NATIONAL RADIO ASTRONOMY OBSERVATORY
Green Bank, West Virginia

May 7, 1988

TO: Users Committee
FROM: Ronald J. Maddalena
SUBJECT: Data Reduction in Radio Astronomy

Attached are my views and philosophy on data reduction and analysis. They are my opinions and may not reflect those of other astronomers. What I have attempted to describe are the basis by which I judged the functionalities of the various software systems I have encountered. I have stressed the philosophy behind data reduction, as opposed to the mechanics, since I believe that the 'how' of data reduction follows almost directly from the 'why'.

I. Introduction

Data reduction is a process by which only the most salient numbers are extracted or deduced from a large set of numbers and are then displayed in formats such as graphs or tables. This process allows the experimentalist to "play" with data in order to look for trends and structure in the data and to summarize their data so that other scientist unfamiliar with the original data can also appreciate the meaning of the data. There is, then, two kinds of analysis. That which the experimentalist does for his or her own learning and that which the experimentalist does for the education of his or her peers.

Data reduction usually involves multiple steps which are chosen by the scientist according to what he or she feels is correct. The only laws governing how data is reduced are those dictated by the conventions used by the scientist's peers and are usually very loose. Thus, each scientist must be allowed to reduce data in a unique fashion. Whether or not the reduced data reflect what is present in the original data depends solely on the skills and honesty of the experimenter. Anything which unnecessarily increases the size of a set of numbers or severely restricts the kinds of process which can be performed on a set of numbers is against the practices of data reduction.

For very large sets of numbers, a computer is usually essential. But, in order not to violate the principles of data reduction, each scientist would necessarily have to write their own program so as to reduce data in their own personnel way. However, since some of the steps one scientist will use are identical to those used by others, every computer program need not be written from scratch. Instead, the scientist's program usually uses a subset of routines from a large library of data reduction software interspersed with original computer code. Some types of data reduction, however, are sometimes so unique that most of the computer code must be written from scratch.

The following sections describe the data reduction process as applied to single-dish observation in radio astronomy and are based on my experiences in data reduction and my perception of the kinds of data reduction performed by other astronomers, particularly those who use the NRAO telescopes. I will first describe the major steps taken in data reduction (Sec. II), followed by what should be included in a data reduction library (Sec. III). My philosophy on how one invokes library routines will be discussed in Section IV.

II. Steps in Data Reduction

Data reduction is usually a multi-step process. Many times a series of steps are tried but prove to be unsuccessful, inadequate to the needs of the scientists, or just plain incorrect. In these cases, the scientist must have ready access to the original data or data at some preliminary stage of reduction in order to try a different series of steps. Thus, any data reduction procedure must leave the original data intact and allow for the temporary storage of processed data at various stages of reduction. Although most times impossible, information about the processing previously performed on the data should be stored with the data. Good book keeping is an integral part of data reduction and can be done with pencil and paper by the scientist, or, more ideally, by the computer program used by the scientist.

In almost all cases in radio astronomy, the analog signal produced by a telescope are digitized and then passed to a computer. A series of data points, typically collected under a unique identifier called a scan number, are stored along with header information -- details about how the data was taken. Typically two type of scans are possible: Continuum or spectral line. After an observations has been performed, the data are usually stored on computer tapes or disk and the astronomer begins analyzing the stored data.

With today's faster radio telescopes with multiple backends or feeds and with large mapping projects, the use of scan numbers to identify observations becomes very awkward because of the large number of scans involved. Instead of having to address data by scan numbers, an alternative way would be to address scans by locations in the sky, name of the object, frequency, etc. For example, if an astronomer wanted to do some analysis on the observations of object XYZ, then he or she should only need to give the name XYZ to the computer and have it find the appropriate scan numbers.

Most observations go through three stages of data reduction.

Stage 1 -- A quick look

While observations are going on, the astronomer typically wants to quickly check whether the instrument is performing well. Most times the astronomer must decide on how to continue his or her experiment by looking at the results of the reduction performed on the data just collected. As with all subsequent stages in reduction, graphic and tabular summaries of the data, although possibly not of the highest quality at this stage, are needed to assist the astronomer in making a decision. The time between the completion of a scan and when the data can be looked

at must be kept as short as possible. Thus, some type of data reduction capability must be available to the astronomer during the time of the observations. Although the range of library functions needed by any one astronomer at this stage of data reduction will only be a subset of a more extensive library used for more complete data reduction, the full library should be provided to the astronomer since it is impossible to predict what subset of functions would satisfy even 50% of experiments. Even at this early stage of data reduction, a significant fraction (10 - 20%) of experiments require that their data be manipulated in a unique fashion which probably was not anticipated by the programmers who created the library.

Stage 2. -- "Playing" with and learning about the data

After the data has been taken, or during leisurely moments during the experiment, the astronomer starts the next stage of data reduction which is usually more carefully done and more extensive than the previous stage. Most times the astronomer does not use the reduced data from the first stage, which can be in error since the data was quickly reduced under hectic circumstances, but turns to the original data to insure that the reduction is done well and under calm circumstances. The astronomer typically uses mostly library functions but spends more time using software developed specially for that experiment. It is at this stage in the reduction process that the astronomer makes conclusions about the data. It is typically the trial and error stage in reduction, the "lets look at the data in this bizarre way" stage where the discoveries and revelations occur. Stage 2 is were most of the time is spent in data reduction.

While the previous stage of reduction usually involves an interactive program, some experiments would benefit if the second stage was performed in a batch type of program. Batch processing insures consistency in the data reduction, reduces human errors, and speeds the data reduction.

Stage 3. -- Preparations for publications

Usually months after the observations are completed, the astronomer starts the third stage of reduction which typically entails the production of high quality graphs and tables which summarize the data and which are suitable for publication. The graphs and tables produced by a computer will occasionally need the helping hand of a typist or a draftsmen. The tastes of the experimentalist dictate, so the types of graphs and tables produced in this stage of reduction will differ significantly from those produced for a similar experiment by another astronomer. Any data reduction system must then have graphics and table capabilities which are under the control of the astronomer. For example, something as simple as the length of the tick mark on a graph, the arrangement of columns in a table,

or as complicated as the viewing perspective for a 3-dimensional plot must be under the control of the astronomer.

Therefore, during each subsequent stage of data reduction, each experiment becomes more and more unique as does the data reduction software used by the observer. In principle, each stage in reduction can be performed by different computer programs but, to make matters simple for the experimentalist, the syntax of how to enter commands should be identical. Few astronomers would enjoy having to read more than one set of documentation or learn more than one system.

III. Library Functions

A complete list of functions which should be part of a data reduction library would be as extensive as the number of astronomers who use the library, as implied by what I have described in the previous sections. However, a list of the basic functions can be easily created; my list, which may differ with lists made by others, is given below. It is based on careful reflection on what I have been actually doing to the various types of data I have encountered. Note that many functions astronomers usually consider imperative within a reduction system are, in actuality, combinations of functions in my list. For example, overlapping frequency switched spectral line data involves the use of functions 4, 5, 9, and 10. Averaging scans together entails functions 4 and 5. Thus, many complex library functions are built out of the simple functions listed here. The list is in no particular order.

1. Obtaining data from a user-specified file located on computer disk or tape and storing the data as an array which can then be accessed and altered by the user.

2. Fitting functions to certain specified points within an array containing data. These functions should be:

   Polynomials
   Sine Waves
   Multiple Gaussians
   Any linear or non-linear function

3. Constructing a model of a function and storing it as an array. The type of functions are the same as given in 2 above.

4. Multiplying, Dividing, Adding, Subtracting data points in one array with another array

5. Multiplying, Dividing, ... a constant with the data points in an array.

6. Smooth the data points in a scan using:
   Any user supplied smoothing function.
   Certain types of standard smoothing functions

7. Auto-correlate, cross-correlate the data in an array with that in another array.

8. Produce a fourier or inverse fourier transform of the data in arrays.

9. Shift the x-coordinate (time, position, frequency, velocity, ...) of an array by an arbitrary amount or an amount which will align the x-axis in one array with that in another array.

10. Re-griding of the data in a scan in the x-coordinate. Expand or contract the x-axis of an array.

11. Calculate the parameters of a subset of data points in an array and store the results for later use. The parameters which should be calculated are, for example:
   rms and higher moments of the data
   average intensity across the data
   maximum/minimum intensity in the data and the x-coordinate at those points.
   integrated intensity
   intensity weighted average of the x-coordinate.

13. Summarize in a table either:
   Data from an n-dimensioned array
   Any parameters produced by any stage of data reduction
   Header information
   Contents of the data file

   (For example, one should be able to produce a table with columns containing scan numbers, RA, DEC, system temperature, rms, the intensity of the 25th data point, integrated intensity, etc.)

14. To manipulate columns and rows within tables similar to what can be done in many of today's spread sheet or data base management programs. This should include:
   Fitting arbitrary functions to columns/rows
   Graphing any two columns/rows against each other
   Producing 1, 2, 3, ... dimensional arrays from the entries in a table.

(Using the example in 13 above, one should be able to produce a 2 dimensional array of integrated intensity versus RA, DEC which can then be contoured plotted.)

15. The ability to store and retrieve arrays, tables, or graphs.

16. Accessing any number from the scan header or from the data itself.
Storing the number for later use,
Replacing the original number with any number.

17. Summarize in a 1 dimensional graph the data in an array
The graphics must allow for:
    Altering the range of x and y values which will be plotted.
    Altering the number and label type for tick marks
    Altering the labeling of the graph
    Allowing multiple graphs to be displayed simultaneously.
    Interactive cursor for determining the x and y coordinates of any point in the graph.

18. Storing one-dimensional data in a two or higher dimensional arrays. For example:
    Storing peak or integrated intensities as a function of the coordinates in the sky at which the data was taken (2-dim array).
    Storing the intensities at each data point in a scan as a function of the scan's x-coordinate (e.g., velocity) and as a function of position in the sky (so called position-velocity maps; 2-dim. array).
    Storing intensities at each data point in a scan as a function of the scan's x-coordinate (e.g., velocity), and the two sky coordinates (so called data cubes; 3-dim array).

19. Turning high dimensioned arrays into lower dimensioned arrays by, for example, averaging or adding data along one or more dimensions. For example:
    Adding together all the data within a 3-dimensional data cube in the two sky coordinates so as to produce a 1-dimensional average spectra for a source.
    Specifying two points within a 2-dimensional continuum map of a source so as to produce a 1-dimensional profile through the source.

20. Displaying, using contour, gray-scale, or color raster plots, the contents of subsets of these high dimension arrays.

21. Performing all the same functions (*, /, +, -, FFT, etc.) on

these higher dimensioned arrays as can be performed on the 1-dimensional arrays.

IV. Implementation

For some simple minded applications or for first time users of an analysis system, it would be helpful if the user could be prompted for input through menus or through question and answer sessions. The speed at which one initially learns simple operations should be fast; sophisticated operations should not be too difficult to learn by a competent astronomer.

Sophisticated users would find things like menus very cumbersome and time consuming so the quick way of typing commands on a terminal would also be needed.

Help files should be accessible from within the program and both a cookbook type of manual and a complete, larger manual should be available and kept up to date.

Hard copies of graphs and tables should b of publishable quality so that later work by a draftsman or typist will be minimized.

The analysis system should have complete and up-to-date system documentation so that the program can be easily maintained by a small staff of programmers. The languages in which the program was written should be few and one of the standard languages. The program should run on commonly found machines and have few machine dependent feature so that the program will be as portable as possible. It would be great if the program could also be run on a PC.

The input and output of the program should be easily converted into FITS image files so that AIPS or IRAF can further process the data or provide input back into the analysis system.

Many existing analysis software are interactive programs but also give the user the capability of defining procedures for batch-like processing. Procedures can be built out of standard library functions or out of other user defined procedures. Thus, if a user wishes to create a new way of reducing data, he or she can write a procedure to do exactly what is desired. The syntax for procedures are usually unique in that they seldom resemble any of the standard computer languages but the rules of the language are usually easy to learn. Such items as assignment statements; loops; conditional and branch statements; defining variables to be logical, integer, real, character, or arrays; and math functions (sin, cos, atan, sqrt, int, ...) are part of many

analysis languages though some languages do not provide all. Header information and data should be available as variables so that a procedure can examine and modify those items as needed.

Full editing capabilities for user-defined procedures must also be part of the analysis system. If a procedure becomes useful to many observers, then it should be easy to install it into the standard library so that everyone can use it. A procedure, if written correctly, can also do the batch type of processing suggested in section II.

Some very complicated analysis steps would be more easily performed not through procedures but through user-supplied functions or subroutines written in such standard languages as Fortran, C, or Pascal. The steps needed to get data into these subroutine and the steps needed to compile and link the subroutines should be spelled out carefully and made as simple as possible since most of the users of an analysis system are one-time user.

It is my opinion that any analysis system which does not EASILY allow for the writing of procedures will be useful in only a few rare, simple cases and will have a short existence. A system which is only a batch-type of program and is not interactive or is only menu driven will also be very much less than the ideal.