how many frames have been exposed since a film load or a counter reset. Towards the middle is the machine's status, which usually displays Idle, Calibrating, Computing Geometry, or Exposing. When the machine is idle, output can be dumped to it (assuming there is no PostScript in the process of being interpreted for the film recorder already). Image recorder calibration is done every half hour, as well as when any job is sent to the recorder. Computing Geometry is displayed whenever a new film holder has been placed onto the Image Recorder. On the upper right, G2 is normally displayed, which means that it's listening to GPIB address 2 (GPIB is similar to SCSI). Under G2, the number of frames left on the roll is shown. The Solitaire has two status lights on the console; a green light for exposing (the light will vary in intensity with the actual light beam), and a red light to signify that it's not ready.

You should check the status display, then send your PostScript file to the film recorder, and then check the display again, all to make sure that the desired operation has taken place. The queues are named `35mm2k`, `35mm4k`, `35mm8k`, `4x5-2k`, `4x5-4k`, and `4x5-8k`, all with the obvious meaning. Have the film holder changed to the desired one before submitting a job to its queue. After you have exposed the film, ask someone from the Graphics Department to remove the film and have it processed. Rolls of 35mm slides taken to be processed before 10 a.m. can be back as early as 2 p.m. All other processing takes until the next business day. If you are not in a hurry, you should wait for more pictures on a roll to be exposed. However, the cost of processing a whole roll for one picture, on an occasional basis, is not significant.

Even though you are in Charlottesville, you should consider Pat Smiley's offer of help with your photographs. See the comments and instructions given in § Z.1.2.3.

Z.3. *AIPS* at the NRAO AOC in Socorro

The computing power at the NRAO Array Operations Center ("AOC") in Socorro is no longer concentrated solely in the *AIPS* "Caiges" — there are now several public Sun IPX workstations scattered throughout the building, most with at least one Exabyte and one DAT tape drive and 2 Gbyte or more of disk space. There are also eight public IBM RS/6000 machines, ranging from a 320 (Zuni) to several 580s. Each IBM is equipped with at least one Exabyte tape drive, and many have a DAT drive as well. The disk space on the IBMs ranges from 4 to 16 Gbytes. Several printers are accessible, most of which are located in the computer room, ranging from a DEC LP27A line printer to a Tektronix color printer ideal for color prints or overheads. Color slides may also be made (with about a week turn-around time) by directing the output to our Solitaire film recorder. (More specialized assistance with photographs and a faster turn-around time, if needed, can be obtained by sending your picture files to Charlottesville; see § Z.1.2.3.)

Z.3.1. Using the AOC workstations

Z.3.1.1. Signing up for *AIPS* time in Socorro

Jon Spargo (505-835-7305) is responsible for the assignment of time on all public AOC machines; reservations must be made at least two weeks in advance. Last-minute sign-ups for vacant time and any exchanges of assigned time must also be cleared by Jon. Normally, sign-ups will be limited to a maximum of two weeks, although exceptions are made occasionally. Visitors have exclusive rights to their assigned workstations during their visit, although tape drives (when not in use) are fair game for remote users. Any data left on disk after a user's scheduled time will be deleted promptly. See "Dr. Delete's Public Workstation Allocation Board" next to room 258 for the machine assignments.

Z.3.1.2. Managing workstation windows at the AOC

The Suns and IBMs have different window managers at the AOC.

To start **AIPS** on either architecture, choose the window that you want to work with (or create a new one), and type:

```
aips tst pr=7
```

This example command selects the **TST** version of **AIPS** (the default) and chooses printer number 7 (the **psnet** printer in the computer room) as the default printer for text and graphics output. Other options (such as cross mounting data disks from other machines and checking the status of remote tape daemons) can be chosen on the command line as well. These options are explained in some detail in § 2.2.3 and may be viewed in even greater detail by typing HELP AIPS $^{C}_{R}$ inside **AIPS** or by typing man aips $^{C}_{R}$ from the Unix command line.

Z.3.1.2.1. IBM workstation windows at the AOC

The IBMs use the Motif window manager (**mwm**). Before you begin on your assigned IBM, you should have a black screen with a login prompt. If a window system is already running, that means someone else has not completely logged out of the machine. Check the windows to see if any processes are running; if not, press **ALT**, **DELETE**, and **BACKSPACE** keys (all at the same time) to exit the window system. Then log in as

user-id **aips**, and the window system will start afresh. If you wish to use your own user account, you must make sure you are in the **aipsuser** group (type **id** $\subset_R$ and check your group listings).

You will be greeted by several windows: a cordial lawyer-ese greeting from IBM (feel free to kill the window), a window-manager screen, and a blue window which might exhibit some strange characteristics at first. If you type **stty sane** $\subset_R$, the window will behave in a much more civil manner.

The default behavior of the window manager is "focus follows mouse pointer." What this means is that in order to use one of the windows, you merely move the cursor with the mouse so that it is within the main part of the window and start typing. Click once on a border and that window will pop to the front. Most windows will have a title bar on the top. You can "grab" this title bar by moving the mouse cursor onto it and holding and keeping down the left mouse button; then moving the mouse will also move the window. You will see the outline of the window as it is being moved. A window can be resized by moving the cursor to a window corner and holding the left mouse button and "dragging" the corner. A window can be resized in width or height only by grabbing a side of the window. A window can be closed (iconified) by clicking once on the "dot" on the right side of the title bar. A window icon can be opened by moving the cursor to the icon and clicking twice rapidly with the left mouse button. Clicking once on the square to the extreme right of the title bar will enlarge the window to fill the workstation screen; clicking again will reduce it to the previous size. Clicking once (and holding) on the "dash" to the extreme left of the title bar will give a menu to move, resize, open and close the window. *Note, depending on the nature of the window, if you click on this dash* twice *in rapid succession, the window will disappear and all processes in it will die.* Don't do this unless you really mean it.

There are some shortcuts to these window functions as well; pressing **ALT** and **F3** at the same time will push the highlighted window (the one with the cursor in it) behind all the other windows. Pressing **ALT** and **F4** at the same time will close (*i.e.*, kill) a window. **ALT** and **F9** will minimize (iconify) the current window.

The left mouse button brings up a series of menus when it is pressed on the background, or root, window. The first item will open an **xterm** window well-suited for the running of *AIPS*. The second item starts an *AIPS* session; this window will ignore **CTRL-Z** commands and will disappear when **AIPS** is exited, so it might not be well suited to all applications. The next item will open an **emacs** window for editing text files. There is also an option to open a window suitable for running **OBSERVE**. Finally, one of the options will allow you to exit the windowing system.

Z.3.1.2.2. SUN workstation windows at the AOC

The AOC Suns use the **OpenWindows** system. At the login prompt, type **aips** and enter the password. You will then be asked if you wish to start **OpenWindows** — answer yes, and the window manager will begin. You will see several windows open up — a virtual desktop, a clock, a performance meter, a console window, and an **xterm**. The **xterm** is well suited for running *AIPS*, but does not allow scrolling back a large distance; opening a **command tool** enables a larger scroll range.

A whole host of **OpenWindow** options can be chosen by clicking once with the right mouse button on the background (*i.e.*, not in a window). If you go to **Programs** and drag the cursor to the right, you will see a second menu with various types of windows (xterm, command tool, shell tool), tools (mail tool, print tool, calculator, etc.) and managers (file manager, calendar manager, etc.). The next option of the main menu, **Utilities** allows you to refresh the screen or lock the terminal. Note that if you change your window setup, you should *not* **save the workspace**, since this will affect all other users of the **aips** account. Finally, you can exit **OpenWindows** by dragging down to **Exit**; note that it will not, however, log you out of the *AIPS* account.

Windows can be moved or resized in much the same manner as the IBM windows. Clicking (and holding) once on the title bar with the left mouse button allows you to drag the window to another location. Clicking once (and holding) at a window corner with the left mouse button allows resizing of a window. Each of these jobs can also be done from a menu for each window; click once on the title bar with the right mouse button to see the options.

An icon can be opened into a window by clicking on it twice in rapid succession with the left mouse button. A window can be closed by clicking once on the box to the left side of the title bar.

Z.3.1.3. Data disk management at the AOC

A public 2-Gbyte disk partition on the Auspex server (**Arana**) was set aside for use as the area where message and **SAVE/GET** files are kept. *AIPS* looks for these files only on disk "1," whichever data area that happens to be. If the same disk area is always disk 1, no matter what computer you are using, then you will always get the same set of message and **SAVE/GET** files. The public Sun workstations are set up so that the **Arana** data disk is always mounted as the first disk. The drawback is that the Network File System (NFS) has to wait while disk reads and writes are completed, while reads and writes on local disks may be done asynchronously using large memory buffers in the Unix operating system. Thus, the writing of messages, in particular, is virtually instantaneous on local disks, but is time consuming over NFS. It is for this reason that the **Arana** data disk is no longer automatically mounted on the IBM workstations; if you wish to mount the **Arana** disk, you must do so with the **da=arana** option at *AIPS* startup. If you don't want to mount the **Arana** data disk, you can create your own disk access file which excludes all external disks. Talk to your assigned *AIPS* friend or the *AIPS* Manager for more details.

The same consideration applies to disks used for image and *uv* data files. It is not too expensive to read such files over NFS, but you should *only* write data to disks on the computer you are using. You should also restrict all scratch files to be on local disks, using the adverb **BADDISK** to inhibit all NFS-mounted disks. It also helps to "spread around" your data on multi-disk machines; if you are reading files from one disk, write the output file to a different disk.

Z.3.2. Using the tape drives at the AOC

For a general discussion of magnetic tapes, including the *required* software mount, see § 2.4. The following describes how to deal with the individual tape drives at the AOC.

Z.3.2.1. Mounting and removing tapes on 9-track drives

The Sun workstation **Sol** in the second- (ground-) floor computer room has a nine-track HP drive. If there is no tape in the drive, the readout panel will read **READY**. If you want to make sure there is no tape in the drive before pushing any buttons, the top panel can be lifted (push the button on the right side of the box) and the reel inspected. Pushing the **REW/UNLD** button will open the drive. Put a write ring in the tape only if you intend to write on the tape; if you only want to read, make sure there is no write ring. Take the outer plastic rim off the tape before loading. Slide the tape in sideways with the label facing up until it sits on the central hub (it will feel like it's balanced), then close the door. The display panel will read **LOADING**, with the **ONLINE** light flashing. When the tape is ready to be accessed, the panel will read **BOT**, with the **ONLINE** indicator on (not flashing). After the display indicates that the drive is ready, you can perform the software mount from **AIPS**. As you access the tape, the panel will read **IDLE** in between operations. When you are finished with the tape, dismounting the tape in the **AIPS** window will free the tape lock and unload

the tape by opening the drive door. Please close the door after you've removed the tape to keep out dust and dirt.

Two more nine-track drives are located on the Sun workstation `Iolani`, also located in the second-floor computer room. This machine is primarily for use of the tape archivists, and the drives should only be used if urgency reigns and `Sol` is in use. Contact the data archivists (currently Theresa McBride and Gayle Rhodes) before using these drives.

Z.3.2.2. Mounting tapes on Exabyte drives at the AOC

There are several different types of Exabyte (8mm) drives at the AOC. All of the public Sun workstations have dual-density Exabyte drives, either a TTi 8501 or a COCOMP drive. The TTi 8501 drive has a window through which the tape (or lack thereof) can be seen; the COCOMP drives have a screen displaying the status of the tape. Before opening the drive to insert your tape, make sure there is not a tape present (on the COCOMP drives, the `current tape operation` reads `NotRdy`). If there is a tape present, ask around to make sure that the tape is no longer in use. Remember that the user of the drive may be in an office as much as two floors away and that Unix does not provide much protection. If you dismount a remote user's tape and mount your own, that user may well write on it, thinking that he is writing on his own tape, without knowing that he is destroying all your data.

To open the drive, push the button on the lower left of the tape door. If there was already a tape in the drive, it will be ejected after some whirring and clicking and a few seconds. If a tape is ejected, remove it. Exabyte 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. Now put your tape in the drive, label facing upwards; gently push the door closed.

It is necessary to wait until the mechanism in the drive has "settled down" before you can access the drive. On both the TTi 8501 and COCOMP drives the flashing green light will stay lit (and the whirring and clicking will stop) when the tape is ready to be accessed. The COCOMP `current tape operation` will read `IDLE`. Each drive will also show some indication of the tape remaining (or capacity); the number should be on the order of 2200 for low-density tapes and 4500 for high-density. If it is a new tape, the display will be at 4500; if you mount at low density, the display will not change to the appropriate value until you start writing. Once the green light on the drive has stopped flashing, the tape is ready to be accessed. The first step is the software `MOUNT` command from inside `AIPS`.

The IBM workstations also have varying types of Exabyte drives. Most have an internal Exabyte drive (in the cpu box) which work at low density only; they have windows allowing you to see if a tape is currently mounted. Kiowa's internal drive is dual density, but has no display (the green light will be lit or flashing quickly if a tape is being accessed). Several of the IBMs also have external Exabytes. The TTi 8501 on Hopi works as those on the Suns. Several IBMs have TTi 8505 dual-density drives (about half the height of the TTi 8501 drives); these drives don't have viewing windows (although you can push the door in *gently* to look for a tape). If there is a tape in the drive, the green light will be lit (or flashing quickly) and the `tape remaining` indicator will be non-zero. If there is no tape in the drive, the green light will either not be lit or blink very slowly. *Note:* please use only "data-grade" tapes on the IBMs.

Z.3.2.3. Mounting tapes on DAT drives at the AOC

The Falcon DAT tape (4mm) drives (manufactured by HP) at the AOC have a window which has a cover. When the cover is down and the two green lights under the door are not lit, there is no DAT tape in

In **15JUL95** and later releases of *AIPS*, **UVFLG** has been upgraded to minimize the required editing of the VLBA flag information. As a first step, the parameter **TIMEOFF** should be set to zero for each station since both flag information and data are stored with UTC times. The antennas are now identified through either the **ANTENNA** or **ANT_NAME** keywords. In the former case the file needs to be edited to insert the antenna numbers as listed in the **AN** table (use **PRTAN** on your *AIPS* file to find these). For **ANT_NAME** the antenna names can be used directly. If using keyword **ANTENNA** then replace the absolute day numbers with relative day numbers with respect to the *AIPS* reference date. This is not necessary if keyword **ANT_NAME** is used, as in the new VLBA calibration files. A new keyword **DTIMRANG** is also supported which pads each flagged time interval to insure that very short flag intervals are applied. Set **INFILE** inside **UVFLG** to point to the text file and run **UVFLG** to flag the data.

If your observation used non-VLBA antennas you will need to edit the text file manually to add any log information supplied for these antennas. In the near future, this procedure will no longer be necessary as the flagging information will be generated automatically by the VLBA correlator software and written into an **FG** table by **FITLD**. **UVFLG** can, of course, be run without an external control file, using instead the *AIPS* inputs to delete particular stations, IF channels, etc. which are thought to be bad.

Another editing task which may be useful is **QUACK**, which can edit, say, the first 30 to 60 seconds after each source change. This might be needed because (at non-VLBA antennas) the telescopes were still slewing or system temperature measurements were being made.

Task **IBLED** can be used to inspect and edit the data interactively. **IBLED** is similar in many respects to **TVFLG** (*q.v.*) but is 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 §§ 4.4 through 4.4.2 and §§ 5.5 for more details on the editing of data.

9.3. Calibration strategy

We can now begin the process of calibrating VLBI data. 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 recommended that version one of the **CL** table, as produced by **FITLD**, be copied to **CL.2** before any calibration is begun, and that **CL.2** be used as the starting point in the calibration sequence. An effort is made within *AIPS* to insure that **CL.1** is not deleted inadvertently. If this does occur however it can be re-generated using task **INDXR**.

It is also useful to use the task **SNPLT** periodically to inspect the contents of the latest **CL** table. The example below plots the antenna-based amplitude corrections stored in **CL** table 2.

> TASK 'SNPLT' ; INP ↵ — to review the inputs.

> INDISK *n* ; GETN *ctn* ↵ — to specify the input file.

> OPTYPE 'AMP' ↵ — to plot amplitudes.

3.9. Magnetic tapes

Large volumes of data are usually brought into, and taken away from, *AIPS* using magnetic tape. 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.

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 AIX or SunOS) of reserving a tape drive globally for your exclusive use, though once you have it **MOUNT**ed 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* $\subset_R$ to specify the drive labeled *m*.

4.5. Antenna-based complex gain solutions

At this point, we assume that you have removed the worst of the bad calibrator data (if any) and have run **CALIB** over as large a **UVRANGE** as possible for each calibrator. The resulting gain tables can be brought to a consistent amplitude scale, bootstrapping the unknown fluxes of the secondary calibrators. Final pass(es) of **CALIB** are done if needed and then the solution tables are merged into a full calibration (**CL**) table.

4.5.1. 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 C_R	
> SOURCES '*cal*1' , '*cal*2' , '*cal*3' ... C_R	to select secondary flux calibrators.
> CALSOU '3C286' , ' ' C_R	to specify primary flux calibrator(s).
> CALCODE ' ' C_R	to use all calibrator codes.
> BIF 1 ; EIF 2 C_R	to do both IFs.
> FREQID 1 C_R	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. 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.

4.5.2. Full calibration

Once you have determined the flux densities of all your gain calibrators, you are ready to complete the first pass of the calibration. At this point, many observers take a conservative viewpoint and delete their existing **SN** table(s) with

> INEXT 'SN' C_R	to specify the **SN** table.
> INVERS -1 C_R	to delete all versions.
> EXTDEST C_R	to do the deletion.

This step forces you to re-run **CALIB** for all your gain calibration sources and is not required if the previous bootstrapping calibrations included all antennas and most correlators, for these calibrators.

antennas, and not correlated between IFs or polarizations, then you have simply noise and/or background confusion. In tis case, do *not* edit the data — the randomness of the "errors" nearly always averages out nicely and the solution is jsut fine. Large or systematic errors indicate either that the calibrator source is resolved or that there are problems with the data requiring editing. If a calibrator is being resolved, delete the bad **SN** table and re-run **VLACALIB** with an appropriate **UVRANGE**.

4.4. Assessing the data quality and initial editing

At each stage in the data calibration process, it is a good idea to take a look at the data to determine their quality and then to "flag" (edit, delete) those that are suspect or clearly bad. Having begun the actual calibration, it is important to get an impression of the overall quality of the data and to edit out any obviously corrupted data, (*e.g.*, bad integrations that were not detected and expunged by the on-line monitoring system, high amplitudes due to interference, unstable amplitudes due to undetected equipment problems, *etc*). During the initial calibration, you need to do this only on the observations of calibration sources. However, at a later stage, you may also need to apply techniques similar to those described below to your program sources. If you do edit any calibration data at this point, you must re-run **CALIB** following the instructions given above for the affected sources.

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 **SNPLT** or **LISTR** and the calibrator data viewed with the matrix format of **LISTR** show only a few problems.

There are two 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 "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** for all baselines and times (but only one IF, one Stokes, and one spectral channel) at a time, **SPFLG** for all spectral channels and times (but only one baseline, one IF, and one Stokes) at a time, and **IBLED** for all times (but only a single baseline, channel, IF, and Stokes) at a time.

Now ask: did the image improve so much that the improvement would be noticeable even in the (line minus continuum) images? That is, would such dynamic range improvement of a (line minus continuum) image be above the noise? If the answer is "yes":

7. Use **SPLIT** to apply the same correction to all your channels.
8. Use **HORUS** to make the images (§ 10.3) which will be stored in a "cube". (You may want to use **MX** for wide field continuum imaging, or in the exceptional case that *no* continuum has to be subtracted *and* the line signal needs CLEANing. In this case, you will need to run **UVSRT** first.)

10.1.3 Keep going!

Continuing from § 10.1.1 or § 10.1.2

9. Use **ALTSWTCH** to display velocities rather than frequencies.
10. Use **TVMOVIE** to look at all your images (§ 10.6).
11. Use **OFFZOOM** and make hard copies of TV planes with sets of channel images (**TVON 1 ; TVOFF 234** $\subset_R$, **TVOFF 1 ; TVON 2** $\subset_R$, etc.)

Do you need to subtract a continuum (§ 10.7)? If so,

12. Use **SQASH** to combine sets of adjacent channels.
13. Use **COMB** to combine mean continuum images at the outer ends of the band.
14. Use **COMB** again to subtract the continuum image from the cube.
15. Use **TVMOVIE** to look at the cube with continuum subtracted.
16. Use **OFFZOOM** and make hard copies of line images.

As an important check:

17. Use **IMEAN** to calculate the noise in some line-free channels.

Did you reach the theoretical noise? If not, *think*, and go back to steps 8 or 12. Do not proceed until you understand what went wrong.

Do you need to CLEAN the line signal?

18. Use **APCLN** to CLEAN the images (if made by **HORUS** or **UVMAP**), otherwise **MX**.

Once you are satisfied with the CLEANing:

19. Use **TRANS** to change the axis order in the cube to let you look at profiles (§ 10.9).

Now you can do a variety of things (§ 10.10):

20. Use **COMB** to compute optical depth.
21. Use **BLSUM** to make integral profiles. (If profiles are all that you require, it may be possible to use **POSSM** which works on the *uv* data directly.)
22. Use **SLICE** and **TKSLICE** to look at single profiles.
23. Use **PLCUB** or **XPLOT** to look at many profiles.
24. Use **XBASL** to remove baselines.

TVFLG begins by constructing a "master grid" file of all included data. This can be a long process if you include lots of data at once. It is probably better to use the channel selection, 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 source 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. It is rather harder to undo errors if you use **TVFLG** on a single-source data set since the flagging commands are applied directly (and destructively) to the data. For this reason, we normally recommend that you use **TVFLG** on multi-source data sets, converting single-source ones with **MULTI** before running **TVFLG**.

TVFLG displays the data, for a single IF, channel, 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.

whether or not that space is reserved. The right-most column of **FREE**'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 **FREE** 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, 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.

> 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 1 ; GO DISKU ↵ to run the job

will give you the size of every one of your 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.

SCRD is run automatically whenever **EXIT**, **RESTART**, or **ABORT** *task_name* are executed. Note that the imaging and deconvolution tasks **MX**, **UVMAP**, **APCLN**, **HORUS** and **VTESS**, the data editor **TVFLG** and the sorter **UVSRT** may create large scratch files, so you should watch for "dead" copies of scratch files from these programs in your disk catalog. Both **MCAT** and **UCAT** will show scratch files as well as the requested file type. 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.

The verb **TIMDEST** destroys all user data sets that have not been used in some minimum time interval. In unmodified versions of *AIPS*, this time interval is 14 days. **TIMDEST** also deletes messages over 3 days old from all users' message files. The adverbs of **TIMDEST** allow you to request less stringent cutoffs. Your local *AIPS* Manager may set other limits on the time ranges. **TIMDEST** may take a long time to run if disk usage on your computer is not well policed. This is a design "feature" intended to promote regular use of **TIMDEST** by authorized *AIPS* Managers rather than by individual users. However, you are welcome to use it. Be aware, however, that "all" in the sentences above includes you.

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. To use these methods:

> EXIT ↵ to exit from **AIPS**, saving your **AIPS** inputs in the **LASTEXIT** area.

Then consult with your local *AIPS* Manager. Normal users should not employ system methods of disk-space creation without being fully apprised of the possible consequences.

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. If your observations are affected by antenna positional errors, you should be informed in a covering letter for your observations. This letter, issued only when there are large errors, will advise you of the affected antennas and the time range through which the antenna position corrections must be made. The appropriate corrections are in the covering letter. Since smaller errors are not reported by covering letters, you may wish to check for antenna position corrections using the local utility "**vlais**" on *zia* (**zia.aoc.nrao.edu**). This on-line utility summarizes recent observational details for the VLA and VLBA. It can be invoked by typing (at system level)

```
% vlais ↵
```

and following the menu. For information on how to get to system level, see Appendix Z. Listed are the date and time that the antenna was moved to its station, the baseline changes in meters, and the date and time at which the changes were installed. Noting these, and the time of observation, you can determine, using the relationships given above, whether you need to make corrections. If no corrections are required for your data set, go on to the next section.

To apply the antenna position corrections, first copy the first version of the **CL** table to version 2 with task **TACOP**.

> TASK 'TACOP' ↵	
> INDISK m ; GETN n ↵	to select the data set, $n = 3$ and $m = 1$ above.
> CLRONAME ↵	to copy to the input file by default.
> INEXT 'CL' ; INVERS 1 ; OUTVER 2 ↵	to copy **CL** table version 1 to version 2.
> NCOUNT 1 ↵	copy only one version.
> KEYWO ' ' ; KEYVAL 0 ; KEYSTR ' ' ↵	these adverbs not required here.
> GO	to run task **TACOP**.

This will create a higher version of the **CL** table 1 and append it to your multi-source data set. Next, use **CLCOR** to enter the antenna position corrections (in meters) in the new version of the **CL** table. This must be done for each affected antenna in turn.

> 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 ' ' ↵	to do all sources,
> BIF 0 ; EIF 0 ↵	and all IFs,
> STOKES ' ' ↵	and all Stokes.
> SUBARRAY x ↵	to choose the correct sub-array.
> OPCODE 'ANTP' ↵	to select the antenna position correction mode.
> GAINVER 2 ↵	choose the correct version of the **CL** table to modify.
> 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. 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 2 (or higher).

> 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.
> FLAGVER 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. (Note that, if you use different REASONs for your different flag entries, then you can also undo all flags with a given **REASON** using OPCODE = 'REAS' in **UVFLG**.) 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**.

4.4.2. 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 following inputs are suggested:

> TASK 'TVFLG' ; 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.
> SOURCES ' ' C_R	to select all sources.
> TIMER 0 C_R	to select all times.
> STOKES 'RRLL' C_R	to select both right and left circular polarizations; you can then toggle between RR and LL interactively.
> FREQID 3 C_R	Select **FQ** entry 3.
> BIF 1 ; EIF 2 C_R	to specify both VLA IFs; you can then toggle between the two interactively.
> ANTENNAS 0 C_R	to display data for all antennas.
> BASELINE 0 C_R	to display data for all baselines.
> DOCALIB FALSE C_R	to avoid applying calibration to raw data.
> FLAGVER 1 C_R	to use flag (FG) table 1.
> DPARM = 0 C_R	to use default initial displays and normal baseline ordering.
> DPARM(6) = 30 C_R	to declare that the input data are 30-second averages, or to have the data averaged to 30 seconds.
> DOCAT 1 C_R	to save the master grid file.
> INP C_R	to review the inputs.
> GO C_R	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 the same version of the **FG** table. To ensure this you should set **FLAGVER** to 1 and keep it that way for all runs of **TVFLG**.

```
          The default behaviour of this option if only one of tvdisp
          and tvhost is specified is:
             tv=tvhost   tvdisp defaults to tvhost.
             tv=tvdisp:    tvhost defaults to where AIPS is running.
             tv=:tvhost  tvdisp defaults to where AIPS is running.
          For the remote TV options to work, you must be able to use
          the rsh command; see the notes on it under the tp= heading
          above.

notv      Prevents automatic activation of the TV servers if no
          display is wanted.

remote    Indicates that the user is running from a terminal with
          Tektronix display capability.    Graphics output will be sent
          directly to this terminal.
```

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.)

2.3. 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 aand try to describe general characteristics of normal setups.

2.3.1. General window management

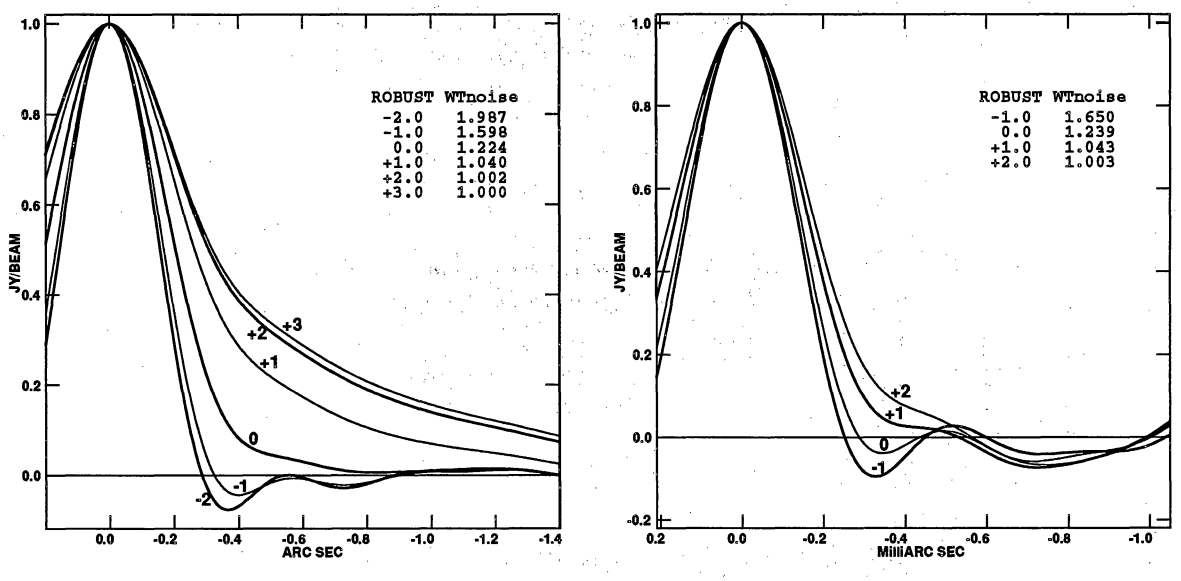
Most window managers allow multiple windows to be created on the screen at the same time. These windows can either be closed in a small "iconified" form or opened in a larger and more usable form. Windows are normally opened by positioning the cursor on the icon with the mouse and clicking either once or twice with the left button. You can type only into open windows. An open window can be resized 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).

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. **IMAGR** is new and the impact of all of these parameters on imaging is not well understood. You may wish to experiment since it *is* known — see figures below — 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.

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.



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).

9.1.2. Other image combination options

COMB may also be used to add or subtract images, to rescale them, and to compute spectral indices, optical depths, *et al.* Type:

> HELP COMB ↵ to review the available options.

At the time of writing the options include:

'SUM '	Addition	$a_1 M_1 + a_2 M_2 + a_3$
'MEAN'	Average	$a_1 M_1 + a_2 M_2$ *except* M_j *where* M_i *blanked*
'MULT'	Multiplication	$a_1 M_1 M_2 + a_2$
'DIV '	Division	$a_1 M_1 / M_2 + a_2$
'SPIX'	Spectral Index	$a_1 \ln(M_1/M_2) / \ln(\nu_1/\nu_2) + a_2$
'OPTD'	Opacity	$a_1 \ln(a_3 M_1/M_2 + a_4) + a_2$
'POLI'	RMS sum	$a_1 \sqrt{M_1^2 + M_2^2} + a_2$
'POLC'	RMS sum	$a_1 C(M_1, M_2) \sqrt{M_1^2 + M_2^2} + a_2$ where C is a noise-based correction for Ricean bias
'POLA'	Arctangent	$a_1 \tan^{-1}(M_2/M_1) + a_2$
'REAL'	Real part	$a_1 M_1 \cos(a_2 M_2) + a_3$
'IMAG'	Imaginary part	$a_1 M_1 \sin(a_2 M_2) + a_3$
'RM '	Rotation measure	$a_1 RM(M_1, M_2) + a_2$
'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 and M_1 and M_2 are the images selected by **INNAME**, *et al.* and by **IN2NAME**, *et al.*, respectively. **COMB** may also be instructed to write an image of the estimated noise in the combination rather than the direct result of the combination.

9.1.3. Considerations in image combination

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.

When combining two or more images, **COMB**, **PCNTR**, *et al.* must decide which pixels in the 2nd image go with which pixels in the 1st 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. Task **HGEOM** provides this service with up to 7th-order polynomial interpolation. See § 9.4.1 and type EXPLAIN HGEOM ↵ for more information.

> TASK 'MX' ; INP ↵ — to review your inputs (there are so many that it's easy to forget some).

> GO ↵ — to begin execution.

6.1.3. What to do while CLEAN is running

The *AIPS* monitor displays useful information about the progress of the CLEANing. The TV displays the residual images after each major cycle if you set **DOTV** to **TRUE** or **1**. You may change the image display parameters (*e.g.*, coloring, transfer function, zoom, etc.) using the display modification commands while the CLEAN is in progress to keep the residual image looking intelligible. When using the TV, **APCLN** and **MX** schedule 15-second pauses after each major cycle. The new residual image will be displayed on the TV monitor just before these pauses. Pressing button **D** on your trackball or mouse during the 15 seconds will end **APCLN/MX** in an orderly fashion at that point. Pressing any of buttons **A**, **B**, or **C** causes the execution to resume (as does waiting 15 seconds). The *AIPS* monitor prompts you for this action, which allows quasi-interactive CLEANing. We encourage use of DOTV TRUE ↵ when you are CLEANing an image for the first time. Watching the TV display as the CLEAN proceeds will help you to gauge how to set up control parameters for future CLEANs, how long to iterate for. 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.

Note that the number of CLEAN iterations, and other parameters, may be changed interactively while **APCLN** or **MX** is running by use of the *AIPS* **SHOW** and **TELL** utilities. Type SHOW MX ↵ or SHOW APCLN ↵ while the task is running to see what parameters can be reset, and their current values. Then reset as appropriate, and TELL MX ↵ or TELL APCLN ↵ to change the parameters in the running task. (The changes are written to a disk file that **MX** and **APCLN** check at appropriate stages of execution, so they may not be passed on to the program immediately — watch your *AIPS* monitor for an acknowledgement that the changes have been received, perhaps some minutes later if the iteration cycles are long or your machine is heavily loaded).

If you do not specify **BMAJ** and **BMIN**, 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 this.

When **APCLN** and **MX** terminate, a record of the CLEANed image is entered into your disk catalog and the image can then be displayed, contoured, etc. as described in §§ 6 and 7 below. The most important parameters of the CLEAN are logged in a "history" file that is cataloged as an extension of the CLEANed image file. This history file can be printed by:

> INDI *n* ; GETN *ctn* ↵ — to select the CLEAN image, where *n* and *ctn* select its disk and catalog numbers.

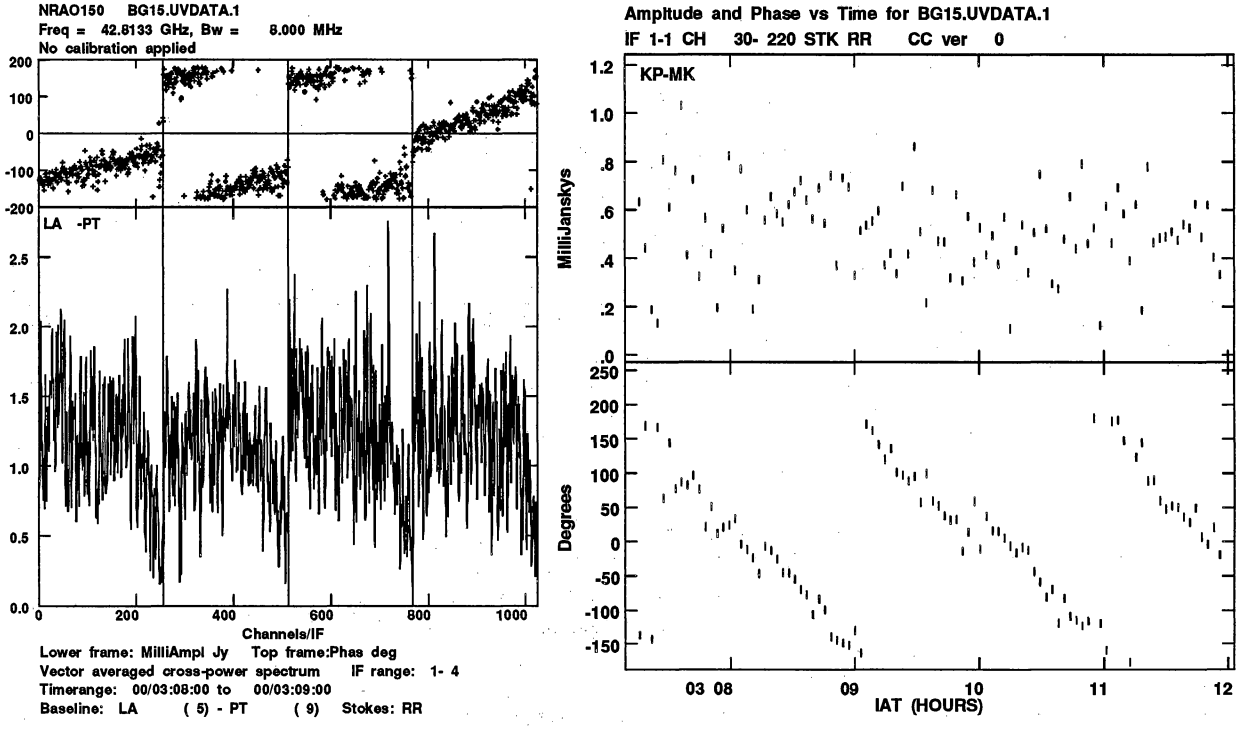
> PRTHI ↵

6.1.4. Restarting CLEANs

You can restart a CLEAN under control of the parameter **BITER** in the **APCLN** inputs, or of **BCOMP** in the **MX** inputs. **BCOMP** is an array of up to 16 values, one for each field imaged by **MX**. Set **BITER / BCOMP** equal to the number of components to be used from the previous CLEAN when CLEANing is restarted. When you are restarting a CLEAN, the **OUTNAME** and **OUTSEQ** parameters *must* be set explicitly to those of

> INTERPOL ' ' C_R	to use linear vector interpolation with no **SN** table smoothing.
> SMOTYPE ' '; INTPARM 0 C_R	to clear interpolation and smoothing parameters.
> SNVER 1 ; GAINVER 2 ; GAINUSE 3 C_R	to specify which **SN** table(s) and which input and output **CL** tables are used. If multiple runs of **FRING** were required, then set **SNVER=0** so that all **SN** tables are combined before being applied.
> REFANT 1 ; BADDISK 0 C_R	to set **REFANT** equal to the antenna number used in **FRING**.
> INP C_R	to check the inputs.
> GO C_R	to run the task.

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. It is also useful to run **POSSM** on another scan containing a strong calibrator in order to check that the assumption of constant IF phase offset holds.



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 and 9.6).

right: A **VPLOT** plot of the 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 § 9.6), the data cannot be averaged in time.

THIS PAGE DELIBERATELY LEFT BLANK.

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. If you have done extensive data editing, **UVCOP** may produce a rather smaller data set even when the whole time range is copied. Antenna, gain, and other table extension files are copied, but plot files are not.

3.8. Finding helpful information in *AIPS*

Much *AIPS* documentation can be displayed on your terminal by typing HELP *word* $\subset_R$, 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.

To print the **HELP** information on your line printer, set **DOCRT = -1** and enter EXPLAIN *word* $\subset_R$ 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 $\subset_R$	to get a list of all adverbs in the symbol table
> HELP ARRAYS $\subset_R$	to get a list of all array adverbs in the symbol table
> HELP REALS $\subset_R$	to get a list of all real adverbs in the symbol table
> HELP STRINGS $\subset_R$	to get a list of all character string adverbs in the symbol table
> HELP VERBS $\subset_R$	to get a list of all verbs, pseudoverbs, and procedures in the symbol table
> HELP PSEUDOS $\subset_R$	to get a list of all pseudo verbs in the symbol table
> HELP PROCS $\subset_R$	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 $\subset_R$	for help on **HELP**.

A few general help files remain. They are **POPSYM** (symbols used in POPS interpretive language), **WHATSNEW** (major changes in *AIPS* since the last update — notoriously incomplete unfortunately), **NEWTASK** (writing and incorporating a new task into *AIPS*), and **PANIC** (solutions to common problems). The **15JUL94** versions of these files are listed in § 15 of this *CookBook*.

The **HELP** verb is very useful, but only if you know that the function you want exists in *AIPS* and know its name. Two new functions have appeared in **AIPS** to assist you in this search. The first of these,

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* $^{C}_{R}$ (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** 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 $^{C}_{R}$ 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, data disks, and tape devices. 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 15JAN94 AIPS with normal priority

BEGIN THE ONE TRUE AIPS NUMBER *n* (release of 15JAN94) at priority 0

where **15JAN94** identifies the release of *AIPS* and n is a number between 1 and 5 (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* $^{C}_{R}$

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:

8. TAKING HARD COPY OF IMAGES

A listing of available *AIPS* tasks for two-dimensional displays and hard copies of images can be obtained at your terminal by typing HELP PL2D. The listing is also given in § 15 of the *COOKBOOK*. Representative samples of the hard-copy displays are given in § 8.9 at the end of this section.

8.1. Ordinary contour plots (CNTR)

In addition to the usual image selection parameters, you may specify the following parameters to the contouring task **CNTR**:

> TASK 'CNTR' ; INP ↵ — to tell you what you may specify.

Examples of the command syntax (*not* necessarily a recommended combination):

> BLC 250, 230 ↵ — to set the bottom left corner of plot at 250,230.

> TRC 300, 330 ↵ — to set the top right corner of plot at 300,330 (in pixels with 1,1 at extreme bottom left of the image).

> 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 § 6.3). See also § 8.7 for some additional labeling options. Then check:

> INP ↵ — to review what you have specified.

When you're satisfied with the inputs, then:

> GO ↵

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 inadvertedly 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. In addition, there may be local limits on the size of files that you can queue to the print device. Check with your local manager.

To look at the results, enter:

> GO TKPL ↵ — to display the plot file produced by **CNTR** on the Tektronix 4012 (graphics) terminal. The **HARD COPY** switch on the terminal can be used to send the TEK image to the copier (often a printer/plotter).

> ABORT TKPL ↵ — to stop an overly messy plot on the 4012.

> GO TVPL ↵ — to display the plot file on a TV graphics plane.

> GO PRTPL ↵ — to display the plot file to the printer/plotter.

1. INTRODUCTION

1.1. The NRAO *AIPS* Project — A Summary

The NRAO Astronomical Image Processing System (*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 800,000 lines of code, 80,000 lines of on-line documentation, and 400,000 lines of other documentation. It contains over 300 distinct applications "tasks," representing approximately 50 man-years of effort since 1978. The *AIPS* group in Charlottesville and Socorro has five full-time scientist/programmers, and several other computing and scientific staff with partial responsibility to the *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 about 200 non-NRAO sites that have requested copies of *AIPS*. It currently offers *AIPS* installation kits for a variety of UNIX systems, with updates available semi-annually.

In 1983, when *AIPS* was selected as the primary data reduction package for the Very Long Baseline Array (VLBA), the scope of the *AIPS* effort was expanded to embrace all stages of radio interferometric calibration, both continuum and spectral line. The *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 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.

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 *AIPS* in the IAU-standard FITS (Flexible Image Transport System) format. *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 3.8 Mbytes of "help" text that provides on-line documentation for users. There is also a suite of printed manuals for users and for programmers wishing to code their own applications "tasks" within AIPS.

An important aspect of *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. *AIPS* tries to present as nearly the same interface to the user as possible when implemented in different 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 recoding. (*AIPS* was developed in ModComp and Vax/VMS environments with Floating Point Systems array processors, but was migrated to vector pipeline machines in

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 defined time interval. Optionally, data from up to 9 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 and 9.6 for more information and a sample plot. 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.
> SOURCE '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 not apply any calibration.
> APARM 1 , 1 , 0 , 0 , -180 , 180 , 0 , 0 , 1 , 0 ↵	to control the plot as **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)=1** to plot all IFs together in one diagram.
> BIF 0 ; EIF 0 ; BCHAN 0 ; ECHAN 0 ↵	to include all IFs and spectral channels.
> CODETYPE ' ' ; POLPLOT ' ' ↵	to plot amplitude and phase.
> SOLINT 0 ↵	to average over the full time range.
> NCOUNT 9 ; BPARM 0 ; OUTFILE ' ' ↵	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.
> 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} correlation coefficients).

While **POSSM** allows us to view the data as a function of frequency, the task **VPLOT** allows us to view the visibility data as a function of time. 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, you must use task **CLPLT**. Note that **VPLOT** will plot data from a single IF and spectral channel or can coherently average the data over many spectral and IF channels. Calibration can be applied before averaging these channels. Also, if desired, a model can be plotted against the data. 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.
> SOURCE 'OQ208' ↵	to specify the source name.
> BIF 4 ↵	to give first included IF channel.

input text file, use your favorite editor to extract the system temperature information (*i.e.*, remove weather and flagging information) from the calibration file for your observation. The calibration file can be found on "aspen", and will be called *xxxx*`cal.vlba` (where *xxxx* is your experiment code). See § 9.2 for details on how to access this file. Copy it to your local computer and rename it in upper case letters. The calibration information in *xxxx*`CAL.VLBA` consists of blocks of T_{sys} numbers for each antenna as a function of time. Replace the string `TIMEOFF=*` by `TIMEOFF=0.0` everywhere. Since the VLBA and MkIII correlators generate time tags in UTC and *AIPS* now assumes by default that times are in UTC for VLBI data, `TIMEOFF` need no longer be specified.

The rows of numbers supplied in *xxxx*`CAL.VLBA` are T_{sys} values measured for different IF channels (BBCs). In order for *AIPS* to use these numbers to calibrate the data they must be associated with the corresponding IF channel number and polarization in the *AIPS* file. This assignment is achieved through the use of the INDEX command and is discussed further below.

A "gain group" must also be added to the input text file. This group contains information about the antenna zenith gains (in Degrees Kelvin per Flux Unit, DPFU) and gain curve information. For the VLBA, this information can be extracted from file `vlba_gains.key` in directory `pub` accessible by anonymous ftp to host `ftp.aoc.nrao.edu` (or `[146.88.1.4]`). This file contains gain information for all VLBA antennas, observing bands and polarizations. Since the gains change slightly over time scales of months and years, be sure to select the entries whose time range best corresponds to your observations. A separate gain entry in the text file is required for each antenna used. Extract the relevant lines from the `vlba_gains.key` file and, after modification, insert them in the text file. Note that, if your data is dual polarization, two entries for the DPFU per antenna are required (*e.g.*, `DPFU = 0.165, 0.175`). The first value will be used for RCP IF channels and the second for LCP IF channels.

If your observation uses non-VLBA antennas, including the VLA, you will have to combine the calibration files sent to you from these stations manually with the VLBA information in the text file. Some non-VLBA stations supply data in a compatible format, but, for other stations, extensive editing will be required. If you do such editing, please read the explain file carefully in order to generate the correct format. Note that you must be very careful to define the range of IFs to calibrate and ensure that you have entries for each IF in the T_{sys} and T_{ant} cards.

9.4.2.1 Amplitude calibration using ANCAL

`ANCAL` works by modifying `CL` files rather than creating new ones. Therefore, it is important to save the original `CL` table version 1 before applying this task. *Read carefully* the `HELP` and `EXPLAIN` files for `ANCAL` and follow the examples given there to ensure that the input text file has the correct format.

As discussed above, `ANCAL` assigns the measured T_{sys} values to individual *AIPS* IF channels and polarizations using the INDEX command. For instance the command INDEX = 'R1', 'L1', 'R2', 'L2' at the beginning of a block of T_{sys} information for a given antenna associates four entries for T_{sys} on each line as being the correct values to calibrate *AIPS* IF channel 1 RCP, IF channel 1 LCP, IF channel 2 RCP, and IF channel 2 LCP data, respectively. Ensure that `INDEX` associates the columns within the T_{sys} table with the corresponding IF channels in the *AIPS* files and that the assumed polarizations (RCP or LCP) for each column are also correct. The channel association can be checked by comparing the frequency information for each IF channel in the *AIPS* file (use `LISTR` with `OPTYPE = 'SCAN'`) with that at the head of the calibration text file. The polarization of each column is also specified at the head of the text file.

A "control group" is required by `ANCAL` at the beginning of the text file. This group should end with a / and must be present even if it is empty. The correlator B-factor (a factor scaling all amplitudes) is set in this control group using `BDEFAULT`. This should be unity for VLBA data if digital corrections have been applied

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 $W \propto 1/\sigma^2$. You can get this by specifying

> 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 would contribute 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 = -4 ⏎ 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

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

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

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

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. This "DDT" 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 is expected to provide the main software platform supporting radio-astronomical data processing in the latter half of the 1990's. Future 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 **aipsmail@nrao.edu** or by paper mail to the AIPS Group, National Radio Astronomy Observatory, Edgemont Road, Charlottesville, VA 22903-2475, 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 § 15 of the *CookBook*. In this edition, some sections provide an overview of a few less basic, but nonetheless interesting, programs which often seem to be forgotten even by experienced *AIPS* users.

AIPS software is changing and growing continually. This seventh edition of the *CookBook* describes the 15 January 1994 release of *AIPS*, frequently referred to as **15JAN94**. Much has changed in *AIPS* software since the previous (**15JUL90**) edition of the *CookBook*. This edition includes a chapter on the use of the *AIPS* calibration package for continuum, spectral-line, solar and VLBI data (§ 4). The index of current *AIPS* tasks (§ 15) has been updated and reflects the extensive improvement and expansion of *AIPS* software and of our ability to do the indexing. The chapter on spectral-line imaging (§ 10) has been revised substantially by Elias Brinks and Bill Junor. Phil Diamond has rewritten the chapter on VLBI imaging (§ 11). Alan Bridle has revised the chapters on *Making Images* (§ 5), *Improving Images* (§ 6) and *Displaying Images* (§ 7). 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 IBM workstations. Appendix Y deals with estimating file sizes on disk and tape.

This edition also contains the helpful glossary of astronomical and computing terms written by Fred Schwab. Suggestions for additions to the *CookBook* and reports of any inadequacies or errors in it should be made to the *AIPS* group in Charlottesville (**aipsmail@nrao.edu**).

As a result of typesetting, the *CookBook* does not exist in a form which can be listed intelligibly on ordinary line printers. Copies may be obtained by writing to the NRAO Computer Division in Charlottesville

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' ↵ 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 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 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 UseNet newsgroup **sci.astro.fits** is devoted to discussion of FITS, and there is an anonymous ftp archive of FITS information available at **fits.cv.nrao.edu**. World-wide web users (with clients like *lynx* and *NCSA Mosaic* can access the FITS home page at

http://fits.cv.nrao.edu/FITS.html

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

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

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

> INFILE '*filename*' ↵

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

> OUTFILE = '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.

Remote FITS disk files may be read in much the same manner as remote magnetic tapes. Type HELP INFILE ↵ or HELP OUTFILE ↵ 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. For VMS, change the **RECL** value to 720 longwords. 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 14 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.

TABLE OF CONTENTS

> BCHAN 0 ; ECHAN 0 C_R	to flag all channels.
> FREQID 1 C_R	to flag only the present **FQ** number.
> ANTEN 6 , 9, 22 C_R	to select the antennas.
> BASEL 0 C_R	to select all baselines to these antennas.
> STOKES ' ' C_R	to select all Stokes.
> REASON = 'ZOMBIE ANTENNA' C_R	to set a reason.
> FLAGVER 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.

4.3.2. Primary flux density calibrators

The flux densities of 3C286 (1328+307) and 3C48 (0134+329) on the scale of Baars *et al.* (Astr. & Ap., **61**, 99 (1977)) are given in the 1990 VLA Calibrator Manual as:

3C286:

$$\log S = 1.480 + 0.292 \log \nu - 0.124 (\log \nu)^2$$

3C48:

$$\log S = 2.345 + 0.071 \log \nu - 0.138 (\log \nu)^2$$

where S = flux density in Jy and ν is Frequency in MHz. These values are, at a few selected frequencies:

Frequency (MHz)	S 3C286 (Jy)	S 3C48 (Jy)
1465	14.51	15.37
1680	13.55	13.76
4885	7.41	5.36
8415	5.20	3.15
14765	3.48	1.75
15035	3.44	1.71
22485	2.53	1.09

Careful measurements made with the D array of the VLA have shown that the Baars *et al.* (1977) coefficients are in error slightly, based on the assumption that the Baars' expression for 3C295 is correct; see the 1990 VLA Calibrator Manual. Revised values of the coefficients have been derived by Rick Perley. Task **SETJY** has these formulae built into it, giving you the option (OPTYPE 'CALC') of letting it calculate the fluxes for primary calibrator sources 3C295, 3C48, 3C286, 3C147 3C138, and 1934-638. The default setting of APARM(2) = 0) will calculate the flux densities of 3C48, 3C147, and 3C286 according to the new (Perley) coefficients, while APARM(2) = 1 will calculate the flux densities using the original Baars *et al.* coefficients. **SETJY** will recognize both the 3C and IAU designations (B1950 and J2000) for these sources. You may 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.

Unfortunately, since both 3C48 and 3C286 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. 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.

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 other parameters may be set wrongly.

The final delay and rate solutions 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 and 1 for phase.
> 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' ⏎	to plot rate solutions, **OPTYPE 'DELA'** for delay solutions.
> NCOUNT 5 ⏎	to plot five antennas 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 out the solutions for **OPTYP 'RATE'**, **'DELA'** (single-band delay), and **'MDEL'** (multi-band delay). 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 **SNSMO**. If, in this task, 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. The solutions are smoothed after clipping so that deleted entries can be replaced with mean values based on a boxcar- or median-window-filter average. See the explain file for details. Since **SNSMO** modifies the **SN** table, rather than creating a new one, it is safest to copy the **SN** table to a higher version using **TACOP** and then apply **SNSMO** to this higher version. If you have a lot of discrepant **SN** values, you should also consider using option **INTERPOL = 'POLY'** in task **CLCAL** (see below).

If, for some reason, you set the parameters of **FRING** or **SNSMO** wrongly and the resulting **SN** table is unusable, it is probably 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**:

> TASK 'CLCAL' ; INP ⏎	to review the inputs.
> INDISK *n* ; GETN *ctn* ⏎	to specify the input file.
> SOURCE ' ' ⏎	to calibrate all sources.
> CALSOUR ' ' ⏎	to use all solutions without regard for which sources they were determined.
> OPCODE 'CALI' ⏎	to generate a **CL** table from **SN** tables.

AIPS COOKBOOK

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 132 C_R	to route the output to terminal.
> 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.

9.6. Fringe fitting

Even after removing instrumental phase offsets from each IF (see § 9.5), 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.

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). 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 plots are illustrated on the previous page.

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 in order to correct for delay and rate effects. In § 9.6.1 we discuss in detail the use of **FRING** and **CLCAL**. An alternative to **FRING** is to use the tasks **BLING** and **BLAPP**; these are discussed briefly in § 9.6.2.

Note that the process of fringe fitting, and then interpolating the solutions using **CLCAL**, is a very time consuming process. Although it depends a lot on the size and structure of the data set, a Sun IPX can take a time equal to, or longer than, the observing time to fringe fit a large data set. For this reason it is probably wise to run through the fringe fitting procedure described in § 9.6.1 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 avoids the possible confusion of having many versions of extension tables.

9.6.1 Fringe fitting using FRING

Suitable inputs for the fringe-fitting task **FRING** are:

> TASK 'FRING' ; INP CR	to review the inputs.
> INDISK *n* ; GETN *ctn* CR	to specify the input file.
> CALSOUR ' ' CR	to find solutions for all sources.
> TIMER 0 CR	to find solutions for all times.
> DOCALIB 1 CR	to apply the most complete calibration file including amplitude calibration and IF and channel phase offsets.
> GAINUSE 3 CR	to use **CL** table 3; use **CL** table 2 if the supplied phase-cal data aligned the IF phases.
> SNVER 2 CR	to write solutions into **SN** table 2.
> BCHAN 0; ECHAN 0 CR	to use all spectral channels within each IF channel.
> SMODEL 0 CR	to use a point-source at the origin model for the sources.

> TASK 'IMAGR'; INP $\subset_R$	to set the task name and see what to specify.
> INDI *m*; GETN *n* $\subset_R$	to select the desired *uv* database.
> STOKES 'I' $\subset_R$	to construct a total intensity image — use 'Q', 'U', 'L', 'R', etc. for other Stokes parameters one at a time.
> DOCALIB FALSE $\subset_R$	to apply no calibration.
> IMSIZE 1024 $\subset_R$	to make a square image 1024 pixels on each side.
> CELLSIZ 1 $\subset_R$	for 1 arc-second cells.
> UVTAP *xtap ytap* $\subset_R$	to specify the widths to 30% of the Gaussian taper in x and y in kλ (kilo-wavelengths).

Other inputs are defaulted sensibly, which is why we started with a **RESTORE 0**. 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). Consider both:

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

> HELP *xxx* $\subset_R$

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 $\subset_R$

then:

> GO $\subset_R$ to run **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. 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 8192, *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 $\subset_R$

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    .IMAP    .    1 MA 22-MAR-1995 13:50:10
 43   76 3C138 A C    .IBEAM   .    1 MA 22-MAR-1995 13:59:58
```

Note that the default beam class is **.IBEAM**; the default image class will be **.IMAP**. The images produced with **NITER = 0** by **IMAGR** can be deconvolved by various image-plane methods (§ 5.3.6).

A much more general geometric transformation is performed by **HGEOM**, which converts 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. **HGEOM** should be used before comparing images (with **COMB**, **GREYS**, **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 HGEOM ↵ to obtain the details and useful advice.

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 § 9.4.4. However, **PGEOM** can also "de-project" elliptical objects to correct for their inclination and "unwrap" spiral objects. Type EXPLAIN PGEOM ↵ for information.

9.4.2. 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 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 **GREYS** / **QMSPL** combination (see § 9.4.4). Enter EXPLAIN NINER ↵ for additional information.

MWFLT, at present, applies any one of four nonlinear, 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. The others 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 last 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.

9.4.3. Modeling

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 **IMMOD** and **UVMOD** will add (or subtract) a point, Gaussian, disk, or rectangular source to the (scaled) input image or *uv* data, respectively. Both tasks can also add noise and both allow the original data to be replaced by the model. Type EXPLAIN IMMOD ; EXPLAIN UVMOD ↵ for details.

The task **CCMOD** will create a clean-components file representing the chosen Gaussian or 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 small (< 2000 visibility) *uv* data sets.

10. SPECTRAL-LINE SOFTWARE

This chapter deals with the manipulation of *calibrated* spectral line data. *Calibration* of spectral line data is discussed in § 3.

Spectral-line software generally involves three-dimensional images, often called "cubes", in which the third dimension is frequency or velocity. Special programs are needed to build and transpose these cubes and to display them properly. Much of the continuum software will work on data cubes or on appropriate two-dimensional images from a data cube. Spectral-line *uv* data can be read into *AIPS* from "*uv*" FITS tapes. Often, however, the data already exist on disk having been calibrated first within *AIPS*.

This section is organized as follows. First, a "flow-diagram" is given of how a typical spectral-line data set could be processed in *AIPS*. Second, input examples are given for many of the programs. Finally, some more advanced profile analysis programs are discussed. Some aspects of the art of spectral-line imaging are discussed in Chapters 17 and 18 of *Synthesis Imaging in Radio Astronomy* *.

10.1. Data reduction strategies

Usually the spectral line data have been calibrated in *AIPS* (see § 3) and the *uv* data reside on disk as two files containing the so-called "Channel 0" data and the line data. "Channel 0" data is formed from the scalar average of the inner 75% of the line database. The first step after calibration is to inspect the Channel 0 data or a line channel which has a very strong signal present and to decide if *self-calibration* is possible.

Self-calibration can improve the "dynamic range" of your maps 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. Consult Lecture 9 of *Synthesis Imaging in Radio Astronomy* to decide if your data meet the criteria required for *self-calibration.*

10.1.1. Self-calibrating data from the *AIPS* calibration package

If your data have followed the conventional calibration route through the *AIPS* calibration package, read on. Otherwise go to § 10.1.2.

Starting from *uv* data

1. If your data warrant *self-calibration*, first use **HORUS** to make an image from the Channel 0 data or a selected line channel. **HORUS** will construct an image and a "dirty beam" (§ 3).
2. Next use **APCLN** to *clean* the dirty image and generate a set of clean components which can be used as the starting model for *self-calibration.*

* *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).

> UVRANGE = x_1, x_2 C_R — 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 C_R — 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 C_R — to select the reference antenna; for best results, choose one known to be good over most of the time range.

> SOLMODE 'A&P' C_R — to solve for amplitude and phase corrections simultaneously.

> SOLMODE 'P' C_R — to solve for phase weighted by amplitude, the default for single-source files.

> SOLMODE 'P!A' C_R — to solve for phase ignoring amplitude.

> SOLTYP ' ' C_R — to use a normal (non-linear) least squares solution.

> SOLTYP 'L1' C_R — 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' C_R — 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 $w_1, w_2, w_3, \ldots$ C_R — to apply additional weights to each antenna (in order) in generating the solutions; 0 implies 1.

> APARM(1) = x_5 C_R — to reject solutions from fewer than x_5 antennas; default is 6.

> APARM(2) = x_6 C_R — to tell **CALIB** whether the data have already been divided by a model ($x_6 > 0$) or not.

> APARM(3) = x_7 C_R — 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 C_R — 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 C_R — to set the level of diagnostic information as 0 (very little), 1 (some including time and S/N ratio), or 2 (too much).

> APARM(7) = x_{10} C_R — to discard solutions having S/N ratios $< x_{10}$; default is 5.

> SOLINT = x_{11} C_R — to set the length of the solution interval (in minutes); default is 10 seconds for single-source files.

> CPARM(2) = 1 C_R — 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.

> CPARM(3) = y_1 C_R — to display the values of amplitude errors which are $> y_1$ %.

> CPARM(4) = y_2 C_R — to display the values of closure phase errors which are $> y_2$ degrees.

> CPARM(5) = 1 C_R — 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.

> STOKES 'I' ↵	to image total intensity.
> FREQID 1 ↵	to select **FQ** value to image.
> BIF 1 ; EIF 0 ↵	to image all IFs — multi-channel mode images only one IF.
> BCHAN *n* ; ECHAN *m* ↵	to combine a range of channels.
> CHINC 1 ↵	to set channel increment.
> DOCALIB TRUE ↵	to apply calibration.
> GAINUSE 0 ↵	to use highest numbered **CL** table.
> FLAGVER 1 ↵	to edit data.
> DOPOL TRUE ↵	to correct for feed polarization.
> DOBAND 1 ↵	to correct bandpass.
> BPVER 1 ↵	to select **BP** table to apply.
> OUTNAME 'CAL:' ↵	to select the 1st four characters of the output file name. The remaining eight are the source name.
> OUTDISK 0 ↵	to use any output disk with enough space.
> IMSIZE 512 512 ↵	to set the size in cells of image.
> CELLSIZE 0.25 , 0.25 ↵	to set the size of each image cell in arc-seconds.
> SHIFT 0 0 ↵	to (not) shift image center.
> UVWTFN ' ' ↵	to use uniform weighting.
> UVBOX 0 ↵	to do uniform weighting over a single grid cell.
> ZEROSP 0 ↵	to introduce no zero-spacing flux.
> XTYPE 0 ; YTYPE 0 ↵	to use the default convolving function.
> XPARM 0 ; YPARM 0 ↵	to use default convolving-function parameters (important!).
> INP ↵	to review the inputs.
> GO ↵	to run the program when inputs set correctly.

4.10. Additional recipe

4.10.1. 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.

The right-most column has only the option:

EXIT Go 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.

4.4.3. Baseline corrections

Sometimes, *e.g.*, during an 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, 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. The maximum phase error in degrees, $\Delta\phi_G$, caused by incorrect gridding of the u,v,w data is

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

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 reverse the flag all IFs status; applies to subsequent flag commands

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.

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 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"

The tape will be rewound if necessary and will then begin to move forward. All files will be read. A printout will appear on the system printer with a valuable summary of header information for each file on the data tape. The printout may be routed, instead, to your terminal by specifying DOCRT 1 C_R before running **PRTTP** or it may be saved in a disk file by setting **OUTPRINT** (see § 3.10.1).

As **PRTTP** starts executing, look for the message **PRTTP BEGINS** on the *AIPS* "monitor" (the **MSG_SERVER** window on your workstation or, in its absence, your own *AIPS* window or some nearby terminal on antique systems). If you can see the active tape drive from your terminal, look also for movement of the tape. The **AIPS** prompt > should have already returned on your terminal, however, since **PRTTP** is running as a detached "task." As described in Chapter 3, tasks are the more complicated *AIPS* programs, run by the **GO** command after setting the task name with TASK *'taskname'*. They are shed from the terminal, as **PRTTP** has been here, allowing you to use **AIPS** for further processing (except running the same task at the same time).

The file at which the tape is currently positioned can also be "indexed" by the **AIPS** verb **TPHEAD**. This verb, which also works on FITS-disk files (§ 3.10.3), displays the data header to let you decide if you are pointing at the desired data file.

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 the third and fourth files on a tape mounted on tape unit number 2:

> TASK 'FITLD' ; INP C_R	to set the task name and review the required inputs.
> INTAPE 2 C_R	to specify the tape drive number; the tape must already be mounted as in § 3.9.
> NFILES 2 C_R	to skip to the third file on the tape.
> CLRONAME C_R	to use the file names on the tape.
> OUTDISK 3 C_R	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 C_R	to use compressed *uv* disk format to save space.
> NCOUNT 2 C_R	to read 2 consecutive tape files.
> OPTYPE 'UV' C_R	to restrict reading to *uv* files.
> REWIND C_R	to rewind tape before skipping files.
> GO C_R	to run **FITLD**..

The tape will begin to move and appropriate messages should appear on the *AIPS* monitor. When the prompt > appears on your terminal, you are free to use **AIPS** for other purposes.

FITLD may also be used to read a FITS-disk file as:

> INFILE 'FITS:*filename* C_R	to read the FITS-disk file in the public area known as **$FITS** of name *filename*.
> GO C_R	to run **FITLD** with the adverbs set above — **NFILES** and **NCOUNT** are ignored when **INFILE** is not blank.

See § 3.10.3 for a discussion of FITS disk files.

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 C_R for details.

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 more than 140[1] 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 including Apollo, Data General, Hewlett Packard, IBM, MassComp, Nord, Silicon Graphics, Stellar and SUN products. It is available for use on 80386 and 80486 personal computers under the public-domain Linux operating system. 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 in 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 and film recorders. 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 that is sent to over 775 *AIPS* users outside the NRAO soon after each semi-annual "release" of new *AIPS* code. There is also a mechanism whereby users can report software bugs or suggestions to the *AIPS* programmers and receive written responses to them; this provides a formal route for user feedback to the *AIPS* programmers and for the 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" `http://info.cv.nrao.edu/aips/aips-home.html`). The NRAO knows of

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

In looking over the output from **LISTR**, 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. Beginning with the **15JUL94** release, you may reset a calibrator code to blank by specifying **CALCODE = '----'**.

4.3. Beginning the calibration

After loading the data to disk, it has been traditional to begin with a substantial session of data checking and editing. With data from the VLA, this is always time consuming and often not necessary. Nonetheless, it is probably a good idea to check for two spcific kinds of problems before beginning the actual calibration. These are corrupted data in the first record of most scans and totally dead antennas. Many other problems in the data are quickly and easily diagnosed by carefully inspecting the solution tables produced from the calibrators on un-edited data. Missing antennas and erratic amplitudes due to sampling problems and RF interference can be spotted from the **SN** tables and the closure-error messages produced by **CALIB**. If you *can't* spot errors from these, you may not need to edit the calibrator data. If the **SN** tables have well-behaved phases for most antennas and rapidly rotating phases for one or two, then you may need to apply baseline corrections rather than editing. See § 4.4.3 for details of how to make antenna-position corrections.

The next section tells how to detect simple problems in the data and eliminate them to reduce the warnings from the calibration tasks. The following sections tell you how to enter fluxes for the primary calibrator sources and do a preliminary calibration for all calibrators. In so doing, you should generate one or more solution (**SN**) tables containing the complex gains at the times of the calibration observations. These tables may be examined for problems with the observations. If you find problems, then you need to edit the data or apply baseline corrections and should consult § 4.4. If you do not find problems, you may proceed directly to § 4.5. (Of course, you may decide to edit the data from your program sources at a later stage of the data reduction and have to return to § 4.4 then.)

4.3.1. Initial editing

The warning messages from the calibrations described in the next sections may be reduced by flagging those antennas which were not actually working, but which were not flagged by the on-line system. Another problem that has plagued the VLA (and other interferometers) persistently is that the first record in scans can be corrupted; usually its amplitudes are lower than they should be. These data can be flagged using **TVFLG** or **UVFLG**, but this can be time consuming. We recommend that you do the following before beginning your regular data editing. Use the task **LISTR** on your terminal (to save time and paper) to see if you have the problem:

> TASK 'LISTR' ↵ to set the data listing task

> INDI n ; GETN m ↵ to select the data set, $n = 3$ and $m = 1$ above.

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 tape 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 'UVMAP' ↵	to examine only history information from **UVMAP**.
> DOCRT *m* ↵	to direct the display to your terminal of width *m* characters.
> PRTHI ↵	to print the **UVMAP** history.
> 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 some selectivity 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 parameters ("adverbs") are global throughout **AIPS**. When a adverb value is specified for any task or verb, it remains at that value for any other task or verb that uses an input adverb of the same name (until you change it). This global nature of the AIPS input 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.

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

using the task. Chapters 8 and 15 of the NRAO Summer School on *Synthesis Imaging In Radio Astronomy* also provide useful general background.

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 and, if needed, call Tim Cornwell at the NRAO's Socorro office before using them.

6.2. Self-calibration

The task **CALIB** is a tool for obtaining images with high dynamic range when there is sufficient signal to noise in the *uv* data. (**CALIB** contains all the functionality of the older *AIPS* self-calibration task **ASCAL** with some extra features. We encourage you to use **CALIB** rather than **ASCAL** as the latter's development is now frozen.) It does this by comparing the input *uv* data set with the predictions of a source model in order to compute a set of antenna-based amplitude and phase corrections which would bring the data into better agreement with model, as a function of time throughout the observations. 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. The input model may be either a point source (parameters set by **SMODEL**) in **CALIB** or a set of CLEAN components derived from a previous image (parameters set by **IN2DI**, **IN2NAME**, **IN2CLASS**, **IN2SEQ**, **INVERS**, **NCOMP** and **NMAPS**).

Do not use **CALIB** unless your data have enough signal-to-noise to warrant improvement. Ask yourself whether your externally-calibrated CLEAN images contain unCLEANable 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!

6.2.1. Programs to run

If you decide to use **CALIB**, 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. *N.B.,* for large data sets, use **XINC** to reduce the number of points plotted to no more than a few thousand; otherwise it will take too long to make and plot the plot file.
2. Use **TKPL** and the hard-copy switch on your TEK terminal, or **PRTPL** or **QMSPL**, to get hard copy of the above plot file.
3. 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.
4. 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 either (**HORUS** + **APCLN**) or (**UVMAP** + **APCLN**) or **MX**. Note that you may want to use a 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.
5. Use **PRTCC** or **TAPLT** (as in the example above) to help you decide how many components from this CLEAN to include in the **CALIB** model; when you have decided this,

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
TVFIDDLE	interactive zoom, black & and white enhancement, and pseudo-color contours as in **AIPS**
TVTRANSF	black & white enhancement as in **AIPS**
TVPSEUDO	many pseudo-colorings as in **AIPS**
DO WEDGE ?	switches choice of displaying a step wedge
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

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.

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^{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 **LOAD**s 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 an asterisk following. 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
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 length in units of the master grid cell size

are lost in using **FXPOL** so, if needed, this task must be run immediately after **FITLD**. The **CL** and **CQ** tables can be re-generated using **INDXR** and **FXVLB** respectively.

In case (ii), LL data may simply be mis-labeled as RR. 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 (§ 9.4). The Stokes axis can however be modified using **PUTHEAD**.

9.1.3 Data examination

Before proceeding further it is important to examine the data, to make sure it is all loaded, and (especially if this is the first time you have reduced VLBI data) to familiarize yourself with the data structure. As processing of the data continues it is also important to inspect the data periodically to check on the progress of the calibration process.

As a first step, the task **LISTR** can be used 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. 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.

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.
3. **PRTAN** provides a listing of the antennas and their associated index 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.

Your data file will probably contain a number of channels, observed at different frequencies, corresponding to the separate "IF channels" used during the observations. While the VLA has a maximum of 4 such IF channels, the VLBA has up to 8 IFs (and effectively 16 if double sideband recording was used). MkIII Mode A observations can produce up to 28 IF channels. Within each IF channel are a number of equally spaced and contiguous "spectral channels" generated from each IF data stream by the correlator. For continuum applications, the correlator will generally produce 8 or 16 such spectral channels; in the spectral line case there will be a much larger number. Use **IMHEAD** to find the number of IF channels and the number of spectral channels per IF channel, or examine the **LISTR** 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

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 offerred 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 iconified form. Its icon shows a cute drawing of an ape along with words like **New Xas** and **AIPS TV**. In many ways, this is a normal window which can be resized, 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 positioned 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 two, of grey-scale memories and four 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. 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 239, 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. Type HELP XAS $\subset_R$ when in **AIPS** to see how to control the number of levels, the colors of the graphics overlay planes, and numerous other parameters.

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.
> INFILE 'MYVLB:BC25CAL.VLBA' ↵	to specify the text file.
> SUBARRAY 1 ↵	select subarray one.
> TYVER 1 ; GCVER 1 ↵	write to **TY.1** and **GC.1**.
> BLVER 1 ↵	write any specified baseline factors to **BL.1**.
> PRTLEV 1 ↵	select print level.
> GO ↵	to run the program.

Note that **PRTLEV=1** will echo the calibration file as it is processed, which can be useful in locating format errors.

The **TY** table can be examined using **SNPLT** with **OPCODE='TSYS'** or **'TANT'**, and the **GC** table using **PRTAB**. **APCAL** can then be used to derive an amplitude calibration (**SN**) table. Typical inputs to **APCAL** are:

> TASK 'APCAL' ; INP ↵	to review the inputs.
> INDISK *n* ; GETN *ctn* ↵	to specify the input file.
> ANTENNAS 0 ; SUBARRAY 0 ↵	default selection adverbs.
> SOURCES ' ' ; STOKES ' ' ↵	
> BIF 1 ; EIF 0 ; FREQID 1 ↵	
> TIMERANG 0 ; OPCODE ' ' ↵	no opacity solution required.
> TYVER 1 ; GCVER 1 ↵	read **TY.1** and **GC.1**.
> SNVER 2 ↵	write amplitude solution to **SN.2**
> GO ↵	to run the program.

Information concerning opacity corrections can be found in the **APCAL** help file. The resulting solution (**SN**) table can be smoothed using **SNSMO**, and primary amplitude calibrators selected when applying the table using **CLCAL**.

9.5 IF channel phase offsets and single-band delays

After carrying out amplitude calibration, the remaining calibration steps are involved with calibrating the phase of the data within and between the separate IF channels and between the different integration periods in order to the integrate the data over both frequency and time without loss of coherence.

If you run **POSSM** on a short ($\approx$ 1 minute) section of data on a strong calibrator using the inputs described in § 9.1 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 video converter units). The VLBA and MkIII systems should inject narrow band signals ("phase-cals") into the data recorded at each antenna from which the IF channel phase offsets and, for VLBA antennas only, the instrumental single-band delays can be determined. Data produced by a MkIII correlator are provided with the phase offsets in version 1 of the **CL** table (see § 9.12). At some point the VLBA correlator will write the phase-cal information to a **PC** table. However, at present (August 1995), the phase-cal information is only available through a text file and is read into the **PC** table using the task **PCLOD**. Type **EXPLAIN PCLOD** for further information. VLBA pulse-calibration data loaded into a **PC** table in this manner can be used by

> 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 on your terminal 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 *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) `15JUL94` list printed in § 15 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. Enter

> HELP MSGSRV C_R — for details.

In the Charlottesville and most non-NRAO *AIPS* installations, you get a message server by default. The AOC has chosen — wrongly — to make the default be no message server. 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 C_R.

You may review the contents of the message file by typing PRTMSG C_R 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:

> BIF 2 ↵ to view the second ("BD") IF.

> WAIT ; GO ↵ to wait for the first display and then run the second.

The matrix of scan-averaged averaged right minus left phase angles (actually RL and conjugate of LR polarizations) will be printed. Check that none of the phases differ from the mean by more than a few degrees. If any do, then use **UVFLG** to edit these data and go back to step 1. After the matrix of phases, the average over the matrix of the right minus left phases is displayed. This is the number to be used in step 4.

This method will fail if the calibrator source (3C286 or 3C138, usually) is heavily resolved and the atmospheric phase stability is poor. (These two are frequently coupled!) Under these conditions, the self-calibration of the calibrator will have failed and will have to be done especially for the polarization calibration. In the steps below, you may safely relax the *uv* limits by about 20%, but should solve only for phases using **SOLMODE = 'P'**. The process consists of:

1. Apply **CALIB** to the inner (short-baseline) antennas on the calibrator source using the rules in the table found in § 4.3.3 but relaxed a bit. Set DOCALIB = 1 ; GAINUSE = 2 ; SOLMODE = 'P'.
2. Use **CLCAL** to apply these solutions to the calibrator source using GAINVER = 2 ; GAINUSE = 3.
3. Run **LISTR** for cross-hand phases using *only* the antennas used with **CALIB**.
4. Use **EXTDEST** to delete **CL** table 3, a most important step. After correcting the calibration, repeat steps 1 and 2 and the special calibration until satisfactory results are obtained.

Step 3: 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 is done incorrectly. (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. The notes below also assume that we make a copy of the current **CL** table, so as not to destroy it. Thus,

> TASK 'TACOP' ↵

> CLRO ↵ To use the default output file, which is the input file.

> INEXT 'CL' ↵ To copy **CL** table.

> INVER 2 ↵ Copy table 2.

> NCOUNT 1 ↵ Copy only one table.

> OUTVER 3 ↵ Copy to 3.

> INP ↵ to review the inputs.

> GO ↵ to run the program when inputs set correctly.

Step 4: The right minus left phase offset corrections are made using task **CLCOR**. The phase offset correction is the expected value (66 degrees for 3C286 or -18 for 3C138 (at L band, perhaps -24 at higher frequencies) minus the observed phases from step 2.

> TASK 'CLCOR' ↵

> SOURCE ' ' ; ANTENNAS 0 ↵ to correct all sources and all antennas.

> TIMERANG 0 ↵ to correct all times.

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 *xxxxxxx*	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	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.

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 TIME	To flag all baselines for a specific time
FLAG BASELINE	To flag all times 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

The last operation allows you to apply a clipping method already used on one channel/IF to other channels and/or IFs. **CLIP BY FORM** 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.

25. Use **XGAUS** to fit Gaussians.

26. Use **BLANK** to filter out noise in many different ways.

27. Use **XSMTH** to Hanning smooth.

28. Use **CONVL** to smooth spatially.

29. Use **XMOM** to calculate moments.

30. Use **MOMNT** to smooth, blank and calculate moments in one go.

31. Use **GAL** to fit a rotation curve.

32. Use **TV3COLOR**, **TVHUEINT**, **TVSLV** to have fun with TV displays of the data.

33. Use **KNTR** to plot all the channel images in one plot.

10.2. Reading in multichannel *uv* data

UVFITS tapes written from the now-defunct VLA Pipeline system contain files with 8 frequency channels at a time. Other systems (*e.g.*, *ISIS*) may produce files with other numbers of channels. (Any number can be handled in *AIPS*.)

As a VLA-specific example, consider a spectral-line data set with 31 frequency channels. The **UVFITS** tape contains 2 files: the first, channel 0 and the second channels 1 to 31. Suppose that the source is N315. To read both files:

> TASK 'UVLOD' ; INP ↵	to review the inputs.
> OUTCL ' ' ; OUTN ' ' ↵	to use default outname and outclass.
> POINTS 1 ↵	to make it different from zero.
> DOWAIT TRUE ↵	to have **AIPS** wait for each **UVLOD** to finish before starting the next one.
> FOR I = 1 TO 2; PRINT I ; GO ; END ↵	to run **UVLOD** twice.

Now check the headers with **IMHEAD**. This gives a lot of information, but the line giving the frequency differentiates the files. For example, if you centered at 1400 MHz and used a total bandwidth of 3.125 MHz and a channel spacing of 97.656 kHz, you should see the following:

```
TYPE    PIXELS   COORD VALUE  AT PIXEL    COORD INCR   ROTAT
FREQ         1  1.4000000E+09     1.00 2.3400000E+06    0.00
FREQ        31  1.4000000E+09    16.00 9.7656000E+03    0.00
```

for the two files loaded. The listing shows the number of channels (**PIXELS**) there are in the file, the frequency at the reference pixel (which can be outside the file), and the channel spacing (**COORD INCR**).

10.3. Making spectral-line images

There are three programs you can use, **HORUS**, **UVMAP** and **MX**; examples for all three are given below. In most cases, **HORUS** will be the preferred choice since it can run on both multi- and single-source databases and on TB- as well as XY-sorted data. Selection adverbs in **HORUS** allow the user to work on the required sections of data. **HORUS** can also apply a CL (calibration) table and BP (bandpass) table if required. **HORUS** produces two "cubes" — one containing "dirty beam" images and the other containing images for each line

```
XASERVERS: Starting TEKSRV on monkey, DISPLAY :0
XASERVERS: Starting MSGSRV on monkey, DISPLAY :0
XASERVERS: Starting XAS on monkey, DISPLAY :0
XAS: *** Using shared memory option for speed ***
XAS: Using screen width height 1142 800, max grey level 189
```

Each of the first three 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 iconified form. 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: 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** file for these three windows. After **AIPS** begins, type HELP XAS $\subset_R$, HELP MSGSRV $\subset_R$, and HELP TEKSRV $\subset_R$ for details. Among these is a parameter controlling how many colors **XAS** tries to use. 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 $\subset_R$. This information is reproduced in part below:

```
Use:    Start AIPS itself from the command line.    For Unix systems, the
      following usage applies:

         aips [old, new, or tst]
              [da=host[,host,...]]
              [pr=lpdev]
              [tp=tphost[,tphost,...]]
              [tv=[tvdisp][:][tvhost]  or  notv]
              [remote] [debug] [local]

        For VMS systems, the "=" options are not available.    The last
        release to support VMS was 15APR91.    Note that most of the
        information here is also available in the Unix MAN page for AIPS
        which in turn is found in the SYSUNIX area as "AIPS.L".

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

Options:  All command line options are case insensitive.

      The DA=, PR=, TP=, TV=, and NOTV options are new to the 15APR92
      release of AIPS.  They will not be recognized properly by older
      versions.

      AIPS allows up to three versions to co-exist (disk space
      permitting) in one installation.    They are identified either by
      date (e.g.  15JUL93) or name (old, new, or tst).    On most
      installations, these will all be the same.
```

After your image has been displayed on the TV by **TVALL**, your trackball or mouse and its buttons (which should be labeled **A**, **B**, **C**, and **D**) can be used to modify the display transfer functions, coloring and zoom. (For displays with a 3-button trackball or mouse (*e.g.*, the IVAS), *AIPS* defines button **D** to be **any two** of the three buttons **A**, **B** and **C**.)

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**.

The I^{2}S Model 70 display can show images up to 512 by 512 in size. If the image is larger than about 424 pixels 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). Other television systems will have comparable, but not necessarily identical, numeric limits. For example, the IVAS displays used at the NRAO can handle images of up to 1024 pixels on a side. A useful technique for displaying large images is to load only alternate pixels. The command:

> TXINC 2 ; TYINC 2 C_R to load every other x and y pixel.

before **TVALL** would do this. Also use:

> TBLC = *n1, n2, n3, ...* C_R bottom left pixel to load.

> TTRC = *m1, m2, n3, ...* C_R 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 C_R.

You will find that the TV display does not respond instantaneously to the changes in the transfer function requested by adjustment of the cursor, especially when the computer is being used heavily. It's best to move the trackball or mouse slowly.

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, 2, 4 and 8 are available on the I^{2}S Model 70. The IVAS displays have factors 1 through 16.

Pressing button **D** will exit from the **TVALL** display modification routine, leaving your display parameters as they were when button **D** was pressed. 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. (For image displays controlled by only three buttons, button **D** corresponds to pushing **any** two of the three buttons simultaneously.)

You may continue to modify the image display transfer function without reloading the image by typing:

> TVFIDDLE C_R

and you may continue to modify the zoom by typing:

> TVZOOM C_R

7.2.1. Alternative color coding

To obtain many alternative color codings of the TV image type:

9.9. Verifying amplitude calibration

Before proceeding to map your data, it is very worthwhile to check that the amplitude calibration performed in § 9.4 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 **IBLED** 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 (in particular non-VLBA stations) is out by a constant factor, either due to uncertainties in the antenna gain or in the noise calibration signal.

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 25,000 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** and **AVSPC**:

> TASK 'UVFIT' ; INP CR	to review the inputs.
> INDISK *n* ; GETN *ctn* CR	to specify the input *uv* data set.
> OPCODE 'GAUS' ; NGAUS 1 CR	to specify 1 Gaussian component.
> GMAX 1.2 ; DOMAX FALSE CR	to fix the flux at 1.2 Jy.
> GPOS 0 ; DOPOS FALSE CR	to hold the position fixed at the origin.
> GWIDTH 0.002 , 0.001 , 45. CR	to provide an initial guess of the Gaussian widths as 2 × 1 mas at a position angle of 45°.
> DOWIDTH TRUE CR	to fit for size.
> GAINERR 1 CR	to fit for all antenna gains with initial guess = 1.
> NITER 50 CR	to limit the fitting to 50 iterations.
> IMSIZE 0.0005 , 0.01 CR	to limit sizes to be in the range 0.5 to 10 mas.
> DOCAT TRUE ; INVER 1 CR	to save the solution in a **CC** file of version number 1.
> INP CR	to check the inputs.
> GO CR	to run **UVFIT**.

UVFIT can also be applied to the multi-source file, but **SOURCE**, **DOCAL**, **GAINUSE** etc. must then be set.

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 is 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' CR and **CLCORPRM(1)** set to the amplitude scale factor found in **UVFIT** squared; 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.

Another way of incorporating amplitude corrections is to edit the calibration text files used by **ANCAL**. This can be accomplished by setting the **FT** parameters for affected stations. For instance, if the Bonn scale

9.3.3. Gaussian fits to slices (SLFIT)

The task **SLFIT** fits Gaussian components to one-dimensional data in slice files (see § 8.5). 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^{th} file for analysis.
> TKSLICE ↵	to plot the slice on the graphics terminal.
> EDROP 840 ; BDROP 700 ↵	to select a subsection to fit.
> TKSLICE ↵	to replot just the subsection.
> NGAUS 2 ↵	to fit 2 Gaussians.

> TKSET ↵

This verb will prompt you to **POSITION CURSOR AT CENTER & HEIGHT OF GAUSSIAN COMP 1** and will turn on the Tektronix crosshairs. Move the crosshairs with the thumbwheels to the requested position and press any key *except* ↵ on the Tektronix 4012 keyboard. The terminal then requests you to **POSITION CURSOR AT HALFWIDTH OF GAUSSIAN COMP 1** and turns the crosshairs back on. Move the crosshairs until they are at the half-intensity point of the component and press any key except ↵. Continue until all components have been entered. Then type:

> TKAGUESS ↵	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 on the *AIPS* monitor, 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 on the graphics terminal, enter:

> TKSLICE ↵	to replot the slice.
> TKAMODEL ↵	to add the model results to the plot.
> TKARESID ↵	to add the residuals (data – model) to the plot.

To get a higher quality plot of the results, an example of which is shown in § 8.9, type:

> DORES TRUE ; DOMOD TRUE ↵	to request the model and the residuals.
> 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.
> GO PRTPL ↵	to display it on the printer/plotter.

9.3.4. Other one-dimensional Gaussian fits (XGAUS)

XGAUS is an interactive task which can fit up to four Gaussians and a linear baseline to each row of an image. It 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 § 10.10), it has a wide variety of other applications. The interaction is optional and uses both the terminal and the graphics screen (Tektronix 4012 or TEK). The data, initial guess, model fit, and the residual for each row may be plotted on the TEK 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 TEK crosshairs. This process is similar to that of **TKSET** described above (§ 9.3.3). This task has too many options to do them justice here. Enter EXPLAIN XGAUS ↵ for details.

Fourier transform of a set of Clean components from visibility data. You may then use **UVPLT** to display the residual *uv* dataset 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 of course run **UVSUB** again after doing the flagging or clipping, to add the transform of the Clean components back into the remaining data (using the input FACTOR = −1.0). Note that **IMAGR**'s workfile is also a *uv* dataset from which the current Clean component model has been subtracted. It may also be used with **UVPLT** to help you to diagnose problems.

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.)

TVFLG, **SPFLG**, and **IBLED** are TV-based, interactive editors. **TVFLG** is most suitable for datasets 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 § 4.4.2 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 § 8.1. **IBLED** has a different philosophy; it plots one baseline at a time in a graphical rather than gray-scale (image) fashion. This is obviously more useful for small arrays — *e.g.*, VLBI, MERLIN, the Australia Telescope. This task is described below.

5.5.2 Baseline-based *uv*-data editing — IBLED

Data sets with small numbers of baselines can be edited effectively with **IBLED** — the Interactive **B**ase**L**ine **ED**itor. This task will handle either multi-source or single-source files. To run it, enter:

> TASK 'IBLED' ; INP ↵	to select the task and review the inputs.
> INDI *n1* ; GETN *ctn1* ↵	to select the 'TB' sorted *uv* data base.
> DOCAT *true* ↵	to save the *uv* master file for future use; like **TVFLG**, editing can be done over several sessions before the flags are actually applied to the data.
> IN2SEQ 0 ↵	to create a new *uv* master file.
> DOCAL FALSE ↵	to apply no calibration. The **SN** table from previous self-calibrations can be applied.
> DPARM 0 ; DPARM(5) = Δt ↵	to have the data averaged over a time interval Δt seconds. The task will attempt to find an appropriate time interval if **DPARM(5) = 0**, but it often does a poor job. *Set this parameter.*
> 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) and can choose to edit all the IFs separately, selecting them interactively, or average all IFs together. If you do some averaging over channel, IF, and/or time, **IBLED** will compute a decorrelation coefficient defined to be the vector average amplitude divided by the scalar average amplitude. Low values of this parameter indicate poor phase agreement between the samples averaged and can be a powerful tool in editing.

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.

You specify the multiple-field information with:

> NFIELD n C_R	to make images of n fields.
> IMSIZE i,j C_R	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 $i_1, j_1, i_2, j_2, i_3, j_3, \ldots$ C_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 later.)
> RASHIFT $x_1, x_2, x_3, \ldots$ C_R	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 $y_1, y_2, y_3, \ldots$ C_R	to specify the y shift of each field center from the tangent point; $y_n > 0$ shifts the field center to the North (up).

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.

The **OUTCLASS** of the fields is controlled by **IMAGR** with no user assistance. For dirty images it is **IMAP** for the first field, **IMAP 1** for the second, and so forth. The **I** is replaced by **Q**, **U**, *et al.* for polarized images and the **MAP** is replaced by **CLN** when Cleaning.

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). 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

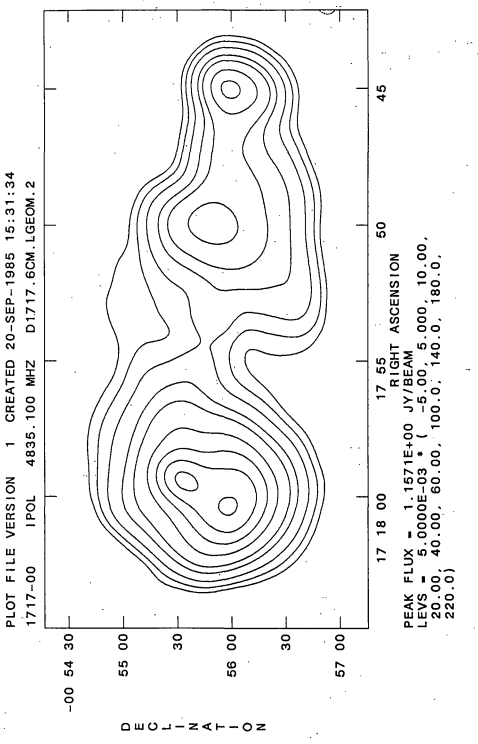
$$x = \cos\delta \sin\Delta\alpha$$

$$y = \cos\delta_0 \sin\delta - \sin\delta_0 cos\delta \cos\Delta\alpha$$

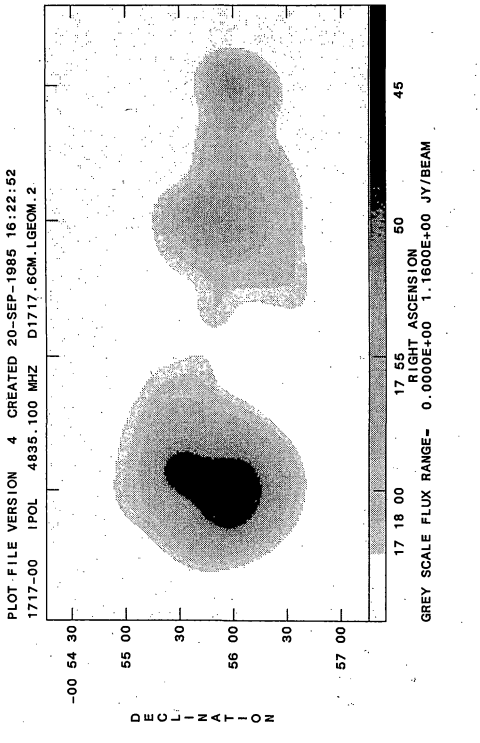
where $\Delta\alpha = \alpha - \alpha_0$ and the coordinates with subscript "0" are those of the tangent point that serves as the origin of the image coordinate system.

For many practical purposes, it is sufficiently accurate to suppose that imaging parameters such as position shifts (*e.g.*, **RASHIFT** above) 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 spcified input pixel. This is rarely necessary, however.

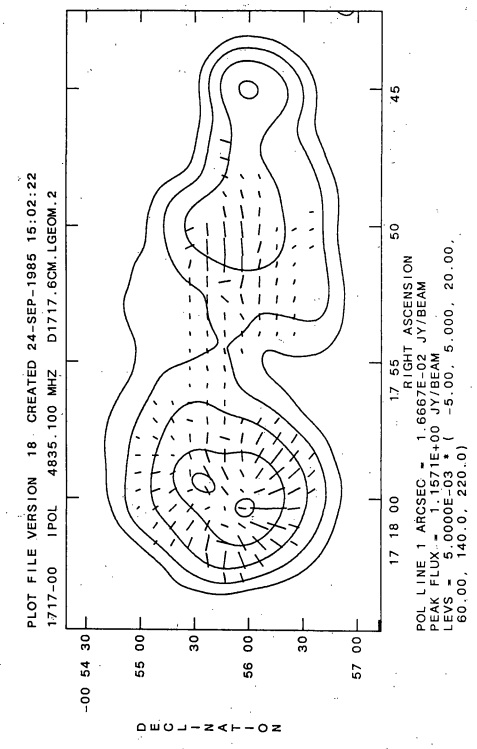
8.9. Sample displays



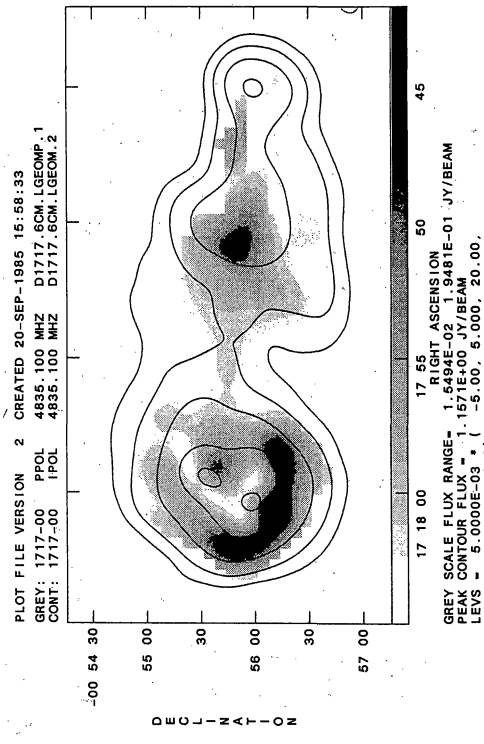
CNTR



GREYS



PCNTR



GREYS with contours

3C48, 3C147, 3C138:

Band	UVRANGE	Array	No. ant. per arm	Notes
90cm	0- 40	All	All	
20cm	0- 40	A	7	
	"	B,C,D	All	
6cm	0- 40	A	3	
	"	B,C,D	All	
3.6cm	0- 40	A	2	
	"	B	6	
	"	C,D	All	
2cm	0- 40	A	1	Not recommended
	"	B	4	
	"	C,D	All	
1.3cm	0- 40	A	1	Not recommended
	"	B	3	
	"	C,D	All	

3C286:

Band	UVRANGE	Array	No. ant. per arm	Notes
90cm	0- 18	A	7	
	"	B,C,D	All	
20cm	0- 18	A	4	
	"	B,C,D	All	
	90-180	A	All	Reduce flux 6%
6cm	0- 25	A	1	Not recommended
	"	B	4	
	"	C,D	All	
	150-300	A	All	Reduce flux 2%
3.6cm	50-300	A	3	Reduce flux 1%
	"	B	7	Reduce flux 1%
	"	C	All	Reduce flux 1%
	0- 15	D	All	
2cm	0-150	A	3	
	"	B,C,D	All	
1.3cm	0-185	A	2	
	"	B	7	
	"	C,D	All	

5.4.1. Self-calibration sequence and SCMAP

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*.) *N.B.*, for large data sets, use **XINC** to reduce the number of points plotted to no more than a few thousand; otherwise it will take too long to make and plot the plot file. 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.
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.
5. Use **PRTCC** or **TAPLT** (as in the example in § 5.3.5) 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. 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.
8. Use **SNPLT** on the input data file with **DOTV = TRUE** to review the gain corrections before proceeding further. To take hard copy for future reference, run **SNPLT** with **DOTV = FALSE** and then run **LWPLA** of 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 **GAIN**s and higher **NITER** to produce deeper and more careful Cleans are appropriate as the self-calibration progresses.

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.

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. For spectral line work, where **CALIB**'s companion program **ASCOR** may be used for interpolation of the antenna/IF gains, it is also preferable to select a stable antenna with a full duty cycle as the reference.

Note that **CALIB** should almost always be run with **SOLMODE** set to its default (selecting 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 will edit out bad data according to the following criteria:

1. if there are insufficient antennas (**APARM(1)**) to form a solution
2. the solution does not converge
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 — see § 6.2.3.

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.*

6.3. Editing *uv* data

There are many programs which aid in the processing, display, and editing of *uv* data. A summary of this software is listed on your terminal by:

> HELP UVPR ↵

and is also in § 14 of the *COOKBOOK*. In particular, there are facilities in **CALIB**, **CLIP**, and **CORER** to flag *uv* data in *AIPS* based on deviations from specified norms. There is also a task, **UVFLG**, which allows flagging and unflagging by antenna-IF or by correlator. Type HELP CALIB ↵, HELP CLIP ↵, or HELP UVFLG ↵ for details. The task **UVPLT** plots various combinations of *uv* data — type HELP UVPLT ↵ for details. The task **UVFND** is also recommended for printing out suspicious portions of the data base. Note that **CLIP** examines the data correlator by correlator, but **UVFND** normally converts the data to Stokes

Thanks to the United Fresh Fruit and Vegetable Association.

6.5. IBLED

Data sets with small numbers of baselines can be edited effectively with **IBLED** — the **I**nteractive **B**ase**L**ine **ED**itor. This task will handle either multi-source or single-source files.

6.5.1 An example

For example, to plot and edit all data for baselines with antenna 6 in the following single-source, single-polarisation (LL), single-IF, database, use:

> INDI *n1* ; GETN *ctn1* ↵	to select the 'TB' sorted *uv* data base.
> DOCAT *true* ↵	catalog the *uv* work file for future use.
> IN2SEQ 0 ↵	
> SOURCES ' ' ↵	if this is a single-source file.
> STOKES ' ' ↵	plot all data *i.e.*,'LL' fringes.
> CALCODE ' ' ↵	
> BCHAN 0; ECHAN 0 ↵	
> TIMERANG 0 ↵	plot all suitable data.
> SELBAND=-1 ↵	
> SELFREQ=-1 ↵	
> FREQID=-1 ↵	these three adverbs not needed for single-source files.
> ANTENNA 6; BASELINE 0 ↵	plot all baselines with antenna # 6.
> SUBARRAY 0 ↵	
> UVRANG 0 ↵	
> SOLINT 0 ↵	no averaging, use the raw integration period.
> DPARM 1 0 0 0 16/60 ↵	plot phases — this can be changed interactively.
> INPUTS ↵	
> GO ↵	if all is well.

If you are unsure what a parameter does, just leave it at it's default value. *AIPS* usually does something sensible. Note that the basic integration period in minutes must be set with DPARM(5). (In this case, the integration period was 16 seconds so DPARM(5) = 16/60.)

First, **IBLED** will find the data matching your selection criteria and then grid the data (with optional averaging) to a *uv* work file called *INNAME*.**IBLEDR**.*n* where *n* is the lowest unique sequence number. After a little while, **IBLED** will display the first baseline on your chosen TV device. When **IBLED** starts up, the display will comprise two "windows". The main window in the center of the screen will show a subsection of your data (or possibly all if the program can fit it into the available area). At the top of the screen, a small window — the global window — will display all of the selected data for that baseline with the subsection used for the main display delimited by two vertical lines. On either side of the main window are the menu panels. At the bottom of the screen is listed current information for your editing session *e.g.*, the Stokes' flag mask (in this case, "0111"). Should the main window not contain data that requires editing, you may select another sub-section of your data by placing the cursor over the **SELECT FRAME** option on the left hand

> REFANT *m* ↵	to select the reference antenna; needed only if **REFANT** reset since **CALIB** was run.
> INTERP '2PT' ↵	for linear interpolation, no smoothing, *or*
> INTERP 'BOX' ↵	for boxcar smoothing, then interpolation.
> INTPARM *n* ↵	for *n*-hr smoothing if **BOX** selected.
> 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.

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. **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 task **FILLM**, currently puts no special 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. (All past versions of **AIPS** have allowed you to do this, but, beginning with the **15JUL94** release, **AIPS** will ask for special confirmation before allowing you 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 will not contain any on-line calibration, however. If you have made baseline corrections — or any of the many other sorts of corrections allowed by **CLCOR** — then you will probably want to protect (and use for input) **CL** table version 2 as well. In that case, GAINVER = 3 is recommended.

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. **LISTR** with OPTYPE = 'GAIN' will print out the amplitudes and phases in the specified **CL** or **SN** table. Note that these tables can be vary large. Use the **SOURCES** and **TIMERANG** adverbs to limit the output, or look at it on your terminal (DOCRT = 132) so that you can stop the display whenever you have had enough. Task **SNPLT** will provide you with a graphical display which may be easier on the eye.

The most important step in the calibration is your verification that everything has gone according to plan. To check this, you should produce matrix listings for all your calibrator sources. For simplicity in interpretation, limit each listing to the **UVRANGE** to which you limited the calibrator during calibration. Thus:

> TASK 'LISTR' ↵	
> DOCRT -1 ↵	to direct output to the printer.
> SOURCES '*cal*1' , '*cal*2' , '*cal*3' , ... ↵	to list all selected calibrators by name.
> UVRANGE *uvmin uvmax* ↵	*uv* limits, if any, in kiloλ.
> OPTYP 'MATX' ↵	to get the matrix form of listing.
> DOCALIB TRUE ↵	to list with calibration applied.
> GAINUSE 2 ↵	or 3, to point to the *new* gain table.
> FREQID *n* ↵	list data for **FQ** *n*.
> DPARM = 5 , 1 , 0 ↵	to have amplitude and phase using scalar scan averaging.
> BIF 1 ↵	to specify the "AC" IFs; only one can be done at a time.
> INP ↵	to review the inputs.
> GO ↵	to run the program when inputs set correctly.

`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 is required by *AIPS*. 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.

3.11. Additional recipes

3.11.1. 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.

3.11.2. Banana daiquiri

1. Combine in an electric blender: 2 oz. `light rum`, 0.5 oz. `banana liqueur`, 0.5 oz. `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.

3.11.3. 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.

Later on, you might want to replace some of these images by CLEANed images. *E.g.*, assume that you have CLEANed channels 10 to 20. These images got the class **ICLN**. 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 can then be done as follows:

> TASK 'MCUBE' ; INP ↵	to review the inputs.
> INNA 'N315' ; INCLA 'ICLN' ↵	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 'LMVCUB' ↵	to select the existing cube.
> OUTSE 1 ↵	
> GO ↵	to run **MCUBE**.

10.5. 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 labeling.

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.

10.6. Displaying the cube

The easiest way to look at the cube is by using **TVMOVIE**. This verb loads sub-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, etc. To restart the movie, type:

> REMOVIE ↵

Once you are done, type:

7. READING AND DISPLAYING IMAGES

Many images to be displayed inside *AIPS* are cataloged there as soon as they are generated, by an *AIPS* imaging or deconvolution task. Images that are generated by other imaging systems, (*e.g.*, images from non-NRAO radio telescopes or non-radio images) can be transported to *AIPS* by writing them out of the other imaging system on tape or to disk in the standard FITS format. Section 7.1 below describes how to get external images into your *AIPS* disk catalog. Sections 7.2 to 7.6 outline the most commonly-used *AIPS* TV display options.

Several indices of the *AIPS* TV software can be listed on your terminal. HELP CURSOR ↵ lists the interactive cursor functions for TVs (and Tektronix graphics devices). HELP TVGEN ↵ lists the general TV loading and display functions. HELP TVINTER ↵ lists interactive TV functions (loading, transfer functions, zooming, scrolling, blinking and roaming) and HELP TVCOLOR ↵ lists verbs and tasks that control the color functions. The **15JUL90** versions of these indices are reproduced in § 15 of this *COOKBOOK*.

7.1. Getting image data into your *AIPS* catalog

Your data are already there if they have been freshly created by an *AIPS* task. Go directly to § 7.2 if so. This Section is for people whose images are initially outside the *AIPS* system on tape or in a FITS disk file.

The task **IMLOD** is used to load images to your *AIPS* disk catalog. It can read images from tape in either the standard FITS format (which has one image per tape file) or in an old Charlotteville IBM image format (which may have more than one image per file). It can also read images from non-*AIPS* disk files if they have been written in the standard FITS format.

7.1.1 IMLOD from tape

IMLOD will assume that your tape is positioned at the beginning of the first image file that you want to read. First, mount your tape in hardware and software as described in § 2.4 and § 2.5. To move the tape forward by *nf* file marks until it is positioned at the first interesting image, enter:

> INTAPE *n* ↵ to specify the tape drive labeled *n*.

> NFILES *nf* ↵ to specify the number of file marks to move the tape.

> AVFILE ↵ to move the tape.

If $nf > 0$, **AVFILE** will advance the tape the specified number of file marks. If $nf = 0$, the tape is moved backward to the beginning of the current file. Once you have moved part-way into a tape, you may use $nf < 0$ to move backwards to the abs(nf) previous file. In all cases, the tape is left at the beginning of a file. If you happen to come across a CV-IBM format tape from some astronomical museum, the verb **AVMAP** may also be needed. Type HELP AVMAP ↵ for details.

To check that the tape is positioned where you expected, type:

> TPHEAD ↵

Your terminal will then list information about the image header at which the tape is positioned. The tape position is not altered. Once the tape has been positioned at the desired image, enter:

> OUTDI *n* ↵ to specify writing the image to your *AIPS* catalog on disk *n*

```
 28   76 1200+519    .ICLN    .   1 MA 02-NOV-1993 00:35:20 WRIT
 31   76 SCRATCH FILE.MX1     .     1 SC 02-NOV-1993 00:35:37 WRIT
 32   76 SCRATCH FILE.MX1     .     2 SC 02-NOV-1993 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-1993 12:56:50
 36   76 1200+519    .UVXY   .   1 UV 02-NOV-1993 00:32:50 READ
 37   76 1200+519    .UVWORK .   1 UV 02-NOV-1993 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 **MX** *uv* work file on disk 3. There are two scratch (temporary) files on disk 2 which were created by **MX** running out of **AIPS1** (this determines their **MX1** 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* filenames 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* $^{C}_{R}$ 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 and even a third set of input image name adverbs. For these, use:

> IN2D *n2* ; GET2N *ctn2* $^{C}_{R}$ to set the second set, and

> IN3D *n3* ; GET3N *ctn3* $^{C}_{R}$ to set the third 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 **MX** on the sorted *uv* data set in slot 36 — this *uv* data file is opened for **READ**ing. The Clean image file, **ICLN** in slot 28, and the scratch and **UVWORK** files are opened for **WRIT**ing. 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

> 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

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

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

The wider you can make your window display, the more information *AIPS* can put on a line. We recommend 132 for most *AIPS* tasks and verbs, whenever possible.

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** from **AIPS**. (The time limit for "old" messages is set by your local *AIPS* Manager. Usually, it is about 3 days.)

3.3. You *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.

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**,

> CAT ↵ 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**. The **INNAME** and **INCLASS** 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   .IMAP   .    1 MA 27-OCT-1993 22:30:18
 19   76 3C166L5OK   .IBEAM  .    1 MA 27-OCT-1993 23:02:14
 22   76 3C166L5OK   .IMAP   .    2 MA 28-OCT-1993 15:30:45
CATALOG ON DISK  2
CAT USID MAPNAME      CLASS   SEQ  PT     LAST ACCESS      STAT
 22   76 1200+519    .IMAP   .    1 MA 01-NOV-1993 23:50:10
 23   76 1200+519    .IBEAM  .    1 MA 01-NOV-1993 23:59:58
 24   76 1200+519    .QMAP   .    1 MA 28-OCT-1993 00:10:10
 25   76 1200+519    .UMAP   .    1 MA 28-OCT-1993 00:19:19
```

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 'IMAP' ; 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. **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* dataset in *AIPS* has an associated header file that contains information needed to describe the dataset 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 **HI**story file described below, the **CC** or CLEAN component files (see § 6) and the **PL**ot files and **SL**ice files (see § 8).

Multi-source *uv* data files may have many extensions (see § 4). The most important are the **HI**story file, the **AN**tennas 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. § 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
> QHEAD ↵	for a shorter listing.

frequency specified. In order to run either **UVMAP** or **MX** on source data in a multi-source database, it is necessary first to extract the data with **SPLIT**, applying the calibration, and writing a single-source file. The data must be in XY sort order for **MX** or **UVMAP**; sorting is performed with **UVSRT**.

UVMAP can make images of up to 8 frequency channels in a file. It does the gridding once, for the first channel in the file. It produces one beam for the first channel. **MX** does the gridding separately for each channel and produces beams for all channels. It puts the channel images of 1 *uv* file into a cube. In most cases, you will want to subtract a continuum before you do any CLEANing, so we won't discuss the simultaneous imaging and cleaning option in **MX** in this section.

To produce a cube of spectral line images with **HORUS**, follow the example below (only the spectral line specific adverbs are listed):

> TASK 'HORUS'; INP C_R	to review the inputs
> INDI *n*; GETN *ctn* C_R	to select the multi-source file
> SOURCE 'N315' C_R	to select the target source
> STOKES ' ' C_R	to make a total intensity image
> OPTYPE 'LINE' C_R	(Use **'SUM'** for Channel 0)
> BCHAN 8; ECHAN 31 C_R	to select the range of channels to map
> DOCALIB 1 C_R	apply the calibration
> GAINUSE 2 C_R	use CL table #2
> FLAGVER 1 C_R	use flagging table #1
> DOBAND 1 C_R	apply bandpass correction (DOBAND false) for Channel 0 data)
> BPVER 1 C_R	use bandpass table #1
> OUTNAME ' ' C_R	default output name

This will produce two cubes — **N315.IBEAM** and **N315.IMAP** each containing 24 channels. If the range in frequency is small compared to the central frequency, it is not necessary to keep all the beams. **SUBIM** can be used to extract a few beams across the band. The **IBEAM** cube then can be deleted.

The inputs in **UVMAP** that are spectral-line specific are, for example, to image channel 0 (one image and one beam):

> TASK 'UVMAP' ; INP C_R	to review the inputs.
> INNA 'N315' C_R	to select the *uv* data file.
> INSEQ 1 C_R	to select the first file.
> STOKES ' ' C_R	to make a total intensity image.
> NMAPS 1 C_R	to make only 1 image.
> CHANNEL 1 C_R	to start at the first frequency in the file.
> OUTN 'N315C0' C_R	to give channel 0 a special name.
> GO UVMAP C_R	to run **UVMAP**.

To image channels 1 to 7 and make one beam (channel 1):

> INSEQ 2 C_R	to select the file containing the line data.
> STOKES 'LINE' C_R	to make more images at different frequencies.
> NMAPS 7 C_R	to make 7 images.
> CHANNEL 1 C_R	to start at the first frequency in the file.
> OUTN ' ' C_R	to use the default outname.
> GO UVMAP C_R	to run **UVMAP**.

5.3.6. Image-plane deconvolution methods — APCLN, SDCLN, VTESS

The previous sections have described the new 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 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 $^{C}_{R}$ 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-resolution" 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 $^{C}_{R}$ before using

5.2. Basic image making — IMAGR

AIPS has several imaging tasks, each with distinctive capabilities. The older tasks **UVMAP**, **MX**, **WFCLN**, and **HORUS** will not be described here since it is our belief that they have all been superceded by **IMAGR**. See their help files if you wish to use them. 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. data weighting options far more general than those in any other task and including all those used in previous tasks.
4. data imaging in up to 16 simultaneous fields, each up to 8096x8096 in size.
5. 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.
6. correction of Clean components for various wide-field and wide-bandwidth effects.
7. truly interactive TV display of residual images allowing you to alter the areas over which Clean components are sought.

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**.

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 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 § 5.3.6. The following example assumes the use of an already calibrated, single-source data set:

```
> RESTORE 0 CR                    to set all adverbs to null values.
```

* *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)

$\subset_R$ 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** 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 $\subset_R$

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

> STORE 1 $\subset_R$ to save your inputs in area 1, and

> RESTORE 1 $\subset_R$ to recover the inputs you previously stored in area 1.

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

> TPUT *name* $\subset_R$ where *name* is the verb or task name.

and retrieved by the command:

> TGET *name* $\subset_R$

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."

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

> VERSION '*version*' $\subset_R$ 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 § 14.3. Note that toggling between different versions of *AIPS* is possible only when the data formats for each release are the same. Fortunately, changes of format occur only rather infrequently. 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 15 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 **FREE** verb within **AIPS** to show you the space used and available on all disks for your session and also to show

> LEVS -3 , -1 , 1 3 10 30 100 C_R	to plot contours at -15, -5, 5, 15, 50, 150, and 500 mJy/beam.
> DOWEDGE 2 C_R	to plot a grey-scale step-wedge along the right-hand edge; 1 for along the bottom and 0 for no wedge.
> INP C_R	to review the inputs.
> GO C_R	to make the plot file.

When **GREYS** has finished, run **QMSPL** to view the plot file. Note that **QMSPL** has a variety of options which control the plotting and scaling of the grey-scale images without having to rerun **GREYS**.

8.4. Deleting unwanted plot files (EXTDEST)

Plot files generated by **CNTR**, **PCNTR**, **GREYS** and other plot tasks are archived in your disk catalog as **PL** extensions to the image file from which they were derived. Running **CNTR** a second time or some other plot task does not overwrite the previous plot file, but makes another with a higher "version" number.

You can review the parameters of the plot files associated with a given image file by typing:

> INDI *n* ; GETN *ctn* C_R	where *n* and *ctn* select the disk and catalog number of the desired image.
> INEXT 'PL' ; EXTLIST C_R	to select a listing of plot file contents.

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 C_R	to review the inputs required.
> INDI *n* ; GETN *ctn* C_R	where *n* and *ctn* select the disk and catalog numbers of the image file.
> INEXT 'PL' ; INVERS *m* C_R	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 C_R	to do the deletion.
> INVERS 0 C_R	to reset the version number to its default — usually advisable.

8.5. SLICE files (profile plots)

You can generate a one-dimensional slice (profile) through any two-dimensional plane 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**. The set of software dealing with slice file analysis and display can be obtained on your terminal by typing HELP SL1D. The list is also given in § 14.

To generate a slice:

> TASK 'SLICE' ; INP C_R	reviews 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:

> TVSLICE C_R

The values of **UVRANGE** for each secondary calibrator may be determined from the VLA Calibrator manual or by using **UVPLT** to plot the amplitudes as a function of baseline length. Since the latter works correctly only after a complete calibration has been done, it is often reasonable to use the 3C286/3C48 restrictions for all calibrator sources (at this stage). If your secondary calibrators are point sources over most baselines, then it may save you time to do the full calibration now. Not only will it save you, possibly, from re-running **CALIB** at a later time with a wider **UVRANGE**, but it will provide information on the data quality from the longer baselines.

Once you have read in procedure **VLACALIB**, you may invoke **CALIB** as follows:

> INDI *n* ; GETN *m* C_R	to select the data set, $n = 3$ and $m = 1$ above.
> CALSOUR = '3C286' , ' ' C_R	to name primary flux calibrator(s) individually, or
> CALCODE = '*' C_R	to select any non-blank calibrator code.
> UVRANGE *uvmin uvmax* C_R	*uv* limits, if any, in kiloλ.
> ANTENNAS *list of antennas* C_R	antennas to use for the solutions, see discussion above.
> REFANT *n* C_R	reference antenna number — use a reliable antenna located near the center of the array.
> MINAMPER 10 C_R	display warning if baseline disagrees in amplitude by more than 10% from the model.
> MINPHERR 10 C_R	display warning if baseline disagrees by more than 10° of phase from the model.
> FREQID 1 C_R	use **FQ** number 1.
> INP VLACALIB C_R	to review inputs.
> VLACALIB C_R	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 several times. Attached to your input data set is a solution **SN** table. Each run of **CALIB** writes in this table, for the times of the included calibration scans, the solutions for both the "AC" and "BD" 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.) It is possible to write multiple **SN** tables by using **CALIB** directly with SNVER = 1. Later programs such as **GETJY** and **CLCAL** will merge all **SN** tables which they find (if told to do so). Since this does pose some bookkeeping problems, the procedure **VLACALIB** always uses only **SN** table 1.

The **LISTR** outputs provided by **VLACALIB** should be examined carefully to check on the calibration; amplitudes should be consistent (both among antennas and among time stamps) and phases should vary smoothly. If you decide that the solutions are not acceptable (*e.g.*, there are no valid solutions) *and* you are creating a new **SN** table on each run of **CALIB**, then delete that **SN** table using **EXTDEST** before proceeding. The later stages of processing assume that all extant **SN** tables are valid. Note that re-running **CALIB** on the same **SN** table simply over-writes the old solutions with new ones. **CALIB** gives messages which indicate the number of valid and invalid solutions which should help you evaluate the results. If **VLACALIB** is run using the values of **MINAMPER** and **MINPHERR** shown above, it will print a list of baselines and times which show substantial "closure" errors. It is important to remember that normal thermal noise and, at longer wavelengths, background confusion cause closure errors too. Thus, some closure error on weaker calibrators is to be expected and may be ignored. Interpreting closure errors is a real art, but a couple of gereralizations are possible. If the same closure error shows up in both polarizations and both IFs, then you have probably got a resolved object. If one antenna dominates the closure list, especially if it is at only one IF and/or one polarization, then you have got a bad antenna. If the errors are uniformly small, distributed amongst all

> APARM(8) 0 CR	to set the maximum number of antennas; used only if no **AN** table is present.
> DPARM(1) 1 CR	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. See Chapter 19 in *Synthesis Imaging in Radio Astronomy*, 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(2) 0 CR	to set 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.
> DPARM(3) 0 CR	to set 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.
> DPARM(4) 2 CR	to specify the correlator integration time in seconds; use **DTSUM** to find the correct value.
> DPARM(5) 0 CR	to do both the coarse and the least squares solutions; set to 1 if you require only FFT solutions.
> DPARM(6) 1 CR	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 CR	to re-reference solutions to a common reference antenna; when processing polarization data, set this to 1 to avoid the re-referencing.
> INP CR	to check the inputs.
> GO CR	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.

Note that the **SOLINT** interval chosen must be less than an atmospheric coherence time, but long enough that high (> 7) 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, consult an expert or experiment with different length solution intervals on short sections of data.

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 arc-seconds). If you are using multi-baseline searching (*i.e.*, **DPARM(1)** > 0, 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.

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 ≥ the threshold set in

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 continuum calibration for connected-element interferometers, especially the VLA. However, the information in these sections is useful to spectral-line, solar, and VLBI observers as well. For additional advice on spectral-line calibration, see § 4.7; for advice on calibrating observations of the Sun, see § 4.8; and for the gory details of VLBI, read Chapter 11. 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 § 6.2 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. These are

1. **AN** (antennas) for sub-array geometric data, date, frequency, polarization information, *etc.*,
2. **FQ** (frequency) for frequency offsets of the different IFs (IF pairs in VLA nomenclature),
3. **NX** (index) to assist rapid access to the data,
4. **SU** (source) for source specific information such as name, position, velocity, and
5. **TY** (temperature) for measured system temperatures.

A null **CL** table is also created at this time. 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

6. **BL** (baseline) for baseline-, or correlator-, dependent corrections,
7. **BP** (bandpass) for bandpass calibration,
8. **CL** (calibration) for calibration and model information,
9. **FG** (flag) for flagging (editing) information, and
10. **SN** (solution) for gain solutions from the calibration routines.

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

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

5.3.4.4. 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.

5.3.4.5. Non-coplanar effects

IMAGR has a **CPARM(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. See the explain information for further details.

5.3.4.6. 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 **CPARM(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 **CPARM(6)** in x and **CPARM(7)** in y.

5.3.5. 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

> OUTNAME '*your-chosen-name*' ↵ if you want to specify the output disk file name in *AIPS* ; the default is the image name on tape.

The string *your-chosen-name* can be any (< 12-character) title that you want to use as the image name within *AIPS*. **IMLOD** also allows you to specify the 6-character image Class parameter. Use OUTCLASS '*abcdef*' ↵, if you wish to change the Class from that on your input tape as the image is read.

> OUTS -1 ↵ if you want to keep the sequence number the same as that on tape; the default is the highest unique number for images with this name and class in your current *AIPS* catalog.

> NFILES 0 ↵ to have no further files skipped — **important** if you have just used **AVFILE** to position the tape!

> NCOUNT *m* ↵ to load *m* images consecutively starting with the image at the current tape position; default is *m*=1. 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 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 usually 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*).

To load *m* consecutive further images from the same tape using the default **OUTNAME** (the names from their FITS header), skipping *n* from the sequence:

> OUTNAME ' '; OUTCL ' ' ↵ to ask for the system defaults.

> NFILES *n* ↵ to skip *n* file marks.

> NCOUNT *m* ↵ to specify loading *m* consecutive images after the skip.

> GO IMLOD ↵ to run the program.

To dismount the tape when **IMLOD** is done:

> DISMOUNT ↵

7.1.2. IMLOD from FITS disk

IMLOD can also read FITS-format images from external disk files into your *AIPS* catalog. The control parameters are the same as described above for reading FITS tapes, except that **NFILES** and **NCOUNT** are ignored and are replaced by an **INFILE** parameter. Disk image files must therefore be read in only one at a time (per execution of **IMLOD**). **INFILE** 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.

One "feature" of *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). So if your computer distinguishes upper and lower cases for disk, directory, or file names, you must do two things to prepare for this before running *AIPS* . First, you must restrict your external disk file names to upper-case characters and numbers. Second, you must 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* :

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 $^{C}_{R}$	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 $^{C}_{R}$	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 $^{C}_{R}$	to stop Cleaning when the peak of the residual image falls to f Jy/beam.
> NITER n $^{C}_{R}$	to stop Cleaning when n components have been subtracted. There is no default; zero means no Cleaning.
> BCOMP 0 $^{C}_{R}$	to begin a new Clean — see below for restarting one.
> NBOXES 0 ; BOXFIL ' ' $^{C}_{R}$	to specify no Clean search areas in advance; see § 5.3.3.
> CMETHOD ' ' $^{C}_{R}$	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 $^{C}_{R}$	to use the "normal" criteria for deciding when to do a major cycle; see below.
> BMAJ 0 $^{C}_{R}$	to have **IMAGR** use a Clean beam which is a fit to the central lobe of the dirty beam.
> DOTV 1 $^{C}_{R}$	to have dirty and residual images displayed on the TV; see § 5.3.3.
> INP $^{C}_{R}$	to review the inputs — read carefully.
> GO $^{C}_{R}$	to start **IMAGR**.

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 2048 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 **CPARM(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

> CLR2NAME ⏎	to not use a Clean-component model.
> FLAGVER 0 ⏎	to apply the most recent flag table.
> REFANT 1 ⏎	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. 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 ⏎	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. It 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.
> SOLINT 3 ⏎	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.
> APARM(1) 2 ⏎	to accept solutions when only 2 antennas are present; default is 6.
> APARM(2) 0 ⏎	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 ⏎	to treat polarizations separately; **APARM(3)** $>$ 0 averages RR and LL.
> APARM(4) 0 ⏎	to not average frequencies within each IF before the solution.
> APARM(5) 2 ⏎	to do separate least-squares fits for single- and multi-band delays. **APARM(5)** $\leq$ 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, as here, $>$ 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 effecting all IFs. Normally users should use **APARM(5)=2** for multi-IF data.
> APARM(6) 1 ⏎	to get some useful, but limited, messages including the SNR.
> APARM(7) 9 ⏎	to avoid false detections by setting a moderately high minimum for the SNR accepted.

10.9. Transposing the cube

The task **TRANS** will transpose the cube. Typical inputs are:

> TASK 'TRANS' ; INP ⏎	to review the inputs.
> INCLASS 'LMVCUB' ⏎	to select the untransposed cube.
> TRANSC '312' ⏎	to make new axis order 3,1,2 in terms of the old axis order (RA, DEC, VEL becomes VEL, 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.

10.10. Further profile analysis

A wide variety of programs is available to do further analysis of the data.

> HELP CUBE ⏎	to list them all on your terminal.

This help file is also listed in § 14.

By displaying the transposed cube you can inspect RA, VEL or DEC, VEL images. The cube can be rotated with **LGEOM** (if the α–δ pixels are square), *e.g.*, to align one of the axes with the major axis of a galaxy.

A single profile can be produced from these images with **SLICE**, then plotted using **TKSLICE** (see § 7.5). **PLCUB**, **PLROW** and **XPLOT** are convenient programs for displaying multiple profiles.

The task **XBASL** can be used to remove baselines. 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 amplitude.

Smoothing and blanking are important for almost all analysis programs. **CONVL** works on cubes and does a spatial smoothing (on LMV cubes). Using all the defaults in **XSMTH** performs a Hanning smoothing in velocity (on VLM cubes). Smoothing 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.

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 to default, 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**.

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. 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. **RGBMP** computes "integral" images another way — as three weighted sums representing the low, center, and high velocity parts of the cube. An interesting display results from:

> TVINIT ; TBLC 0 ; TTRC 0	to initialize everything.

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.

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

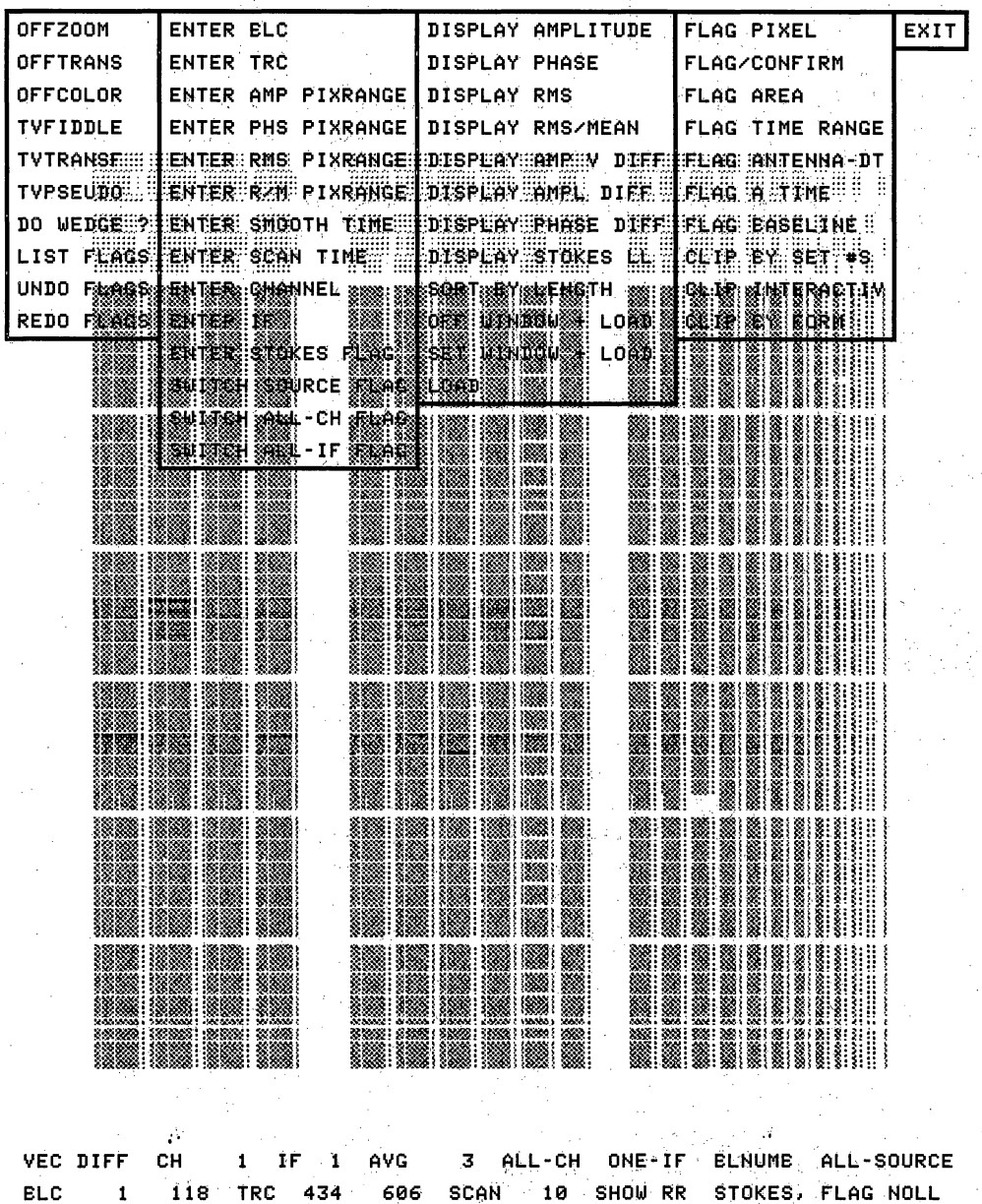
You help Clean to guess what may have happened in the unsampled "hole" at the center of the *uv* plane by including a zero-spacing (usually single-dish) flux density when you make the image. This gives Clean a datum to "aim at" in the center of the *uv* plane. Extended structure can often be reconstructed 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 § 5.3.6). **IMAGR** treats the zero spacing differently from previous tasks. The adverb **ZEROSP** gives five values, the I, Q, U, V fluxes,

```
Image=MULTI     (UV)          Filename=25/11/88    .X BAND.   1
Telescope=VLA                 Receiver=VLA
Observer=AC238                User #=  103
Observ.  date=25-NOV-1988      Map date=05-FEB-1994
# visibilities    191317     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   8.4110000E+09    1.00 1.2500000E+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
Keyword = 'CORRMODE'  value = '        '
Keyword = 'VLAIFS  '  value = 'ABCD    '
```

This header identifies the file as a multi-source file (**Image=MULTI**) with 191317 floating-point visibilities in time-baseline (**TB**) order. There are two entries on the **IF** axis. These correspond to the 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 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 VLA allows 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.

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



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) 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 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 VLA. All baselines are shown once only in baseline number order. Antennas 5 and 12 (at least) appear to be missing. 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, all spectral channels, only one IF, and all Stokes except LL.

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. Task **UVDIF** (§ 6.2.1) may help you to decide whether there have been significant changes to your data due to the previous iteration of **CALIB**.

There is a new task called **SCMAP** which attempts to implement this sequence inside a single task. **SCMAP** contains all of **IMAGR** (for a single field) and all of **CALIB**. It attempts 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 less flexible since it does not let you change imaging and self-calibration parameters (except for Clean boxes and several **TELL** parameters including loop gain and solution interval) and currently does not allow data editing between self-calibration steps. (The editing is anticipated, but has not been implemented as yet.) The **CALIB** and **IMAGR** process is similar and allows for multiple fields, 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 with listed with **LISTR**.

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* data base.
> IN2D *n2* ; GET2N *ctn2* ↵	to select the Clean model image(s) to use.
> 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 four characters of **IN2CLASS** with the field number minus one given in the last two characters as is done by **IMAGR**.
> NCOMP $= n_1, n_2, \ldots$ ↵	to cut off the model at the n_i^{th} Clean component in the i^{th} image.
> INVERS m ↵	to specify the **CC** file version number to use from *every* model image; 0 means the highest.
> 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.

this. At neither site are the image recorders directly connected to an *AIPS* computer by *AIPS* software. Thus the images must be transferred to a non-*AIPS* system. Consult Appendix Z for details of the system for using the NRAO Dicomed film recorders.

Multiple calibrators may be used in determining the feed polarization, but the data from them must be accurately calibrated. In particular, the phase calibration of any calibrator used to determine antenna polarizations should be determined from that calibrator itself (*i.e.*, the source should be self-calibrated). Note that this will normally have occurred for all gain calibrators if the procedure described in the previous sections was followed.

The normal phase calibration technique treats parallel-hand visibilities in the two orthogonal polarizations independently. Thus, there will be a systematic phase difference between the two polarizations systems. This difference may be due to differences in instrumental phase offset for the two systems or due to the propagation medium (*i.e.*, Faraday rotation) or both. Faraday rotation effects are particularly bothersome as they may be time variable and increase rapidly with wavelength. For data at L band or longer wavelengths, *AIPS* should be given an estimate of the ionospheric Faraday rotation measure using task **FARAD**. This task computes the ionospheric rotation measure using either total electron content from a nearby ionospheric monitoring station (Boulder Colorado for the VLA) or an empirical model that uses the monthly mean Zurich sunspot number (R1) as a measure of solar activity. If monitoring data are available, they should be used in preference to the model. **FARAD** enters the ionospheric Faraday rotation measure into the **CL** table. This is used by **PCAL** when determining antenna polarization parameters and is used by other calibration tasks to de-rotate the data when polarization corrections are applied. **FARAD** may be run any number of times with different parameters before **PCAL** is run; each run of **FARAD** over-writes the values written by the previously.

Data from Boulder for the total electron content for year *mm* is contained in the file **TECB**.*mm* in the directory with logical name **AIPSIONS**. Currently, data are available for 1980 onwards through part of 1992. Some gaps in the TEC data occur, particularly for years 1980 and 1988 and for times when solar activity has been high and no reliable estimate of the total electron content could be made. Unfortunately, we are no longer able to get the TEC data from Boulder. If your VLA data are more recent than early 1992, you can consider using the ionospheric model in **FARAD**, but you should be aware that the model is crude and should check that it improves matters before using it in your final calibration.

The phase offsets between the right-hand and left-hand polarizations at a given time may be determined using data from a source with a known angle of linear polarization which have had the effects of imperfect feeds removed. The phase of the right-left correlations or the conjugate of the left-right correlations indicates the phase difference between the two polarizations.

The (initial) need for ionospheric corrections can be bypassed if either (1) you use unpolarized sources in **PCAL**, and/or (2) the ionosphere was well behaved during your observations. Typical rotation measures are only a few and therefore affect only L and longer-wavelength bands. The ionosphere is almost always well enough behaved to be ignored at shorter wavelengths and is usually able to be ignored even at L band. Changes of around 10° in the relative phases of R and L polarizations are not enough to disrupt a **PCAL** solution seriously. And, fortunately, calibrators at long wavelengths, such as P band, tend to be unpolarized. In general, if the ionosphere is well behaved, it can be ignored. If it is bad, no simple model is able to correct it and you may simply have to forget about polarization for that observing run. Note that an apparent position angle variation of 3C286 with time probably indicates that ionospheric rotation is significant. But, if 3C286 shows large rotations, it does not follow that *its* rotation can be applied to other directions in the sky. All it implies reliably is that a *model* is needed.

Polarization calibration may be performed on amplitude- and phase-calibrated VLA data using the following five-step procedure:

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'  CR
```

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  CR
```

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.

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*. **OUTFILE** 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. The adverb **INFILE** 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 hexadecimal number with leading zeros. 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 61. 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.

> SETXWIN (*dx,dy*) ↵	reads pixel coordinate of the center of a *dx*-pixel by *dy*-pixel window and sets the adverbs **BLC** and **TRC**.
> TVBOX ↵	works like **TVWIN** above except that it is used to set up pixel coordinates to define CLEANing areas for the *AIPS* CLEAN routine (**APCLN**). The adverb **NBOXES** must be entered before the **TVBOX** command and is the number of rectangular CLEANing boxes to be set with the TV cursor, trackball, and buttons.
> REBOX ↵	allows revision using the TV of the CLEANing areas set previously with **TVBOX**.
> TVSLICE ↵	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.

Other interactive TV functions are available. Type HELP TVINTER ↵ and HELP CURSOR ↵ for additional information. The **15JUL90** lists of these other TV functions are also given in § 15.

7.4. Comparing two (or more) images

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 two 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 defaulted, **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**. The following commands would load two 256 by 256 pixel images from slots 1 and 2 on disk 1 *side-by-side* on channel 1 of a 512 by 512 TV display, for example:

> INDI 1; GETN 1 ↵	to select the input disk and the first image
> TVCH 1 ↵	to select TV channel 1 for the loading
> TVCORN 1 128; TVLOD ↵	to load the first image
> GETN 2 ↵	to select the second image
> TVCORN 257 128; TVLOD ↵	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 images as you wish to a single TV plane with this technique, which is therefore a powerful method for making "montages". The number of simultaneous images is limited only by your image sizes, the need to avoid overlaps — 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.

The second tool is the classic "blink" technique from optical astronomy. **TVBLINK** allows you to load images to two different planes of the TV memory, then to alternate rapidly between which is displayed. The rate and duty cycle of the blinking, and the transfer functions applied to the planes, are controlled interactively with the TV cursor. The two images described above could be "blinked" against each other by the following command sequence:

> INDI 1; GETN 1 ↵	to select the input disk and first image

2. STARTING UP *AIPS*

This chapter contains general information concerning the steps needed to obtain access to, and use, an *AIPS* system. It attempts (as does the design and coding of *AIPS* itself) to avoid specific references to particular computer devices and to the peculiarities of any one *AIPS* installation. We will assume, for the most part, that you will be running *AIPS* on a Unix workstation although *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 *AIPS* computer

Most *AIPS* sites now possess a number of computers which are networked together and are each individually capable of running *AIPS* while sharing both disk and tape resources. Most such computers cannot support more than a few simultaneous users (or simultaneous incarnations of the same user) of *AIPS*. Thus, most locations are obliged to institute a mechanism for distributing the available *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 *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 "*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.

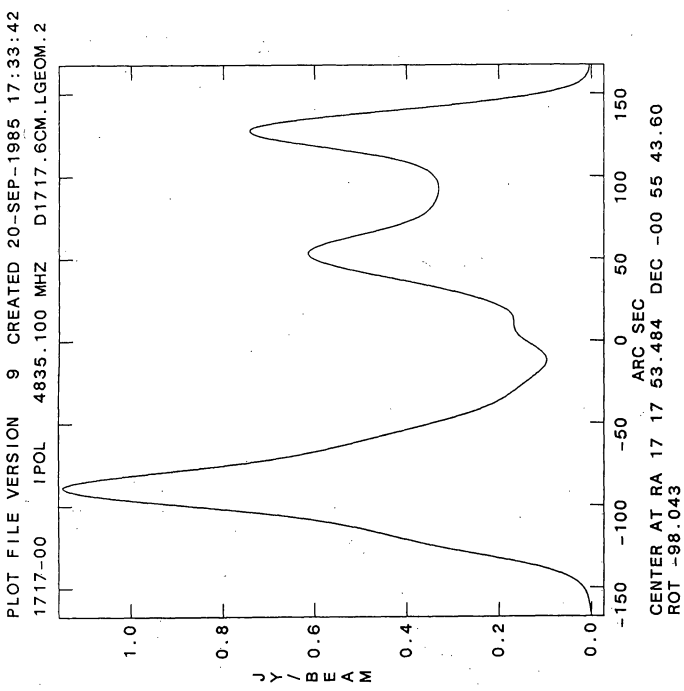
AIPS can support several simultaneous users which it calls **AIPS1**, **AIPS2**, *etc.* In the workstation environment, this is used primarily to allow one user to have separate simultaneous *AIPS* sessions using multiple windows. This also allows users to log in to remote computers (*e.g.*, with the Unix tool **rlogin**) and run *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

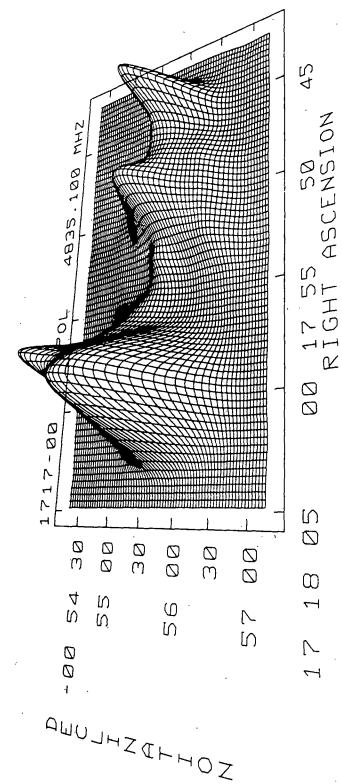
The way that a workstation behaves is a function of the type of workstation, the computer operating system, the window manager program, and the set-up files for the specific computer account being used. Given all these variables, it is hard to give detailed usage instructions. Nonetheless, it is important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

2.2.1. Logging in to the workstation

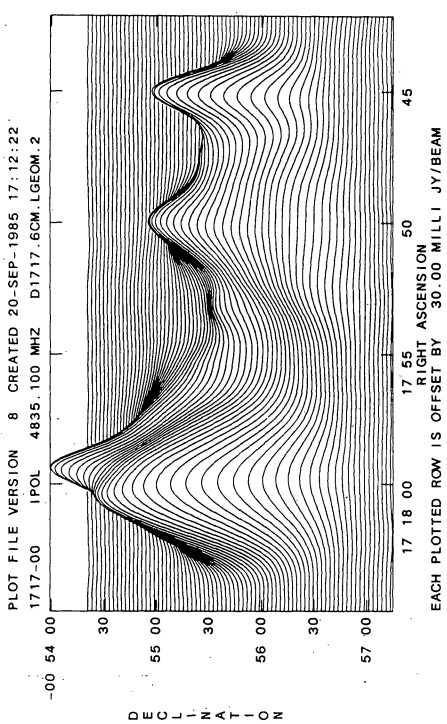
Find your assigned computer in the appropriate *AIPS* caige or office, or an available one intended for general use (checking any sign-up sheets for it). Typing $^{C}_{R}$ on the keyboard will reveal the current state of the workstation. If you see a message prompting you to log in (*e.g.*, **Console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



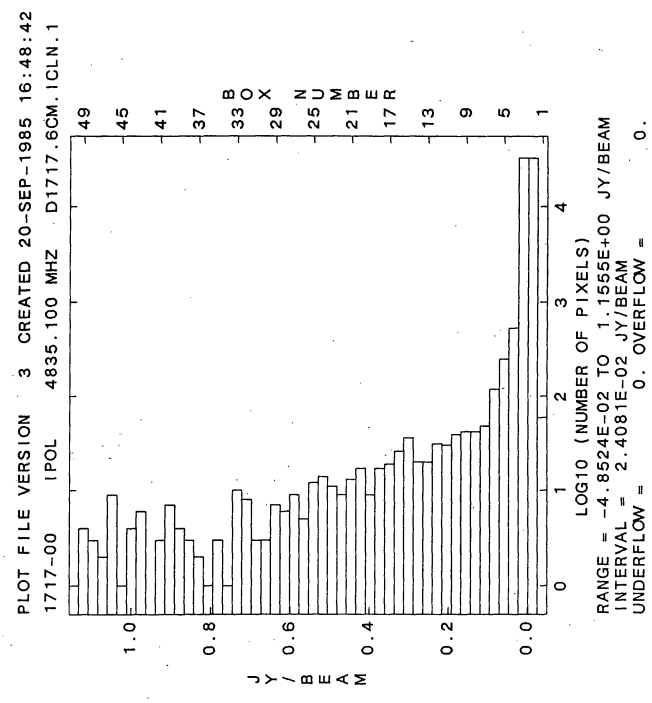
SL2PL



PROFL



PLROW



IMEAN

menu panel and then hitting any button on the mouse or trackball. Instructions will appear then on your terminal screen *viz.*, place the cursor **in** the global window at the start of the required subsection and hit button 'A' then mark the end of the subsection with the cursor in the global window and hit button 'B'. **IBLED** will attempt to fit the selected subset into the main window but, if it cannot, it will do it's best and inform you that there is more data than it can fit in the window. This additional data can be plotted later by placing the cursor over the **NEXT FRAME** legend on the left hand menu panel and pressing a mouse or trackball button. *On-line help* is available for any of the selected options by pressing button 'D'.

If your data needs extensive editing, you may make edits in the first sub-frame displayed and then move on to the next, adjacent frame with the **NEXT FRAME** command. Continue till all the data on this baseline have been edited. At any time, you may choose to go back to the previous frame or the next frame (**PREVIOUS FRAME** and **NEXT FRAME** respectively) using the menu commands or you can choose a specific section of data with the **SEL FRAME** option.

The data selection illustrated here shows the typical manner in which all of the facilities in **IBLED** are invoked. Some of the options may require some additional input from your terminal. For instance, to select baseline 5–6 without having to go through all the other baselines first, use the **SELECT BASE** option. Your terminal will inquire which baseline you require — follow the instructions and type in:

```
> 5 6 CR
```

and the program will display this baseline after a short while. (The antenna numbers corresponding to antenna names can be found using task **PRTAN**.)

There are several principal editing tools in this program. The main method is **FLAG TIMERANGE** where the cursor and buttons are used to select a section of data to flag completely on that baseline. **FLAG AREA** is another useful option — this allows you to flag data within a rectangular box delimited by the cursor and buttons. **CLIP ABOVE** and **CLIP BELOW** are used heavily also — particularly to remove seriously discrepant points. The **INTERACTIVE CLIP** option may prove useful in some cases of good signal-to-noise. This option lets you define a curve and clip either above or below this user-defined line as appropriate. **CLIP ON MEAN** will allow you to interactively flag data outside the range of $mean \pm (n \times \sigma)$; the program will prompt you for the constant n at your terminal. Note that any of the editing options can be used to flag a particular antenna throughout the *whole* database at the times indicated by the current editing option you are using. This can be invoked by the use of the **FL** *telescope name* option on the left hand menu. This will enable you to delete bad data on baselines with that certain antenna if, for instance, the antenna was not on-source for a period of time during the scan.

At any time you may change the information displayed from phase to amplitude (or *vice versa*) with the **SHOW AMP** and **SHOW PHASE** legends.

Once you have completed the editing of a particular baseline, you may move on to the next baseline with the **NEXT BASELINE** command in the left hand menu. Proceed with the edits on all required baselines till you are happy, then select **EXIT** on the left hand menu panel. The program will inform you at your terminal of how many edit commands you have made. It will enquire if you wish to apply these edits or not to the data. If you decide not to use these edits and type "NO", the program will give you one chance to cancel this command before quitting. If you have a multi-source database, **IBLED** will add the editing commands to the chosen flagging (FG) table (usually FG table # 1). Setting **FLAGVER = 0** will cause **IBLED** to add the flagging commands to the highest-numbered, extant version of the FG table. **IBLED** will create an FG table if one does not exist already. These flagging commands can be applied in the usual way by specifying **FLAGVER** in subsequent programs. In the case of single-source databases, the flagging commands created by **IBLED** will be used to flag the data *directly i.e.*, by setting the weights of the offending points to -1.0.

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. 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.
> 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 needs to be low since the SNR is calculated as a phase difference from model and this can be large. Start with a value ≤ 1.
> 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 § 9.10.1).
> 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.
> GO ↵	to run **UVAVG**.

old Start the OLD version of AIPS. For NRAO this is a frozen version which is distributed worldwide.

new Start the NEW version of AIPS. For NRAO this is the most recently released version and is frozen right at the time of initial public release.

tst Start the TST version of AIPS. For NRAO this is the unreleased, development version. This is the default.

local Start a local copy of AIPS.EXE residing in the current directory.

debug Start AIPS in debug mode. The user will be prompted for the name of the debugger (e.g. dbx, adb, csd, xde, dbxtool, debugger), and whether to run AIPS itself in debug; if you answer no, only AIPS tasks will be run in debug mode.

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, no menu will be presented.

da=host[,host,...]

Use disks from the comma separated list of machines, which have AIPS data areas. Data areas from "required" hosts and any disks on the local machine are always added, regardless of the list of hosts.

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

AIPS has a limit of 15 disks in any one session. 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 be assigned as disk 1.

There is a hierarchy of disk "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. If it doesn't exist, AIPS finally looks for:
$NET0/DADEVS.LIST	which is the site-wide disk configuration file.

or "stripes" in smooth low-brightness 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).) This algorithm is embodied in the well-tested *AIPS* task **SDCLN**. 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 **HORUS**, by **UVMAP** or by **MX** with **NCOMP** set to **0**, each twice the (linear) size of the region of interest. 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-resolution" 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.

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**.

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 ↵ or contact Tim Cornwell at NRAO's Socorro office.

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 pointlike 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.)

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 16 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 maps 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.

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

> FREQID 1 C_R	to select which **FQ** value to use.
> ANTENNAS 0 C_R	to solve for all antennas.
> REFANT *n* C_R	to set the reference antenna number.
> DOCALIB FALSE C_R	to avoid applying calibration.
> BCHAN 1 ; ECHAN 0 C_R	to use all channels.
> BPASSPRM 0 C_R	to turn off all "parameters."
> BPASSPRM(5) 0 C_R	to divide by channel 0 before determining antenna-based bandpasses
> IN3DI *a* ; GET3N *b* C_R	to specify the channel 0 data file, or
> CLR3NAME C_R	to have channel 0 found from the input data themselves.
> CHANSEL 20 50 1	to use the average of all channels from 20 through 50, for example, to determine channel 0, when the third input file name is empty.
> FLAGVER 1 C_R	to apply flag table 1.
> SOLINT 0 C_R	to use scan averages.
> BPVER 1 C_R	to select the output **BP** table number.
> INP C_R	to review the inputs.
> GO C_R	to run the program when inputs set correctly.

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

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

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

> DOTV TRUE C_R	to use the TV.
> NCOUNT 1 C_R	to plot one antenna per page/screen.
> GO C_R	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 speed it up by hitting TV buttons A, B, or C, or tell it to quit by hitting button D. If DOTV = -1, then **POSSM** makes multiple plot extension files,

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. In order to ensure that the velocity is constant throughout the data you should run **SETJY** and then **CVEL**.

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.
> VELTYP 'LSR' ↵	to select the rest frame of the velocity.
> VELDEF 'OPTICAL' ↵	to define velocities by the optical convention (which we recommend).
> GO ↵	to run the program.

Then run **CVEL**:

> TASK 'CVEL' ; INP ↵	to review the inputs.
> INDISK *n1* ; GETN *ctn1* ↵	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 in order to protect you from applying it twice.

9.11.2. Amplitude calibration

The calibration scheme suggested in *AIPS* for spectral-line VLBI data utilizes the total-power spectra*, although the continuum method using T_{sys} values and **ANCAL** (see § 9.4) 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**.

* see Lecture 12 in *VLBI, Techniques and Applications*, eds. Felli and Spencer, published by Reidel

9.13. 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.

AN	Antenna table. Contains a list of the antenna names and station coordinates.
CL	Default calibration table. 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 and delay entries.
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.
NX	Index File. Contains information about the time, source, subarray 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.
SN	Solution table. Contains antenna delay, rate, phase, and amplitude corrections solved for by **CALIB** and **FRING**.
SU	Source table. Contains a list of the sources found within the multi-source file, including information on source positions and flux density
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**.
CT	**CALC** table. Contains the input parameters passed to **CALC** to generate the polynomials recorded in the **IM** table.
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.
MC	Model Components table. Contains the various components of the geometric model used in the VLBA correlator to generate the **IM** table.

12. TIDYING UP AND EXITING FROM *AIPS*

Before exiting from *AIPS*, you should do several things to tidy up various kinds of debris you may have left in the system.

12.1. 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 tape in FITS format using the task **FITTP**. **FITTP** can write more than one *AIPS* file on tape in a single execution. For example, to backup all sorted *uv* files (class UVSRT), type

> TASK 'FITTP' ; INP C_R	to review the inputs.
> DOALL TRUE C_R	to specify that all files with the allowed name parameters are to be written.
> INNA ' ' ; INSEQ 0 C_R	to allow any name and sequence number.
> INDISK 0 C_R	to allow any disk.
> INTYP 'UV' C_R	to restrict to *uv* files.
> INCLASS 'UVSRT' C_R	to restrict to class UVSRT files.
> INP C_R	to check the inputs.
> GO C_R	to write the tape.

Then, to write all 3C123 files on disk 2 after the sorted *uv* data files, for example, type:

> INTYP ' ' ; INCLASS ' ' C_R	to allow any class and type.
> INNA '3C123' C_R	to restrict things to 3C123 files.
> INDISK 2 C_R	to restrict to disk 2.
> WAIT ; GO C_R	to have **AIPS** wait for the **FITTP** execution started above to finish and then to run **FITTP** with the new inputs.

(On some systems (*e.g.*, VAX/VMS), a system-dependent global backup procedure may be available. This procedure will allow you to write all of your files on tape quickly, verify the tape, and then, optionally, delete the files. Type EXPLAIN ABACKUP and EXPLAIN ARESTORE for information. Such procedures are very nice, but they produce a highly system-dependent tape. A tape written on one VAX may not be readable on another, for example. Nevertheless, **ABACKUP** should be used (where available) instead of leaving data on disk for prolonged periods.)

If space is short, data sets which remain on the disk for more than a week or two will be ruthlessly deleted by designated *AIPS* gorillas. If you wish to leave data on disk for an extended period of time, please notify the System Manager and bring an appropriate "gift".

12.2. Deleting your data

Please delete redundant images and data as soon as possible to preserve disk space for 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.

12.5.3. 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.

13.4. My program crashed for lack of disk!!

Read this *COOKBOOK*, § 3.6 and Appendix Z for your computer system.

13.5. My tape refuses to read/write!!

1. Is the tape correctly loaded in the drive and is the drive "on line" (check the **ON LINE** light)?
2. 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.
3. Did you actually mount the tape in software from the *AIPS* level with the **MOUNT** verb (or, on those systems requiring it, from the Monitor level)?
4. Have you specified the **INTAPE** or **OUTTAPE** number to correspond with the drive you mounted the tape on?
5. 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.
6. 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. If this fails, refer to the notes in Appendix Z (*e.g.*, § Z.1.7.3) for suggestions.
7. 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.
8. 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.

13.6. My terminal has hung itself up!!

If your terminal is "dead", *i.e.*, refuses to echo characters typed on the keyboard or otherwise to show signs of talking to your computer, you have a problem. There are numerous possible causes.

1. Are you executing a long verb, *e.g.*, **TIMDEST**, **REWIND**, **AVFILE**, **IMSTAT**? 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 (or your terminal).
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.

5. Have you stopped output to your terminal 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.)
6. Can you abort **AIPS** at your terminal using the appropriate system commands (*i.e.*, CTRL C on UNIX machines; CTRL Y on VAX/VMS)?
7. Do other terminals connected to the computer appear to be "alive"? If so, login on one of them and inquire about the status of your **AIPS** program and tasks. It might be necessary to stop them from your new terminal and then log back in on your old terminal. If this doesn't bring your terminal back to life, report the problem to the *AIPS* Manager.
8. If all terminals appear dead, then your computer has probably "crashed". Report the problem to your *AIPS* or System Manager. If you feel you must reboot the system, do so *only* after checking that all current users and the System Manager (if available) agree that that action is required.
9. If even a reboot fails, report the problem to the System Manager or hardware experts and go do something else. UNDER NO CIRCUMSTANCES SHOULD YOU ATTEMPT TO REPAIR THE TERMINAL, TELEVISION, DISK, CPU, OR OTHER HARDWARE DEVICES. Such repairs must be performed by trained personnel.

13.7. *AIPS* has been "suspended"

On UNIX machines, the current job (in this case *AIPS* can be "suspended" by typing **CTRL Z**. The job can then be either put into the "background" with **bg** or returned to the "foreground" again with **fg** where it will continue to accept terminal input. If your *AIPS* appears to be "suspended", try typing **jobs** to see which jobs are attached to your terminal and then use **fg %***n* to bring back job *n* where *n* is the job number of the suspended *AIPS*.

13.8.1. 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° oven for 1 hour.

13.8.2. 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° about 15 minutes or until the bananas are tender (but not soft), basting with the sauce several times.

14. *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 of *AIPS*. Sections 14.1 to 14.8 therefore provide more detail about *POPS*.

The remaining sections describe some advanced features of *AIPS* itself. Section 14.9 points out some important, but often neglected, attributes of the most-used *AIPS* verb — GO. Section 14.10 briefly describes the *AIPS* batch system. Section 14.11 describes a few features of particular interest to "remote users" of *AIPS* systems. Section 14.12 describes how to add your own tasks to *AIPS*.

14.1. AIPS syntax

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 "pseudoverbs" which, with their arguments, must stand alone. For example, GET APMAP ; INDISK = 1 C_R will ignore the INDISK = 1. When in doubt, see the HELP files for the pseudoverb 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 usually suffice. This "minimum matching" has been exploited throughout this *COOKBOOK*.
3. The parameter variables in AIPS are called "adverbs." They are assigned values by the equals verb (=), *e.g.*, INTAPE = 2 C_R. The equals sign may usually be replaced by a space. The exception arises when the variable on the left is a subscripted 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) C_R).
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 C_R. 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 C_R. 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 C_R will produce SHIFT = -21, -21.
5. Adverbs can be used in arithmetical expressions or set equal to other adverbs, *e.g.*, OUTNAME = INNAME ; OUTSEQ = 2.5 * INSEQ + 3.
6. Both upper and lower case letters may be entered by the user, but adverb character string values are converted to upper case before being stored and used by AIPS.

As an example, these shortcuts allow the following AIPS command sequence:

```
> INNAME '3C138' CR
> INCLASS 'IMAP' CR
> INSEQ 0 CR
> BLC 200,200 CR
> TRC 300,300 CR
```

> PROC MFIT (I) $^{C}_{R}$	to define procedure **MFIT** with one argument **I**. (**I** and **J** are two dummy adverbs which are already defined in **AIPS**.)
: GETNAME(I) $^{C}_{R}$	(Notice the prompt symbol : . This means that we are in the procedure-building mode.)
: TVLOD ; IMXY ; MAXFIT $^{C}_{R}$	to load the image, produce and read the cursor, and fit the maximum near the cursor position when a TV button is pressed
: RETURN $^{C}_{R}$	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 $^{C}_{R}$	to designate the end of the procedure-building mode and to get back into the normal (prompt >) 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 § 14.6 below.

To list the procedure **MFIT**, type:

> LIST MFIT $^{C}_{R}$

This will produce the following:

```
1    PROC MFIT(I)
2    GETNAME(I)
3    TVLOD ; IMXY ; MAXFIT
4    RETURN
5    FINISH
```

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 $^{C}_{R}$	to enter Procedure editing mode.
; GETNAME(I) ; TVLOD $^{C}_{R}$	(Notice prompt symbol ; for procedure-editing mode.) This change replaces the old line 2 adding a **TVLOD**.
; IMXY ; MAXFIT $^{C}_{R}$	to add a line between the changed line 2 and old line 3.
; GETNAME(I+1) $^{C}_{R}$	to add yet another line after 2.
; ENDEDIT $^{C}_{R}$	to terminate procedure editing.

> LIST MFIT $^{C}_{R}$

Listing the modified procedure will give:

```
1    PROC MFIT(I)
2    GETNAME(I) ; TVLOD
3    IMXY ; MAXFIT
4    GETNAME(I+1)
5    TVLOD ; IMXY; MAXFIT
6    RETURN
7    FINISH
```

To delete lines *n* through *m* from a procedure, type:

> ERASE *xxxxxxxx n : m* CR 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 CR

; *(Type additional lines as needed.)*

; ENDEDIT CR

Notice that the lines are renumbered after any **EDIT** or **ERASE**. Use **LIST** to determine the new line numbers.

The pseudoverb **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 (CR) 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 CR

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 CR). CORE CR will tell you how much memory is left.

The procedure **MFIT** can be executed by:

> MFIT(*n*) CR 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 CR

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 pseudoverbs.* The "prohibited" pseudoverbs include **SAVE**, **GET**, **STORE**, **RESTORE**, **PROCEDURE**, **EDIT**, **ENDEDIT**, **MODIFY**, **LIST**, **CORE**, and **SCRATCH**. Others do not make much sense in procedures, including **MSGKILL**, **DEBUG**, and **ABORTASK**. Other pseudoverbs are, however, particularly useful in procedures. These include **TGET**, **TPUT**, 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. 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. There are numerous arithmetic functions, useful looping constructions, and powerful methods of building arithmetic, logical, and string expressions in *POPS*. Type HELP POPSYM $\subset_R$ or see § 14 for details.

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 § 14.6 below. To list the names of all procedures currently in your **AIPS** environment, type:

> HELP PROCS $\subset_R$

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**. 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*.

14.6. RUN files

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 set of commands that can be typed on the terminal can be stored in a text file. At present, the computer's own text editor must be used to generate this file so this part of the process is machine dependent. The text files are generally stored in an area designated by the word RUN. Thus, in VAX/VMS these text files are stored in a subdirectory called **[AIPS.RUN]** under the **AIPS** logon. On most systems, the file name must be of the form *Accccc.nnn*, where *A* is any alphabetic character, *ccccc* are any 0–7 alphanumeric characters, and *nnn* is your user number expressed in 3 hexadecimal characters with leading zeros. *AIPS* programmers also provide a few **RUN** files for general use in each release *e.g.*, **VLACALIB**, **VLACLCAL** and **VLARESET**. These are stored under user number 1 and are automatically available to all users. See Appendix Z of this *COOKBOOK* for some helpful information applying to NRAO *AIPS* systems (*e.g.*, § Z.1.8).

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 comments which will not be executed by **AIPS**.

To use a procedure, type:

> RUN *xxxxx* $\subset_R$ — to read and execute file *xxxxx* in **AIPS**. The text will be listed as it is read.

> — continue **AIPS** in normal mode.

The file *xxxxx.nnn* or, if it does not exist, *xxxxx.001* will be executed by the above command. Note that minimum match also applies to **RUN** file names.

RUN files are a convenient place to store procedures that are often used, since the code can be accessed from **AIPS** simply by typing RUN *whatever* $\subset_R$. **RUN** files are also useful for storing the commands associated

to execute either by an immediate argument, *e.g.*, GO UVSRT ↵, or by the parameter **TASK**, *e.g.*, TASK 'UVSRT' ; GO ↵. **GO** has two further parameters, **DOWAIT** and **VERSION**. 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.

VERSION, the last input to **GO**, 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. Type HELP VERSION ↵ for details.

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 ↵.

14.10. 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. The Array Processor, if any, can be used in batch although interactive users generally have priority on it. **RUN** files may be used in batch jobs — as the batch editor facility is also primitive, they are particularly attractive to batch users.

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. There are generally two queues and, if so, jobs which use an array processor cannot be run in queue 1.

> BATCH ↵ to enter batch preparation mode.

< TASK = 'UVSRT' ↵ (Notice < prompt. Begin typing as in **AIPS**.)

< INN = '3C16' ; INCL = 'UVDATA' ↵

14.12.3. 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!

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 '/
```

This should be modified 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 a task, type:

```
$ COMTST NTASK ↵R                (VMS)
% COMLNK NTASK LINK.OPT ↵R       (Convex UNIX)
```

There should be no important error messages. Sometimes, the template tasks have not been fully updated and there may be a few errors to correct, however.

After successful compilation and linking, try executing the task.

Under VMS, log in to *AIPS* and start **AIPS OLD**, **AIPS NEW** or **AIPS TST**, as desired. To run any tasks that have been compiled and linked in your logon, set the adverb

```
> VERSION 'DISK:your-logon' ↵R
```

You may have to explicitly state the disk drive. Check the inputs and run the program.

Under Convex UNIX, stay logged in to your area. To initiate the **AIPS** program, type:

```
% AIPS [OLD,NEW,TST] ↵R (as appropriate)
```

Your *AIPS* Manager may have to make some arrangements for you to activate *AIPS* from your logon. Once you have started *AIPS*, type

```
> VERSION 'MYAIPS' ↵R
```

look at the inputs, then execute the task.

14.12.4. 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 PRGNAM near the beginning of the program 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, usually in a common named **/INPARM/**.
4. Change code as desired. Try to modify **HI** 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 yourself later or if anyone else will use your program.
7. Compile and link the modified task following the instructions in the previous section. When it is error-free, try it out in *AIPS*.

14.12.5. 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 from outside *AIPS*.
3. **FUDGE**—modifies an existing *uv* data base and writes a new data base.
4. **UVFIL**—writes a new *uv* data base. Input 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 straightforward to use and the accompanying documentation should suffice.
2. It is possible to handle several input images and several output images with **TAFFY** by using multidimensional images and the use of the axis transposing task, **TRANS**. Suppose you want to calculate the spectral index from a set of four images. First, use **MCUBE** to put the four images into one data cube. You will have to redefine the frequency axis of each image to 1,2,3,4 in order to run **MCUBE**. Then, transpose the cube so that the frequency axis is first with RA and DEC 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 arbitrarily specified. 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 RA and DEC are the first two axes and the spectral index, etc are the third axis.
3. You can write and debug outside *AIPS* any subroutines that **DIDDLE** will call in calculating the output image. This will speed up debugging of your algorithms.

4. It is possible to read several input rows before writing one output row.

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**.

FUDGE reads an existing *uv* data base point by point and creates a new *uv* data base with the modified data. It can easily be used to make simple changes in a *uv* data base; for example, if the *u*, *v*, or *w* terms are to be recalculated, if all phases are to be changed in sign, etc.

1. If the output image has different dimensions than the input image (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*, from which a *uv* data base is created, cataloged and filled into *AIPS*. This task is useful for transcribing *uv* data in an arbitrary format into an *AIPS* data base.

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* data base, point by point, in the subroutine **FIDDLE**. The comments in **FUDGE** are complete and an example is given in the code.

14.12.6. Further comments

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. While debugging, use the FORTRAN call `PRINT *, ...` 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). Running a task in *AIPS* using the system debugger can be complicated and is not recommended for the casual user.
3. Declare all variables that you use.

DOALPHA	specifies whether some list is alphabetized
DOARRAY	spcifies if subarrays are ignored or the information used
DOBAND	specifies if/how bandpass calibration is applied
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 a single, centered page or multiple pages of plots
DOCIRCLE	select a "circular" display (i.e. trace coordinates, ...)
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)
DODELAY	selects solution for phase/amplitude or delay rate/phase
DOEOF	selects end-of-file writing or reading until
DOEOT	selects tape positioning before operation: present or EOI
DOGRIDCR	selects correction for gridding convolution function
DOHIST	selects a histogram display
DOHMS	selects sexagesimal (hours-mins-secs) display format
DOINVERS	selects opposite of normal function
DOMAX	selects solutions for maxima of models
DOMODEL	selects display of model function
DOOUTPUT	selects whether output image or whatever is saved / discarded
DOPOL	selects application of any polarization calibration
DOPOS	selects solutions for positions of model components
DORESID	selects display of differences between model and data
DOSLICE	selects display of slice data
DOSTOKES	selects conversion from observed polarizations to Stokes
DOTABLE	selects use of table-format for data
DOTV	selects use of TV display option in operation
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
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 during a DDT RUN.
EDROP	number of points/iterations to be omitted from end of process
EIF	last IF number to be included in operation
FACTOR	scales some display or CLEANing process
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
FORMAT	gives a format code number: e.g. FITS accuracy required
FQTOL	Frequency tolerance with which FQ entries are accepted.
FREQID	Frequency Identifier for frequency, bandwidth combination
FUNCTYPE	specifies type of intensity transfer function
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
GMAX	specifies peak values of model components
GPOS	specifies pixel positions of model components

GRCHAN	specifies the TV graphics channel(s) 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	First record number in a print operation
HISTART	First record number in a print operation
I	spare scalar adverb for use in procedures
ICUT	specifies a cutoff level in units of the image
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
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
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
INTPARM	specifies the parameters of the gain interpolation function
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 DDT RUN
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
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
MAPDIF	Records differences between DDT test results and standards
MAXPIXEL	maximum pixels searched for components in Clark CLEAN
MDISK	Deterimins where input DDT data is found

ROMODE	Specified roam mode
ROTATE	Specifies a rotation
SAMPTYPE	Specifies sampling type
SCALR1	General adverb
SCALR2	General adverb
SCALR3	General adverb
SELBAND	Specified bandwidth
SELFREQ	Specified frequency
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
SNCUT	Specifies minimum signal-to-noise ratio
SNVER	specifies the output solution table
SOLCON	Gain solution constraint factor
SOLINT	Solution interval
SOLMODE	Solution mode
SOLTYPE	Solution type
SORT	Specified desired sort order
SOURCES	A list of source names
STARPOS	Text file name
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
SUBARRAY	Subarray number
SYMBOL	General adverb, probably defines a plotting symbol type
SYSVEL	Systemic velocity
TASK	Name of a task
TBLC	Gives the bottom left corner of an image to be displayed
TCODE	Deterimins which type of DDT is RUN.
TDISK	Deterimins where output DDT data is placed
TIMERANG	Specifies a timerange
TIMSMO	Specified smoothing times
TMASK	Deterimins which tasks are executed when a DDT is RUN.
TMODE	Deterimins which input is used when a DDT is RUN.
TNAMF	Deterimins which files are input to DDT.
TRANSCOD	Specified desired transposition of an image
TRC	Specified the top right corner of a subimage
TRIANGLE	specifies closure triangles to be selected/deselected
TTRC	Specifies the top right corner of a subimage to be displayed
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

15.6. CALIBRAT

```
For a lengthy description of the calibration of interferometric
data (VLA and VLB line and continuum) enter:
             HELP CALIBRAT
**************************************************************
List of verbs, adverbs, tasks in category CALIBRATION

ACFIT      Determine antenna gains from autocorrelations
ANCAL      Places antenna-based Tsys and gain corrections in CL table
ANTENNAS   Antennas to include/exclude from the task or verb
ANTWT      Antenna Weights for UV data correction in Calibration
ASCAL      Computes antenna-based gains based on source model (self-cal)
ASCOR      Applies ASCAL gain tables to other data sets
BASELINE   specifies which antenna pairs are to be selected/deselected
BLAPP      applies baseline-based fringe solutions
BLCAL      Compute closure offset corrections
BLING      fringe fit residual rate and delay on individual baselines
BLVER      specifies the version of the baseline-calibration table used
BPASS      computes spectral bandpass correction table
BPCOR      Correct BP table.
BPVER      specifies the version of the bandpass table to be applied
CALCODE    specifies the type of calibrator to be selected
CALIB      determines antenna calibration:  complex gain
CALIBRAT   describes the process of data calibration in AIPS
CALSOUR    specifies source names to be included in calibration
CHANSEL    Array of start, stop, increment channel numbers to average
CLCAL      merges and smoothes SN tables, applies them to CL tables
CLCOR      applies user-selected corrections to the calibration CL table
CLCORPRM   Parameter adverb array for task CLCOR
CLINT      CL table entry interval
CLSMO      smooths a calibration CL table
CSCOR      applies specified corrections to CS tables
CVEL       shifts spectral-line UV data to a given velocity
DECOR      Measures the decorrelation between channels and IF of uv data
DOBAND     specifies if/how bandpass calibration is applied
DOCALIB    specifies whether a gain table is to be applied or not
DODELAY    selects solution for phase/amplitude or delay rate/phase
DOPOL      selects application of any polarization calibration
FARAD      add ionospheric Faraday rotation to CL table
FLAGVER    selects version of the flagging table to be applied
FQTOL      Frequency tolerance with which FQ 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
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
GAPLT      plots GAin table (ASCAL) by antenna, several per page
GETJY      determines calibrator flux densities
GNMRG      merges gain files of ASCAL sort
GNPLT      plots extrema of gain solutions from ASCAL
HLPIBLED   Interactive Baseline based visibility Editor - internal help
HLPSPFLG   Interactive time-channel visibility Editor - internal help
HLPTVFLG   Interactive time-baseline visibility Editor - internal help
```

IBLED	Interactive BaseLine based visibility EDitor
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 gain interpolation function
LISTR	prints contents of UV data sets and assoc. calibration tables
MAPBM	Map VLA beam polarization
MBDLY	Fits multiband delays from IF phases, updates SN table
MULTI	Task to convert single-source to multi-source UV data
PCAL	Determines instrumental polarization for UV data
POLSN	Make a SN table from cross polarized fringe fit
PRTGA	prints contents of an ASCAL gain file
REFANT	Reference antenna
RFI	Look for RFI in uv data
SCMAP	Imaging plus self calibration loop
SELBAND	Specified bandwidth
SELFREQ	Specified frequency
SETJY	Task to enter source info into source (SU) table.
SHOUV	displays uv data in various ways.
SMODEL	Source model
SMOTYPE	Specifies smoothing
SNCOR	applies user-selected corrections to the calibration SN table
SNPLT	Plots selected contents of SN, TY or CL file
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
SOLMODE	Solution mode
SOLTYPE	Solution type
SPFLG	interactive flagging of UV data in channel-TB using the TV
SPLIT	converts multi-source to single-source UV files w calibration
TASAV	Task to copy all extension tables to a dummy uv-file
TIMSMO	Specified smoothing times
TRIANGLE	specifies closure triangles to be selected/deselected
TVFLG	interactive flagging of UV data using the TV
UNCAL	sets up tables for uncalibrating Australia Telescope data
USUBA	Convert a subset of uv data to a specified subarray
UVCRS	Finds the crossing points of UV-ellipses.
UVFLG	Flags UV-data
UVFND	prints selected data from UV data set to search for problems
UVPRT	prints data from a UV data base with calibration
UVSRT	Sort a UV dataset into a specified order
UVWTFN	Specify weighting function, Uniform or Natural
VBCAL	Scale visibility amplitudes by antenna based constants
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
VLAMODE	VLA observing mode
VLAOBS	Observing program or part of observer's name
VLARESET	Reset calibration tables to a virginal state
VSCAL	Perform self-calibration on visibility data
WTUV	Specifies the weight to use for UV data outside UVRANGE

15.12. GENERAL

List of verbs, adverbs, tasks in category 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
APARM	General numeric array adverb used many places
BADDISK	specifies which disks are to be avoided for scratch files
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
CLRMSG	deletes messages from the user's message file
COMMENT	64-character comment string
CPARM	general numeric array adverb used many places
DDISK	Deterimins where input DDT data is found
DDT	verifies correctness and performance using standard problems
DDTSAVE	verifies correctness and performance using standard problems
DDTSIZE	Deterimins which type of DDT is RUN.
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
DOT	verifies correctness and performance using standard problems
DOTSAVE	verifies correctness and performance using standard problems
DOWAIT	selects wait-for-completion mode for running tasks
DPARM	General numeric array adverb used many places
ECOUNT	give the highest count or iteration for some process
EDGSKP	Deterimins border excluded from comparision during a DDT RUN.
EXPLAIN	displays help + extended information describing a task/symbol
FREESPAC	displays available disk space for AIPS in local system
GET	restores previously SAVEd full POPS environment
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
HIEND	First record number in a print operation
HINOTE	adds user-generated lines to the history extension file
HISTART	First record number in a print operation
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
IOTAPE	Deterimins which tape drive is used during a DDT RUN
LSAPROPO	Data input to APROPO to find what uses what words
MAPDIF	Records differences between DDT test results and standards

```
VLAL      verifies correctness of spectral line calibration software
VLALSAVE  verifies correctness of continuum calibration
WHATSNEW  lists changes and new code in the last several AIPS releases
XPARM     General adverb for up to 10 parameters, may refer to X coord
XTYPE     Specify type of process, often the X axis type of an image
YINC      Y axis increment
YTYPE     Y axis (V) convolving function type
```

15.13. HARDCOPY

List of verbs, adverbs, tasks in category HARDCOPY

```
FACTOR    scales some display or CLEANing process
FLMCOMM   Comment for film recorder image.
HIEND     First record number in a print operation
HISTART   First record number in a print operation
ISPEC     Plots and prints spectrum of region of a cube
OUTFILE   specifies name of output disk file, not in regular catalog
OUTPRINT  specifies name of disk file to keep the printer output
PRIORITY  Limits prioroty of messages printed
PRNUMBER  POPS number of messages
PRSTART   First record number in a print operation
PRTASK    Task name selected for printed information
PRTIME    Time limit
PRTLEV    Specified the amount of information requested.
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.
UVPRT     prints data from a UV data base with calibration
```

15.14. IMAGE-UT

List of verbs, adverbs, tasks in category IMAGE-UTIL

```
BDROP     gives number of pooints dropped at the beginning
DOMODEL   selects display of model function
DORESID   selects display of differences between model and data
DOSLICE   selects display of slice data
EDROP     number of points/iterations to be omitted from end of process
HLPTVHUI  Interactive intensity-hue-saturation display - on-line help
HLPTVRGB  Interactive red-green-blue display - on-line help
OHGEO     Geometric interpolation with correction for 3-D effects
```

15.15. IMAGE

List of verbs, adverbs, tasks in category IMAGE

```
MAPBM     Map VLA beam polarization
PROFL     Generates plot file for a profile display.
STRAN     Task compares ST tables and find image coordinates
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 coodinates
```

15.16. IMAGING

List of verbs, adverbs, tasks in category IMAGING

AHIST	Task to convert image intensities by adaptive histogram
APCLN	Deconvolves images with CLEAN algorithm
APGS	deconvolves image with Gerchberg-Saxton algorithm
APVC	Deconvolves images with van Cittert algorithm
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
BPA	gives position angle of major axis of beam or component
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
CELLSIZE	gives the pixel size in physical coordinates
COMAP	Procedure to MAP and Self-Calibrate a UVDATA set
COMAP_DO	MX adverbs not changed by COMAP
COMAP_MX	MX adverbs not changed by COMAP
COMAP_NA	Procedure to MAP and Self-Calibrate a UVDATA set
COMAP_UV	Procedure to MAP and Self-Calibrate a UVDATA set
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
DCONV	deconvolves a gaussian from an image
DECSHIFT	gives Y-coordinate shift of an image center from reference
DOGRIDCR	selects correction for gridding convolution function
DTSUM	Task to provide a summary of the contents of a dataset
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
FLDSIZE	specifies size(s) of images to be processed
FLUX	gives a total intensity value for image/component or to limit
GAIN	specifies loop gain for deconvolutions
GRIDR	makes an image from single-dish data
GUARD	portion of UV plane to receive no data in gridding
HISEQ	task to translate image by histogram equalization
HORUS	makes images from unsorted UV data, applying any calibration
IM2UV	converts an image to a visibility data set
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.
LTESS	makes mosaic images by linear combination
MANDL	creates an image of a subset of the Mandlebrot Set
MAPIT	Procedure to MAP and Self-Calibrate a UVDATA set
MAPIT_MX	MX adverbs not changed by MAPIT
MAPIT_NA	Procedure to MAP and Self-Calibrate a UVDATA set

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

15.17. INFORMAT

List of verbs, adverbs, tasks in category INFORMATION

ADVERB	Obsolete - used to list adverb POPS language symbols
ANALYSIS	Obsolete - used to list image process, analyze, combine
APTASKS	Obsolete - used to list tasks using array/vector processors
BATCHJOB	Obsolete - used to describe using AIPS batch jobs
CALIBRAT	describes the process of data calibration in AIPS
CATEGORY	List of allowed primary keywords in HELP files
CATINFO	obsolete - used to list data directory related things
CUBE	obsolete - used to list spectral-line related tasks and verbs
CURSOR	obsolete - used to list verbs/tasks using the TV and TEK
DELETE	obsolete - used to list ways files are deleted from AIPS
DOTASK	obsolete - used to list operations related to running tasks
GENERAL	obsolete - listed general utilities, mostly verbs & pseudos
INDEX	Obsolete - used to list all verbs and tasks alphabetically
LSAPROPO	Data input to APROPO to find what uses what words
MAPETC	obsolete - used to list image making, deconv., self-cal tasks
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
PL2D	obsolete - used to list info about 2-D displays and plots
POPSDAT	lists all POPS symbols, used to create them in MEmory files
POPSYM	Describes the symbols used in POPS
PSEUDO	Description of POPS pseudoverbs - obsolete list file
SECONDARY	List of allowed secondary keywords in HELP files
SECONDRY	List of allowed secondary keywords in HELP files
SL1D	Obsolete - used to list info about 1-D slices
TAPE	Obsolete - used to list info about reading or writing tapes
TASKS	Information about tasks - obsolete list file
TEKSERVER	Information about the message server
TEKSRV	Information about the message server
TVCOLOR	Obsolete - used to list info about color TV displays
TVGEN	Obsolete - used to list info about AIPS TV display
TVINTER	Obsolete - used to list info about interactive use of the TV
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
UVPR	Obsolete - used to list UV software in AIPS
VLBI	Obsolete - used to list software to handle VLBI and VLBA data
WHATSNEW	lists changes and new code in the last several AIPS releases
XAS	Information about TV-Servers
XVSS	Information about older Sun OpenWindows-specific TV-Server

15.18. INTERACT

List of verbs, adverbs, tasks in category INTERACTIVE

AIPS	AIPS main program for interactive use
HLPIBLED	Interactive Baseline based visibility Editor - internal help
HLPSPFLG	Interactive time-channel visibility Editor - internal help
HLPTVFLG	Interactive time-baseline visibility Editor - internal help
HLPTVHUI	Interactive intensity-hue-saturation display - on-line help
IBLED	Interactive BaseLine based visibility EDitor
OPTELL	The operation to be passed to a task by TELL
READ	Read a value from the users terminal
SPFLG	interactive flagging of UV data in channel-TB using the TV
TK1SET	Verb to reset 1D gaussian fitting initial guess.
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
TVFLG	interactive flagging of UV data using the TV
TVSCROL	Shift position of image on the TV screen
TVSLICE	Set slice endpoints on the TV interactively
TVSPLIT	Compare two TV image planes, showing halves
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
WEDERASE	Load a wedge portion of the TV with zeros

15.19. MODELING

List of verbs, adverbs, tasks in category MODELING

DOMODEL	selects display of model function
DORESID	selects display of differences between model and data
GLENS	models galaxy gravitational lens acting on 3 component source
MODVF	task to create a warped velocity field
SLFIT	Task to fit gaussians to slice data.
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
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.
XGAUS	Fits 1-dimensional Gaussians to images

15.20. OBSOLETE

List of verbs, adverbs, tasks in category OBSOLETE

ABACKUP	VMS procedure to back up data on tape
ADVERB	Obsolete - used to list adverb POPS language symbols
ANALYSIS	Obsolete - used to list image process, analyze, combine
APTASKS	Obsolete - used to list tasks using array/vector processors

ARESTORE	Restores back up tapes of users data
BATCHJOB	Obsolete - used to describe using AIPS batch jobs
CATINFO	obsolete - used to list data directory related things
CODETYPE	specifies the desired operation type
CUBE	obsolete - used to list spectral-line related tasks and verbs
CURSOR	obsolete - used to list verbs/tasks using the TV and TEK
DELETE	obsolete - used to list ways files are deleted from AIPS
DOTASK	obsolete - used to list operations related to running tasks
GENERAL	obsolete - listed general utilities, mostly verbs & pseudos
INDEX	Obsolete - used to list all verbs and tasks alphabetically
MAPETC	obsolete - used to list image making, deconv., self-cal tasks
PFT	The Perley-Feigelson Test; see PFTLOAD.RUN, PFTEXEC.RUN
PHCLN	PHCLN has been removed, use PHAT adverb in APCLN.
PL2D	obsolete - used to list info about 2-D displays and plots
PSEUDO	Description of POPS pseudoverbs - obsolete list file
SAMPTYPE	Specifies sampling type
SL1D	Obsolete - used to list info about 1-D slices
SNCUT	Specifies minimum signal-to-noise ratio
STARPOS	Text file name
TAPE	Obsolete - used to list info about reading or writing tapes
TASKS	Information about tasks - obsolete list file
TVCOLOR	Obsolete - used to list info about color TV displays
TVGEN	Obsolete - used to list info about AIPS TV display
TVINTER	Obsolete - used to list info about interactive use of the TV
UVPR	Obsolete - used to list UV software in AIPS
VLBI	Obsolete - used to list software to handle VLBI and VLBA data
XVSS	Information about older Sun OpenWindows-specific TV-Server

15.21. ONED

List of verbs, adverbs, tasks in category ONED

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.
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
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
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
XGAUS	Fits 1-dimensional Gaussians to images
XPLOT	Plots image rows one at a time on the graphics screen

```
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
****************************************************************
List of PSEUDOVERBs:

ABORTASK   stops a running task
ARRAY      Declares POPS symbol name and dimensions
COMPRESS   recovers unused POPS address space - not implemented
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
```

15.30. RUN

```
List of verbs, adverbs, tasks in category RUN

DDT        verifies correctness and performance using standard problems
DDTSAVE    verifies correctness and performance using standard problems
DOT        verifies correctness and performance using standard problems
DOTSAVE    verifies correctness and performance using standard problems
VLAC       verifies correctness of continuum calibration software
VLACSAVE   verifies correctness of continuum calibration
VLAL       verifies correctness of spectral line calibration software
VLALSAVE   verifies correctness of continuum calibration
```

15.31. SINGLEDI

```
List of verbs, adverbs, tasks in category SINGLEDISH

CSCOR      applies specified corrections to CS tables
GRIDR      makes an image from single-dish data
PRTSD      prints contents of AIPS single-dish data sets
SDCAL      Task to apply single dish calibration
SDTUV      Task to convert SD table files to UV like data.
SELSD      Task to select random position single dish measurments
```

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
BLCAL	Compute closure offset corrections
BLING	fringe fit residual rate and delay on individual baselines
BLOAT	converts pseudo-continuum to proper line UV data set
BLSUM	sums images over irregular sub-images, displays spectra
BLWUP	Blow up an image by any positive integer factor.
BPASS	computes spectral bandpass correction table
BPCOR	Correct BP table.
BSMAP	images weak sources with closure phases
CALIB	determines antenna calibration: complex gain
CANDY	user-definable (paraform) task to create an AIPS image
CANPL	translates a plot file to a Canon printer/plotter
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
CCSEL	Select signifigant CC components
CL2HF	Convert CL table to HF table
CLCAL	merges and smoothes SN tables, applies them to CL tables
CLCOR	applies user-selected corrections to the calibration CL table
CLIP	deletes UV data with amplitudes outside specified range
CLPLT	plots closure phase and model from CC file
CLSMO	smooths a calibration CL table
CNTR	generate a contour plot file or TV plot from an image
COMAP	Procedure to MAP and Self-Calibrate a UVDATA set
COMAP_DO	MX adverbs not changed by COMAP
COMAP_MX	MX adverbs not changed by COMAP
COMAP_NA	Procedure to MAP and Self-Calibrate a UVDATA set
COMAP_UV	Procedure to MAP and Self-Calibrate a UVDATA set
COMB	combines two images by a variety of mathematical methods
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
CSCOR	applies specified corrections to CS tables
CVEL	shifts spectral-line UV data to a given velocity
CXCLN	Complex Hogbom CLEAN
DAYFX	Fixes day number problems left by FILLM
DBCON	concatenates two UV data sets
DCONV	deconvolves a gaussian from an image
DECOR	Measures the decorrelation between channels and IF of uv data
DESCM	copies a portion of a UV data set
DFTPL	plots DFT of a UV data set at arbitrary point versus time
DIFRL	divides the RR data by LL data
DISKU	shows disk use by one or all users
DTSUM	Task to provide a summary of the contents of a dataset
FARAD	add ionospheric Faraday rotation to CL table
FETCH	Reads an image from an external text file.
FFT	takes Fourier Transform of an image or images
FILLM	reads VLA on-line/archive format uv data tapes (post Jan 88)
FILLR	reads old VLA on-line-system tapes into AIPS
FITLD	reads tape to load FITS images or FITS UV files to disk

MCUBE	collects n-dimensional images into n+1-dimensional image
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
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)
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
MX	makes images and deconvolves using UV data directly.
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
OHGEO	Geometric interpolation with correction for 3-D effects
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 pattern.
PBCOR	Task to apply the primary beam correction
PCAL	Determines instrumental polarization for UV data
PCNTR	Task to generate plot file for contour plus pol. vectors
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
PHCLN	PHCLN has been removed, use PHAT adverb in APCLN.
PHSRF	Perform phase-referencing within a spectral line database.
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.
POLCO	Task to correct polarization maps for Ricean bias
POLSN	Make a SN table from cross polarized fringe fit
POSSM	Task to plot total and cross-power spectra.
PROFL	Generates plot file for a profile display.
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.
PRTGA	prints contents of an ASCAL gain file
PRTIM	prints image intensities from an MA catalog entry
PRTPL	Task to send a plot file to the line printer
PRTSD	prints contents of AIPS single-dish data sets
PRTTP	prints contents of tapes, all supported formats
PRTUV	prints contents of a visibility (UV) data set
QMSPL	Task to send a plot file to the QMS printer/plotter
QUACK	Flags beginning or end portions of UV-data scans
REGRD	Regrids an image from one co-ordinate frame to another
REMAG	Task to replace magic blanks with a user specified value
RFI	Look for RFI in uv data
RGBMP	Task to create an RGB image from the 3rd dim of an image
RM	Task to calculate rotation measure and magnetic field
RSTOR	Restores a CC file to a map with a gaussian beam.
RTIME	Task to test compute times
SAD	fits gaussians to portions of an image
SBCOR	Task to correct VLBA data for phase shift between USB & LSB

TVHUI	make TV image from images of intensity, hue, saturation
TVHXF	Task to calculate transfer function based on histogram
TVPL	Display a plot file on the TV
TVRGB	make TV image from images of true color (RGB) images
TXPL	Displays a plot (PL) file on a terminal or line printer
UBAVG	Baseline dependent time averaging of uv data
UNCAL	sets up tables for uncalibrating Australia Telescope data
USUBA	Convert a subset of uv data to a specified subarray
UTESS	deconvolves images by maximizing emptiness
UV2MS	Append single source file to multisource file.
UV2TB	Converts UV autocorrelation spectra to tables
UVADC	Fourier transforms and corrects a model and adds to uv data.
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
UVCOP	Task to copy a subset of a UV data file
UVCRS	Finds the crossing points of UV-ellipses.
UVDGP	Copy a UV data file, deleting a portion of it
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
UVGLU	Glues UV data frequency blocks back together
UVHGM	Plots statistics of uv data files.
UVIMG	Grid UV data into an "image"
UVLIN	Fits and removes continuum visibility, also can flag
UVLOD	Read export or FITS data from a tape or disk
UVLSF	least squares fit to channels and subtracts from uv data.
UVMAP	makes images from calibrated UV data.
UVMOD	Modify UV database by adding a model or models
UVMTH	Averages one data set and applied it to another.
UVNOU	flags uv samples near the V (U=0) axis to reduce interference
UVPLT	plots data from a UV data base
UVPOL	modifies UV data to make complex image and beam
UVPRM	plots data from a UV data base
UVPRT	prints data from a UV data base with calibration
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
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
VBPLT	plots uv data and model from CC file
VLABP	VLA antenna beam polarization correction for snapshot images
VLBIN	Task to read VLBI data from an NRAO/MPI MkII correlator
VSCAL	Perform self-calibration on visibility data
VTESS	Deconvolves sets of images by the Maximum Entropy Method
WARP	Model warps in Galaxies
WFCLN	Wide field and/or widefrequency CLEANing/imaging task.
WTMOD	modifies weights in a UV data set
XBASL	Fits and subtracts nth-order baselines from cube (x axis)
XGAUS	Fits 1-dimensional Gaussians to images

XMOM	Fits one-dimensional moments to each row of an image
XPLOT	Plots image rows one at a time on the graphics screen
XSMTH	Smooth data along the x axis
XSUM	Sum or average images on the x axis
XTRAN	Create an image with transformed coodinates

15.36. TV-APPL

List of verbs, adverbs, tasks in category TV-APPL

BLSUM	sums images over irregular sub-images, displays spectra
GAMMASET	changes the gamma-correction exponent used in the TV OFM
HLPIBLED	Interactive Baseline based visibility Editor - internal help
HLPSPFLG	Interactive time-channel visibility Editor - internal help
HLPTVFLG	Interactive time-baseline visibility Editor - internal help
HLPTVHUI	Interactive intensity-hue-saturation display - on-line help
HLPTVRGB	Interactive red-green-blue display - on-line help
IBLED	Interactive BaseLine based visibility EDitor
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 an 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
OFMTWEAK	interactive modification of current TV OFM lookup tables
OFMZAP	deletes an OFM lookup table save file
SPFLG	interactive flagging of UV data in channel-TB using the TV
TVFLG	interactive flagging of UV data using the TV

15.37. TV

List of verbs, adverbs, tasks in category TV

BLANK	blanks out selected, e.g. non-signal, portions of an image
CNTR	generate a contour plot file or TV plot from an image
COLORS	specifies the desired TV colors
CURBLINK	switch TV cursor between steady and blinking displays
CURVALUE	displays image intensities selected via the TV cursor
DOTV	selects use of TV display option in operation
FACTOR	scales some display or CLEANing process
GRCHAN	specifies the TV graphics channel(s) 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	reads the colors of the specified TV graphics channel
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
NBOXES	Number of boxes
NCCBOX	Number of clean component boxes

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
TVRGB	make TV image from images of true color (RGB) images
TVROAM	Load up to 4 TV image planes and roam a subset thereof
TVSCROL	Shift position of image on the TV screen
TVSLICE	Set slice endpoints on the TV interactively
TVSPLIT	Compare two TV image planes, showing halves
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
XVSS	Information about older Sun OpenWindows-specific TV-Server

15.38. UTILITY

List of verbs, adverbs, tasks in category UTILITY

CCEDT	Select CC components in BOXes and above mininum flux.
CCSEL	Select signifigant CC components
CL2HF	Convert CL table to HF table
MBDLY	Fits multiband delays from IF phases, updates SN table
MK3TX	extract text files from a MKIII VLBI archive tape
OPCODE	General adverb, defines an operation
OPTELL	The operation to be passed to a task by TELL
PRNUMBER	POPS number of messages
PRTIME	Time limit
SHOW	Verblike to display the TELL adverbs of a task.
SORT	Specified desired sort order
SQASH	Task to sum together or average planes in a cube
STRAN	Task compares ST tables and find image coordinates
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
UVNOU	flags uv samples near the V (U=0) axis to reduce interference

15.39. UV

List of verbs, adverbs, tasks in category UV

ADDIF	Adds an IF axis to a uv data set
AFILE	sorts and edits MkIII correlator A-file.
AVER	Averages over time UV data sets in 'BT' order
AVSPC	Averages uv-data in the frequency domain
BAND	specifies the approximate frequency of UV data to be selected
BIF	gives first IF to be included
BLOAT	converts pseudo-continuum to proper line UV data set
CLIP	deletes UV data with amplitudes outside specified range
CLPLT	plots closure phase and model from CC file
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
DAYFX	Fixes day number problems left by FILLM
DBCON	concatenates two UV data sets
DECOR	Measures the decorrelation between channels and IF of uv data
DESCM	copies a portion of a UV data set
DFTPL	plots DFT of a UV data set at arbitrary point versus time
DIFRL	divides the RR data by LL data
DOARRAY	spcifies if subarrays are ignored or the information used
DOCONCAT	selects concatenated or indivudual output files
DOSTOKES	selects conversion from observed polarizations to Stokes
DOUVCOMP	selects use of compression in writing UV data to disk
DTSUM	Task to provide a summary of the contents of a dataset
EIF	last IF number to be included in operation
FILLM	reads VLA on-line/archive format uv data tapes (post Jan 88)
FILLR	reads old VLA on-line-system tapes into AIPS
FRPLT	Task to plot fringe rate spectra
FUDGE	modifies UV data with user's algorithm: paraform task
HLPIBLED	Interactive Baseline based visibility Editor - internal help
HLPSPFLG	Interactive time-channel visibility Editor - internal help
HLPTVFLG	Interactive time-baseline visibility Editor - internal help
HOLGR	Read and process Holography visibility data
IBLED	Interactive BaseLine based visibility EDitor
IM2UV	converts an image to a visibility data set
LISTR	prints contents of UV data sets and assoc. calibration tables
MAPBM	Map VLA beam polarization
MK3IN	translate Haystack MKIII VLBI format "A" tapes into AIPS
MULIF	Change number of IFs in output
MULTI	Task to convert single-source to multi-source UV data
OBJECT	The name of an object
PCAL	Determines instrumental polarization for UV data
PHASE	Baseline Phase coherence measurement
PHSRF	Perform phase-referencing within a spectral line database.
POSSM	Task to plot total and cross-power spectra.
PRTAN	prints the contents of the ANtenna extension file
PRTUV	prints contents of a visibility (UV) data set
QUACK	Flags beginning or end portions of UV-data scans
QUAL	Source qualifier
REWEIGHT	Reweighting factors for UV data weights.
SBCOR	Task to correct VLBA data for phase shift between USB & LSB
SETAN	Reads an ANtenna file info from a text file

```
UVTAPER    Widths in U and V of gaussian weighting taper function
UVWTFN     Specify weighting function, Uniform or Natural
VBGLU      Glues together data from multiple passes thru the VLBA corr.
VBPLT      plots uv data and model from CC file
VLBIN      Task to read VLBI data from an NRAO/MPI MkII correlator
WTMOD      modifies weights in a UV data set
WTUV       Specifies the weight to use for UV data outside UVRANGE
ZEROSP     Specify how to include zero spacing fluxes in FT of UV data
```

15.40. 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.
************************************************************
List of verbs, adverbs, tasks in category VERB

ABOUT      displays lists and information on tasks, verbs, adverbs
ABS        returns absolute value of argument
ADDBEAM    Inserts clean beam parameters in image header
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
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
BATNLINE   specifies the number of lines to process in a batch work file
BY         gives increment to use in FOR loops in POPS language
CATALOG    list one or more entries in the user's data directory
```

```
CEIL       returns smallest integer greater than or equal the argument
CELGAL     switches header between celestial and galactic coordinates
CHAR       converts number to character string
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
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
COS        returns cosine of the argument in degrees
CURBLINK   switch TV cursor between steady and blinking displays
CURVALUE   displays image intensities selected via the TV cursor
DISMOUNT   disables a magnetic tape and dismounts it from the tape drive
DUMP       displays portions of the POPS symbol table in all formats
EGETNAME   fills in input name adverbs by catalog slot number, w error
END        marks end of block (FOR, WHILE, IF) of POPS code
EPOSWTCH   Switches between B1950 and J2000 coordinates in header
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
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
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
GETTHEAD   returns keyword and other values value from a table header
GO         starts a task, detaching it from AIPS or AIPSB
GRADDRES   specifies user's home address for replies to gripes
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     reads 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
IMERASE    replaces an image portion of the TV screen with zeros
IMHEADER   displays the image header contents to terminal, message file
IMPOS      displays celestial coordinates selected by the TV cursor
IMSTAT     returns statistics of a sub-image
IMVAL      returns image intensity at specified pixel
IMWEDGE    load step wedge of full range of image values to TV
IMXY       returns pixel coordinates selected by the TV cursor
```

CROSSPOL	Procedure to make complex poln. images and beam.
CVEL	shifts spectral-line UV data to a given velocity
CXPOLN	Procedure to make complex poln. images and beam.
FRING	fringe fit data to determine antenna calibration, delay, rate
FRPLT	Task to plot fringe rate spectra
HF2SV	convert HF tables from FRING/MBDLY to form used by SOLVE
HLPIBLED	Interactive Baseline based visibility Editor - internal help
HYB	Procedure to run a hybrid mapping loop
IBLED	Interactive BaseLine based visibility EDitor
MBDLY	Fits multiband delays from IF phases, updates SN table
MK3IN	translate Haystack MKIII VLBI format "A" tapes into AIPS
MK3TX	extract text files from a MKIII VLBI archive tape
POLSN	Make a SN table from cross polarized fringe fit
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
VBPLT	plots uv data and model from CC file
VLBA	Procedure to read and process VLBA data (Phil Diamond)
VLBIN	Task to read VLBI data from an NRAO/MPI MkII correlator

architectures. Thus the AIPS programming group tries to avoid writing microcode. But portions of the AIPS *tasks* for mapmaking, 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.

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 approximants 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*.

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 $f * g = h + \text{noise}$, 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:

$$x_j = \frac{1}{n}\sum_{k=0}^{n-1} y_k e^{-2\pi ijk/n}.$$

(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 × 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 $\mathbf{R}^n \to \mathbf{R}^3$ 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 $x_0,\ldots,x_{n-1}$,

$$y_k = \sum_{j=0}^{n-1} x_j e^{2\pi ijk/n},$$

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

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

$$\tilde{V}_{ij}(t,\nu) = a_i a_j V_{ij}(t_0,\nu_0) \times e^{\sqrt{-1}((\psi_i-\psi_j)(t_0,\nu_0)+(\tau_i-\tau_j)(t-t_0)+(\tau_i-\tau_j)(\nu-\nu_0))},$$

where V_{ij} is the true visibility, and where the τ_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 τ_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 mapmaking, 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 pointwise 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, pseudoverb, 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.

Högbom 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 multimodal 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 overdetermined 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 I²S 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*, nonnegativity, 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^{2}S — (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 × 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^{2}S 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 $\hat{f}$—is given by the negative of the Shannon entropy

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 × 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 × 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, pseudoverb,* 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 *pseudoverb* 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

$$(*) \qquad \nu f(\eta) = \int_{-1}^{1} e^{ic\eta t} f(t)\, dt\,;$$

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 $(*)$ 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

$$\frac{\iint_{\text{map}} |\hat{C}(x,y)|^2\,dx\,dy}{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} |\hat{C}(x,y)|^2\,dx\,dy}$$

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 $\iint_{\substack{\text{clean}\\ \text{window}}} [f(x,y)]^2\,dx\,dy$ 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.

pseudoverb — 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 nonnegativity, 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 $\lambda \rightarrow 0$. 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

$$S(\mathbf{g}) = \left(\sum_{1 \le i < j \le n} w_{ij} \left| \left\langle \tilde{V}_{ij}/V_{ij} \right\rangle - g_i \overline{g}_j \right|^p \right)^{1/p} ,$$

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 $t_1 < t_2 < \cdots < t_l$ 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^{2}S*, 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^{2}S* 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^{2}S 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^{2}S* 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^{2}S 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^{2}S*. On the I^{2}S, the three available magnification factors (which multiply the linear dimensions of the original display of the image by a

Appendix Z. SYSTEM-DEPENDENT *AIPS* TIPS

Although *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, the parameters of television and array processor devices) 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 *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 *AIPS* on workstations was given in Chapter 2 and will not be repeated here.

Within the NRAO, *AIPS* is installed on two main architectures — Sun workstations and IBM RS/6000 systems. All our old CONVEX C-1 and DEC VAX 11/750 and 11/780 systems have been decommissioned. Currently the fastest *AIPS* machines in the Observatory are the IBM/RS6000 model 580 systems **rhesus** in Charlottesville, and **kiowa** and its clones at the Array Operations Center. *AIPS* also runs in Charlottesville on a DEC Alpha workstation and on a 486/66 personal computer. The former is available for general use and would be an interesting computer if it had more memory.

Z.1. NRAO workstations — general information

All NRAO workstations run some version of the Unix operating system, AIX on IBMs and SunOS (or Solaris) on Suns. 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. *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 *AIPS* on all NRAO Sun, IBM, DEC, and Intel systems are kept up to date continually with the master versions on the Charlottesville Sun called **baboon**. 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 *AIPS* are copied to the relevant computers and recompiled/relinked. At the time of writing, midnight jobs run on **rhesus** (CV IBM), **tamarin** (CV Solaris 2.3), **tarsier** (CV Intel Linux), **pongo** (CV Dec Alpha), **aguila** (AOC Sun), **laphroaig** (AOC Solaris 2.3), **kiowa** (AOC IBM), and **miranda** (VLA Sun). At "quarterly" (*sic*) update time, **NEW** is also updated via the midnight jobs while it is still unfrozen.

There are at least two sites outside of the NRAO that are also served by the midnight job and this number may be greater by the time you read this.

`ftp>` put *file_name* ↵ to copy *file_name* from your computer to the Solitaire slides area.

`ftp>` quit ↵ to exit and log out.

Note that *file_name* must be a file in PostScript format suitable for the film recorder, such as the files produced by the *AIPS* task **TVCPS**. To assist Pat in previewing your picture, its name should be no longer than eight characters plus an extension of no more than three characters, all preferably in lower case.

Having copied the file, you should then send Pat an e-mail message (**psmiley@nrao.edu**) telling her your name, postal and e-mail addresses, file name, any special requirements you and your picture may have, and whether you would be willing to allow Pat to include your picture in the NRAO image library. There are a number of reasons to ask Pat for this service, even if you are at the AOC. She is able to get 35mm slides processed in one day and mailed off, if needed, by Federal Express the next day. Special requests, such as 4x5 film for very detailed (*e.g.*, multi-panel) images, larger format overheads to bring out special detail or Black-and-white negative films for non-color prints, can be handled a bit more slowly. Pat cleans the film recorder before making pictures, giving the best possible quality in the photos. (The devices have been found to be "dusty" intrinsically.) You may choose to allow Pat to keep your images in the NRAO image library. If you do, she will need extra information from you about the image and the NRAO telescope used, as well as a release date and a statement allowing NRAO to distribute the image. Besides giving your image and science greater currency and helping the NRAO in its publicity and educational rôles, leaving your image in the NRAO image library will allow you to request duplicates and prints at any time at no charge.

Z.1.3. Gripe, gripe, gripe, ...

With the effort currently going into AIPS++, the "designated AIP" program has been discontinued mainly due to lack of people and time. At the AOC in Socorro, contact the *AIPS* Manager if you have a problem. Likewise in Charlottesville.

Suggestions and complaints entered on NRAO computers with the **GRIPE** verb (see § 12.3) are collected regularly and, at least, read. The most urgent are addressed and, sometimes, answered. All gripes are entered into a database which resides on **zia.aoc.nrao.edu** (otherwise known as **146.88.1.4**). Users may read the contents of this database in as much detail as they can stand. To do so, login to the account called **gripe** on **zia**. This is a "captive" account, requiring no password, and allowing you only to execute an especially prepared version of the text editor **emacs**. When you tire and exit the special **emacs**, you will be logged out of **zia**.

After you log in, you will be presented with a selection/options menu. Fill in and/or alter some of the selection criteria to limit which gripes you will view. Then, select display option **index**, and, **only** when you are fully ready, hit a ↵. You will be shown a descriptive list of the selected gripes. If you wish to read one of them in detail, move the cursor to it and hit ↵. The space bar gets you the text of the next gripe and typing the letter **q** returns you to the index. Another **q** returns you to the selection/option form. Typing a **?** in any of the displays will provide you with information on all the options available at that level of the system.

AIPS Memo No. 88 describes the system in some detail. This memo may be available on your *AIPS* system as file **$AIPSPUBL/AIPSMEMO88.PS** in PostScript form. It is also available to the "World-Wide Web" (start with "URL" **http://info.cv.nrao.edu/aips/aips-home.html**) so that it may be examined and retrieved over the Internet. The file used is also available via anonymous **ftp** on host **baboon.cv.nrao.edu** (or **192.33.115.103**) as a PostScript file named **/pub/aips/TEXT/PUBL/AIPSMEMO88.PS**.

Z.2. AIPS at the NRAO in Charlottesville

The Charlottesville AIPS Caige is located in Room 111 on the first floor of the Edgemont Road Office Building. There are three public workstations available there: "**rhesus**'' an IBM RS/6000 580, "**ringtail**" an IBM RS/6000 560, and "**gibbon**" a Sun IPX. For normal plots and print jobs there is an HP LaserJet printer called **ps1** just outside the Caige in the corridor. This printer is on a slow serial line and prints on only one side of the paper. Large graphics plots from **LWPLA** should be sent to the network HP printer in the library (called **ps3**), while long print jobs should be (and will automatically be) sent to this printer in duplex mode (called **ps3dup**). There is a Tektronix 4511a color printer known as **pscolor** in the AIPS Caige for special plotting and color displays (*e.g.*, from **TVCPS**) on either paper or transparencies. Color slides may also be made by directing the output in PostScript form to **35mm2k**, a Solitaire film recorder across the hall.

The IBM workstations run under relatively current releases of AIX, IBM's version of the Unix operating system, while the Suns run under versions of SunOS, now called Solaris. They are equipped with color display screens, 1280 × 1024 on IBMs and 1152 × 900 on Suns. All display in 8-bit pseudocolor, although **ringtail** has the ability to display 24-bit "true" color images for applications that use it (not AIPS' **XAS**). Each system has substantial amounts of disk space. At this writing, **rhesus** has the most with 8 AIPS "disks" (12.2 Gbytes), **ringtail** has 4 (7.8 Gbytes), and **gibbon** has 3 (1.8 Gbytes). All three systems have at least one 4mm DAT and one 8mm Exabyte (low density) tape drive, while **rhesus** has two of each plus two 800/1600/6250 bpi 9-track tape drives.

Z.2.1. Using the Charlottesville workstations

Z.2.1.1. Signing up for AIPS time in Charlottesville

The sign-up sheet for AIPS on **rhesus** is found on the notice board just to the left of the entrance to the AIPS Caige, Room 111. If you wish to be certain of the availability of **rhesus**, it is advisable to sign up for your AIPS time in advance. To promote fair and efficient use of the system, users are asked to restrict the amount of time that they reserve. Formal rules are not now in effect, but may be imposed should the need arise.

AIPS on the RS/6000 systems supports up to eight simultaneous interactive users, plus two batch queues. The user signed up for **AIPS1** has priority for the use of the cpu, console (display), and local tape drives, but is expected to be reasonable about sharing these resources, particularly the tape drives. Visitors to Charlottesville should call Jim Condon, in advance of their arrival, to avoid conflicts with other visitors and to arrange for sign-up time.

Sign-up time for **ringtail** is no longer available, although you may use it on a *caveat emptor* basis. It is implicitly reserved for anyone wishing to use the advanced visualization facilities available through the *AVS* or *Data Explorer* software systems. If you are using AIPS on it, and someone needs to run either of these systems, you can be bumped off the system at that time.

Z.2.1.2. Managing workstation windows in Charlottesville

After you have logged on to any Sun or IBM as user-id **aips**, the X-Window system should appear. Unlike the AOC where there are different window managers for IBMs and Suns, at Charlottesville the **aips** account always uses the **twm** window manager (or possibly **tvtwm** which is a superset of **twm**). AIPS may also be run from your own account — we prefer that you do that — but your account must be included in the **aipsuser** group and your startup procedures must start some X-Windows window manager, preferably **twm** or **tvtwm**.

The window manager allows you to create, destroy, modify, and select an active window on the screen. When you first start up (in the **aips** account, there will be at least two **xterm** terminal emulator windows on the screen, one green and one blue. The green one is for console messages and is usually quite small. The blue one is intended to be the main work area and it is in this window that you probably will start up **AIPS** itself. There may be additional windows like a clock and an **xload** load meter.

The default behavior of the window manager is "focus follows pointer." What this means is that in order to use one of the windows, you merely move the cursor with the mouse so that it is within the main part of the window and start typing. Most windows will have a title bar on the top; you can "grab" this title bar by moving the mouse cursor onto it and holding and keeping down the left mouse button; then moving the mouse will also move the window. There is a small square symbol on the left of the title bar that, if pressed with the left mouse button, will iconify the window (you can grab and drag icons too). Clicking once with the left mouse button on any icon will de-iconify, or open it.

The mouse buttons bring up a series of menus when they are pressed on the background, or root window. The left mouse button shows various window operations such as "redraw screen", "raise", "move", and of course "Exit X11". The middle mouse button brings up a menu of useful programs, including a desk calculator, the **emacs** editor, **xv**, **xgrab**, and more. The right mouse button brings up the System menu; the most useful item on this is a "pull-right" option "X Terminal Emulator" which, when you press it and "pull right" by dragging the mouse a little to the right, shows another menu with different foreground/background colors. Choose one of these and you will get an **xterm** terminal emulator with those color combinations. There is also a pull-right menu for remote login to various other useful machines in Charlottesville and other NRAO sites.

To start **AIPS**, choose the **xterm** window that you want to work with (or create a new one), and type, *e.g.*, from **rhesus**

```
aips tst pr=3 da=ringtail tp=gibbon
```

This example command selects the TST version of **AIPS** (the default anyway), chooses printer number 3 (on the ground floor) as the default printer for text and graphics output, makes the data areas from **ringtail** accessible (via NFS) in addition to local data areas on **rhesus**, and makes sure that the **TPMON** daemons for remote tape access are running on remote host **gibbon**. These options are explained in some detail in § 2.2.3 and may be viewed in even greater detail by typing HELP AIPS $\subset_R$ inside **AIPS** or by typing man aips $\subset_R$ from the Unix command line.

Z.2.1.3. Data disk management in Charlottesville

A public 150-Mbyte disk partition on an IBM RS6000 580 called polaris was set aside for use as the main disk where message and **SAVE/GET** files are kept. *AIPS* looks for these files only on disk "1," whichever data area that happens to be. If the same disk area is always disk 1, no matter what computer you are using, then you would always get the same set of message and **SAVE/GET** files. The fly in this ointment is that the Network File System has to wait while disk reads and writes are completed, while reads and writes on local disks may be done asynchronously using large memory buffers in the Unix operating system. Thus, the writing of messages, in particular, is virtually instantaneous on local disks, but costs about one second per message over NFS.

The same consideration applies to disks used for image and *uv* data files. It is not too expensive to read such files over NFS, but you should only write data to disks on the computer you are using. You should also restrict all scratch files to be on local disks, using the adverb **BADDISK** to inhibit all NFS disks. Note, that the computer you are using does not have to be the one which you are sitting in front of. You may do an **rlogin** or **telnet** from an **xterm** window on, say, the Sun on your desk to, for example, **rhesus**.

the drive. When a tape is in the drive, it is visible from the outside (*i.e.*, the door does not come down and cover the tape) and the two lights will be on (continually lit if the tape is not being accessed; flashing if the tape is in use). If the drive is not in use, insert the tape, label up, and push it gently into the opening; the drive will eventually "grab" it and bring it in the rest of the way. With 4mm DAT tapes, the write-protection works differently from Exabytes; the slide goes to the right for writing (but white or red shows) and to the left for reading (black shows). When the green lights stop flashing, the tape is ready to be accessed.

Z.3.3. Color hard copy at the AOC

Having created a PostScript file containing color commands or pictures (see §§ Z.1.2.), you can print it on a Tektronix Phaser II PXi color printer known endearingly as **Farbdrucker** in the second- (ground-) floor computer room. Two queues have been set up for this printer; one for color paper (**pscolor**) and one for color overhead transparencies (**psoverhd**). Once the file has been routed to the appropriate queue using the **psprint** or **lpr** command, go to the terminal to the right of the color printer (labelled **Farbdrucker only!!!**). To log in, press the **F2** key for the **pscolor** queue, **F3** for the **psoverhd** queue. You will then be shown the print queue, asked if you want to route the files to the printer, and asked to check that the proper cartridge is in place (paper or transparencies). When the printer starts actually printing, it must move the paper in and out three times, one for each color. When it is done, it will eject the paper fully. Please note that color paper and transparencies are not cheap, so be reasonable in their use. To obtain additional paper or transparencies for empty cartridges, see Theresa McBride.

There is also a Solitaire film recorder (**Sol**) in the second floor computer room which is capable of making slides. Once an image is submitted to a Solitaire queue, NRAO staff will expose the slides and return them to you when they are developed. Due to Socorro logistics (the closest slide processor is in Albuquerque) it can take more than a week to get the finished slides returned to the NRAO. If you are leaving before the slides are due to be returned, contact Theresa McBride or Jon Spargo to arrange having the slides mailed to you. You should also consider sending the PostScript files to Pat Smiley in Charlottesville for processing and mailing; see § Z.1.2.3. Faster turn-around and special processing, if needed, are both available through Pat.

There are two ways to route images to the Solitaire queue using the *AIPS* task **TVCPS**. Once you have gotten the desired image set up in the **XAS** window, run TVCPS with a specified **OUTFILE** to save the PostScript file to disk. This allows you to look at the image using various PostScript previewers before sending it to the slide queues; it also makes it easier to save the images to tape. The saved file can then be sent to one of several slide queues (same as those listed in Charlottesville section Z.2.4, as well as **Pol**, which uses Polaroid film using 4096 scan lines). Once all the images are in the proper queues, contact Theresa McBride or Jon Spargo to expose the slides. This is the process to use to send the saved file to Charlottesville as well.

The step of saving the images to disk can be skipped by setting the **OUTFILE** disk logical to **SOL35**; this will route the output directly to the Solitaire **35mm4k** queue. See the **TVCPS** explain file or the memo "How to Make 35mm Slides From Your Images in AIPS" (available from Theresa McBride or within **Mosaic**) for more details.

Note that the Solitaire behaves as if it is printing on a piece of paper about 7.5 inches wide. This means that while the default parameters of **TVCPS** will center an image correctly on **Farbdrucker**, the image will be offset to the right on the Solitaire film recorder. See the **TVCPS** explain file for details on how to correct this.

Z.4. *AIPS* at the NRAO Very Large Array site

AIPS is installed on a Sun IPX workstation called **Miranda** in the control room at the VLA site. While there is not enough disk space to allow large-scale data reduction, there is adequate space to allow running on-line **FILLM** to inspect your data as it is acquired. **Miranda** is equipped with a 800-Mbyte data disk and a low-density Exabyte (8mm) tape drive. There is also a nine-track tape drive on the operator's Sun (**Banshee**) which can be used remotely, but check with the operator before using it. The *AIPS* QMS printer is located in the computer room next to the control room.

Z.4.1. Using the VLA workstation

Z.4.1.1. Signing up for *AIPS* time at the VLA

There is no official sign-up for visitor use of **Miranda**; if the workstation is not in use, feel free to use it. However, if there is more than one observing team at the site at any one time, priority goes to the current observer.

If you are planning on using on-line **FILLM**, you must contact George Martin (505–835–7287) prior to your arrival. It would also be wise to contact George or Gustaaf van Moorsel (505–835–7396) to get some documentation on on-line **FILLM** before heading out to the site.

Z.4.1.2. Managing workstation windows at the VLA

Miranda uses the **OpenWindows** system and will behave in a manner very similar to the AOC Suns. If the window system is not running, type **aips** at the login prompt, then enter the password. You will be asked if you wish to start **OpenWindows** — answer yes, and the window manager will begin. You will see several windows open up, currently some command tools, a clock, a performance meter, a console window, and a file manager. Further information on dealing with **OpenWindows** options can be found in § Z.3.1.2.2.

Z.4.1.3. Data disk management at the VLA

Unlike the AOC, there are no "optional" disks to be mounted on **Miranda**. The two data areas which appear when you start **aips** are actually just two partitions on the same disk.

If, when you start **aips**, you find little or no space on the disks, contact the *AIPS* Manager to clear off some space. Due to the limited size of the data disk, there is an automatic time destroy running on **Miranda** which will delete data which has not been used in three days. Sometimes even this interval is too long to keep a usable amount of space on the disk and the *AIPS* Manager is forced to take draconian measures. *Caveat emptor:* if you want a copy of the data on the **Miranda** disk, copy it onto a tape immediately!

Z.4.2. Using the tape drives at the VLA

For a general discussion of magnetic tapes, including the *required* software mount, see § 2.4. The following describes how to deal with the Exabyte drive at the VLA site.

The Exabyte drive at the VLA is a COCOMP drive. Unlike the COCOMP drives at the AOC, there is no display panel covering the window. Before opening the drive to insert your tape, make sure there is not already a tape in the drive. If there is a tape present, check with any other observers and the telescope operator to make sure that the tape is no longer in use.

To open the drive, push the button on the lower left of the tape door. If there was already a tape in the drive, it will be ejected after some whirring and clicking and a few seconds. If a tape is ejected, remove it. Exabyte 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. Now put your tape in the drive, label facing upwards; push the door closed gently.

It is necessary to wait until the mechanism in the drive has "settled down" before you can access the drive. On the COCOMP drive the flashing green light will stay lit (and the whirring and clicking will stop) when the tape is ready to be accessed. Once the green light on the drive has stopped flashing, the tape is ready for the software `MOUNT` command from inside `AIPS`.

On-line `FILLM` requires several "pseudo-tape" devices in the *AIPS* software, so the first "real" tape device on `Miranda` is tape number 6. Therefore, to access the local tape on `Miranda`, you need to use `INTAPE` (and `OUTTAPE` when needed) `= 6`. If you need to access one of the tapes on `Banshee` remotely, make sure to use one of the remote tape device numbers, either `INTAPE` 7 or 8.

Z.5. Additional recipe

Z.5.1. Delightful banana cheesecake

1. Preheat oven to 350°.
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.