Introduction to the

Caltech VLBI Programs

May 1991

California Institute of Technology
Pasadena, California 91125
(818) 356–4980

Please address comments on the programs or this manual, including requests for copies, bug reports, and requests for improvements, to the editor:

Tim Pearson,
Astronomy Department,
Caltech 105–24,
Pasadena, California 91125,
USA.

Telephone: +1 818 356-4980
BITNET: tjp@citdeimo
SPAN: Deimos::TJP or 6021::TJP
INTERNET: tjp@deimos.caltech.edu

Contents

1 INTRODUCTION 1-1
Capabilities 1-1
The PGPLOT subroutine library 1-3
Status 1-3
Availability 1-3
Acknowledgments 1-3

2 USING THE PROGRAMS 2-1
Introduction 2-1
Commands 2-1
The VLBHELP command 2-2
Parameters 2-2
Graphics 2-3
Command procedures 2-4
Saving files on magnetic tape 2-7

3 GLOBAL FRINGE FITTING 3-1
The correlator 3-1
Get data onto a computer for fringe-fitting 3-2
Calculate u, v, w for your data 3-2
Combine data from the same scan 3-2
Change data format 3-3
Index data 3-3
Protect CL table 3-3
Do amplitude calibration with task ANCAL 3-3
Flag known bad data 3-3
Edit CL table and insert delay offsets 3-4
Use CALIB for fringe-fitting 3-4
Examine results of CALIB with LISTR 3-6
Apply the solution table 3-6
Produce calibrated data 3-7
Run CALIB to correct the phases 3-7
What to do with Good Data 3-8
When should you fringe-fit your data twice? 3-8
An AIPS run-file for fringe-fitting 3-10

4 HYBRID MAPPING 4-1
Preliminaries 4-1
Mapping 4-3
What is a good map? 4-6

Advice on finishing up . . . 4-7
Sample command file . . . 4-7

5 MODELFITTING . . . 5-1
The Need for Model Fitting . . . 5-1
Practical Model Fitting . . . 5-2

6 DEFINITIONS AND CONVENTIONS . . . 6-1
Station Coordinates . . . 6-1
Baseline Vector . . . 6-1
Interferometer Hour Angle (IHA) . . . 6-2
Baseline Coordinates (u, v, w) . . . 6-2
Sky Coordinates (x, y) . . . 6-2
Phase . . . 6-2
Closure Phase . . . 6-3
Orbital Parameters . . . 6-3

7 MODEL FILES . . . 7-1
Model file format . . . 7-1
Model computation in the (u,v)-plane . . . 7-2
Model computation in the sky plane . . . 7-4
Convolution of elliptical gaussians . . . 7-5

8 FITS FILES . . . 8-1
Introduction . . . 8-1
FITS images (maps) . . . 8-1
UVFITS data files . . . 8-2
Use of FITS files with AIPS . . . 8-2
Copying FITS files from tape to disk (VMS) . . . 8-2
Copying FITS files from tape to disk (Convex) . . . 8-3
Copying FITS files from disk to tape (VMS) . . . 8-3
Copying FITS files from disk to tape (Unix) . . . 8-4
Checking the contents of a FITS tape . . . 8-4
References . . . 8-5

9 KEYIN . . . 9-1
Parameters . . . 9-1
Values . . . 9-1
Comments . . . 9-3
End marker . . . 9-3
Special commands . . . 9-3
File inclusion and parameter saving . . . 9-4

10 MOVING FILES BETWEEN VMS AND UNIX . . . 10-1
Note on using netcp . . . 10-1

Transferring TEXT files from VMS to UNIX 10-1
Transferring TEXT files from UNIX to VMS 10-1
Transferring FITS files from VMS to UNIX 10-2
Transferring FITS files from UNIX to VMS 10-2
Transferring MERGE files from VMS to Convex 10-2
Transferring MERGE files from Convex to VMS 10-3

11 FORMAT OF MERGE FILES 11-1
Header Records 11-1
Data Records 11-3

Chapter 1

INTRODUCTION

The Caltech VLBI Analysis Programs are a set of programs for the planning and analysis of Very Long Baseline Interferometry (VLBI) observations. The programs are available for VAX/VMS, Sun UNIX and Convex UNIX.

1.1 Capabilities

The Caltech VLBI Analysis Programs are used for planning an experiment, assisting in correlation, fringe-fitting, data display, calibration and editing, model-fitting and mapping. The following paragraphs summarize the uses of the major programs, excluding some of the less commonly used ones. All the programs are controlled with keyword-directed commands which can be used interactively or in batch command files. Sequences of programs can be linked together using the command language of the host operating system (UNIX or VMS).

Planning an experiment

UPTIME computes rise and set times of sources at one or more telescopes, and displays the results graphically. It can also produce graphs of source elevation and parallactic angle versus time. **HAZI** computes and displays the u, v-tracks of a hypothetical experiment. **SCHED** generates printed and machine-readable schedules for VLBI antennas. **FAKE** generates a Merge-format file containing artificial data (computed from a model) for a specified source and antennas; noise can be added. It is useful for testing programs, hybrid-mapping algorithms, etc. FAKE includes capabilities for simulating antennas in earth orbit and long baseline optical interferometers. **PRECESS** converts radio source coordinates from B1950 or J2000 mean position to B1950 or J2000 mean position, geocentric apparent position, ecliptic longitude and latitude, or galactic longitude and latitude.

Correlating and fringe-fitting

MAKECCR creates a script to run the JPL/Caltech Block-II VLBI correlator. **B2PS** summarizes the contents of output files from the correlator. **B2FITS** converts the output of the Block-II correlator into standard UVFITS format acceptable to AIPS. The first stage in the post-correlation analysis is to calculate the fringe amplitude and phase as a function of time on each baseline; this is usually done with the AIPS system of the National Radio Astronomy Observatory.

Visibility data

Most of the programs operate on interferometer visibility data, which are stored in data files in a standard format (the 'Merge' format). The following programs are used for converting data from other formats into the Merge format: **FITSMERGE** (converts UVFITS data); **EVNMERGE** (converts the European VLBI network format); **MK3MERGE** (converts export data from Mark-III correlators). There are also programs to convert Merge-format data into other formats: **MERGEFITS** (converts to AIPS/UVFITS format), **VLBEXPORT** (converts to ASCII text format).

Display of visibility data

VISPLOT plots the visibility amplitudes and phases, closure phases, and closure amplitudes (if requested) versus UT, GST or interferometer hour angle, with a superimposed model curve if desired. It can plot both calibrated and uncalibrated data. **LIST** lists the contents of a Merge-format file at a terminal or on the line-printer. It can list the file header, (u, v, w) coordinates, amplitudes, visibility phases, closure phases and the errors in these quantities. **UVPLOT** displays the u, v-plane coverage of the dataset. **UVDIST** plots amplitude versus radial distance or position angle in the u, v-plane. **CLAMP** calculates and lists the closure amplitudes. **VPLOT** is an interactive program for data examination. It combines some but not all of the functions of VISPLOT, UVPLOT, and UVDIST.

Operations on visibility data

VLBEDIT is used for removing bad data points from a data file or for making corrections, e.g., for telescope mispointing. It can also be used to change parameters in the file header. **IED** is a convenient tool for interactive data editing, if an interactive graphics device is available. **CAL** uses measured system temperatures, gain curves, antenna temperatures etc to convert the data from correlation coefficients to correlated flux densities (or visibilities). **AVERAGE** averages the data either coherently (complex average of visibilities) or incoherently (rms of amplitudes). **UVCROSS** can be used to identify crossing-points of tracks in the u, v-plane, and for checking the calibration at these points. **UVCOPY** extracts a subset of the data from a Merge file. Data can be selected by time and by radius in the u, v-plane. **MERGE** combines data from two or more Merge files into a single file.

Model fitting

In many cases parameters of the observed source can be estimated by fitting a model of the source brightness distribution to the measured complex visibilities. The Caltech programs allow the use of models of up to 10 000 components, where a 'component' is, e.g., a point source, an elliptical gaussian, or an elliptical disk. **MODELFIT** does a least-squares fit of a model of one or more components to the visibilities and closure-phases. The output is a model that can be compared with the data via program VISPLOT or displayed as a contour map with MODPLOT. **QFIT** computes the chi-squared agreement factors between a model and a dataset. **ERRFIT** is a program for estimating the confidence intervals of the fitted model parameters. **MODPLOT** displays the models produced by MODELFIT or CLEAN as contour maps. **MODSUM** can add and subtract models. **MODFIX** provides a number of operations on models, e.g., scaling, rotating, windowing.

Hybrid mapping

Hybrid Mapping is an iterative self-calibration procedure, in which one attempts to estimate the unknown phase and amplitude calibration errors introduced at each antenna. These errors do not affect the closure phases and closure amplitudes. Three programs are used: AMPHI, INVERT, and CLEAN. **AMPHI** estimates the calibration corrections to minimize the disagreement between the corrected visibilities and a starting model provided by the user. **INVERT** makes a 'dirty' map using the normal aperture-synthesis equations; **CLEAN** cleans the dirty map producing a 'deconvolved' map—an array of point sources—that can be taken as a new starting model. The agreement of this model with the original data can be assessed qualitatively with VISPLOT or quantitatively with QFIT. **VLBMEM** is an alternative imaging program based on the Maximum Entropy algorithm of Gull and Skilling. **MAPPLOT** displays images as contour maps or gray-scale images. **MAPSTAT**

computes simple statistics (mean, rms, total flux) for regions of an image. The images produced by the programs are stored in FITS format and can therefore be analysed with many different image processing programs.

1.2 The PGPLOT subroutine library

Many of the programs produce graphical displays using the PGPLOT subroutine library. PGPLOT is a Fortran-callable, device-independent graphics package for making simple scientific graphs. It is intended for making graphical images of publication quality with minimum effort on the part of the user. For most applications, the program can be device-independent, and the output can be directed to the appropriate device at run time. Although originally conceived as part of the Caltech VLBI Analysis Programs, PGPLOT has much wider applicability and is now widely used at astronomical and other scientific institutes throughout the world. For further information, contact Dr T. J. Pearson at the address given below.

1.3 Status

The programs are written in Fortran-77 with occasional use of some of the common VMS extensions to Fortran-77. As far as possible, machine- or system-dependent functions are encapsulated in a small number of subroutines. At Caltech, the programs run under the VAX/VMS, Sun UNIX, and Convex UNIX operating systems. They can probably be ported to other varieties of UNIX with a moderate amount of work. Development of the programs is continuing, and updated versions of each program are available on request. It is planned to release new versions of the entire package approximately twice a year.

1.4 Availability

The Caltech VLBI Analysis Programs, including complete source code and documentation, are available free of charge to non-profit astronomical research institutes. For information, contact Dr T. J. Pearson, Astronomy Department 105-24, California Institute of Technology, Pasadena, California 91125, USA.

Internet: TJP@Deimos.Caltech.Edu;

Bitnet: TJP@CITDEIMO; SPAN: 6021::TJP.

1.5 Acknowledgments

The development of the programs has been supported by the National Science Foundation. Many people have contributed to the programs, including M. H. Cohen, G. Comoretto, D. Fort, S. F. Gull, M. W. Hodges, R. Linfield, D. L. Meier, T. J. Pearson, G. H. Purcell, A. C. S. Readhead, D. H. Rogstad, J. Romney, R. S. Simon, A. Tzioumis, S. C. Unwin, R. C. Vermeulen, R. C. Walker, P. Wallace, P. N. Wilkinson, and J. Wrobel.

Chapter 2

USING THE PROGRAMS

2.1 Introduction

Most of the Caltech VLBI programs run under both the VAX/VMS operating system and the UNIX operating system; this chapter gives general instructions for both operating systems. It is assumed that the reader has some acquaintance with the command language of each operating system (DCL for VMS, csh for Unix).

Unlike AIPS, the programs do not try to hide the operating system from you. In order to use them, you need to be familiar with the basic operations of the operating system (VMS or UNIX), such as creating, editing, and deleting files. You also need to understand file naming conventions; two important differences between VMS and UNIX are that in VMS file names are insensitive to case, while in UNIX upper- and lower- case names refer to different files; and in VMS it is possible to have multiple versions of a file with the same name (distinguished by version number) whereas UNIX allows only one file with a given name. Data analysis proceeds by stages—at each stage a new file is created, so that it is always possible to go back to an earlier stage if a problem occurs; but if you do so in UNIX, you must change the file names or delete the unwanted files before backtracking.

2.2 Commands

Each VLBI program is run by typing a command, which is usually the name of the program. e.g., to run the VISPLOT program, type `visplot`. Note that in UNIX, the command must be typed in *lower case*. In VMS, the case is not significant. Before the commands can be used, they must be defined.

In VMS, command to run the programs are defined by executing the following DCL command:

```
$ @VLB:LOGIN.COM
```

This will be done automatically every time you log in if you include this command as one line of your personal `LOGIN.COM` file. The directory name `VLB:` is appropriate for the Caltech Astronomy computers, but you may need to use a different name on other computers.

In UNIX, you make the programs available by including the `vlb` directory in your 'path'. As an example, the following might be included in your `.login` file:

```
set path = (. ~/bin /vlb/vlb /usr/convex /usr/ucb /bin\
  /usr/bin /usr/local/bin /aips/aips)
```

The directory `/vlb/vlb` contains the VLBI programs at Caltech (on Ikaros; the directory may be different on other machines). The order of directories in the path is important; in this example, the command `list` will run the VLBI program LIST, even though there is another program by the same name in the `/usr/convex` directory.

2.3 The VLBHELP command

Interactive help about the programs is available by use of the command `vlbhelp`. In VMS, this command uses the normal VMS help utility. In Unix, it runs a special program to display the 'help files' for the VLBI programs. Example:

```
vlbhelp visplot
```

gives information about the VISPLOT program. All the VLBI programs are documented in this way.

To get a printed copy of the help file for a particular program, use the operating system's 'print' command. In VMS, the help files are kept in the directory `VLBHELP:` with file name *program*`.mem`. In Unix, the files are usually in directory `/vlb/vlb/help` (the `vlbhelp` command displays the precise file name).

In VMS, a print command would be:

```
$ print vlbhelp:visplot.mem
```

In UNIX, the corresponding command is:

```
lpr /vlb/vlb/help/visplot.mem
```

A manual containing printed copies of all the help files is available from Tim Pearson.

2.4 Parameters

Most of the programs are controlled by *parameters*: you supply values for the parameters by name, e.g.,

```
list
  INPUT = "3C84A.MRG"
  LISTFILE = "3C84A.LIS"
  AMP
  /
```

When run interactively, the programs prompt for input with an asterisk `*`. In command files, the parameters are taken from the standard input stream (see below). Most parameters have sensible default values and usually need not be specified. Parameters are either single keywords requesting some action (e.g., AMP), or have the form *keyword* = *value* to change the value of some program parameter (e.g., INPUT for the name of the input file). In most programs, the parameter names can be abbreviated to the minimum unique string. Values are either numbers or text strings, and text strings should usually be enclosed in quotation marks. Some parameters can have several values, separated by commas. For more details, see the chapter on *KEYIN*. End the list of parameters with a slash (/) or an end-of-file character (control-Z in VMS, control-D in Unix).

Commonly used parameters include:

- INPUT: the name of the input file for this program.
- OUTPUT: if this program creates a new data file, this parameter is used to give it a name.

- LISTFILE: the name for a file to receive printable listing or log of the program's execution.
- PLOTFILE: If the program produces graphical output, this parameter is used to provide a *device specification* for the graphical output device or file (see next section).
- SOURCE: the name of the source (object).
- STATIONS: the names of one or more stations (antennas).

Some interactive programs ask questions: e.g., the interactive editor IED will ask you to supply an input file name and an optional output file name. Type the answer to the question on the same line, and end with a carriage-return. Usually if your reponse is invalid (e.g., the file you specified for input does not exist), the program will ask the question again.

2.5 Graphics

Several of the VLB programs produce graphical output using the PGPLOT subroutine library, which is described in the manual *PGPLOT Graphics Subroutine Library* by T. J. Pearson. When using these programs, you have to specify the graphics device to be used. Graphics devices fall into two classes: devices which produce a hardcopy output, and interactive devices, which usually display the plot on a TV monitor. Hardcopy devices are not used interactively. One must first create a disk file and then send it to the appropriate device with a print or copy command.

A PGPLOT graphical output device is described by a "device specification" that consists of two parts, separated by a slash (/): the *device name* or *file name*, and the *device type*.

Device name The device name or file name is the name by which the output device is known to the operating system. For most hardcopy devices, this should be the name of a disk file, while for interactive devices, it should be the name of a device of the appropriate type; in both cases, the name should be specified according to the syntax of the operating system in use. If the device or file name includes a slash (`/`), enclose the name in double quotation marks (`"`). If the device name is omitted from the device specification, a default device is used, the default depending on the device type. In Unix, device and file names are case-sensitive.

Device type The device type tells PGPLOT what sort of graphical device it is. The device type is not case-sensitive: you can use uppercase or lowercase letters, or a mixture of the two.

Examples (VMS)

Tektronix 4006/4010 terminal: `TTA4/TEK` (device `_TTA4:`).

X-window image display: `/XW`.

Disk file, PostScript format: `SYS$SCRATCH:PLOT.PS/PS`. (Use `/VPS` for portait mode, *i.e.*, long edge of paper vertical.)

Examples (Unix)

Tektronix 4006/4010 terminal: `/TEK` (the logged-in terminal).
IVAS image display: `/IVAS`.
Disk file, PostScript format: `"/scr/tjp/plot.ps"/PS`.
Sun workstation screen (SunView mode): `/SUN`.
Sun workstation screen (X-window mode): `/XW`.

Getting hardcopy

The commands used for printing a plot file vary from system to system: check with your system manager. To print a PostScript file on Deimos, use

```
$ PRINT/QUEUE=LW PLOT.PS
```

On Ikaros, use

```
lpr -Pps plot.ps
```

To print a Printronix plot file on Deimos, use

```
$ PRINT/PASSALL PLOT.LIS
```

On Ikaros, use

```
lpr -V plot.lis
```

Make sure you send the plot file to the right sort of printer.

2.6 Command procedures

Instead of typing in commands at the terminal, it is often convenient to create *command procedures* to run one or more programs. The command procedure can then be run interactively (as if you had typed in the commands directly), in a sub-process (which leaves your terminal free for other work), or in a batch job (which can be scheduled to run at any time of day). A major advantage is that if something goes wrong, the command procedure can be edited to correct the problem and then re-run. To gain a complete understanding of command procedures, it is necessary to study the operating system manuals, but basically one just creates a text file (using an editor) that contains the commands one wants to execute in sequence.

VMS example

The following VMS example runs two programs, MODMAP to make an image file from a model file, and MAPPLOT to make a contour map of the resulting image; let us suppose the command file is called PLOT.COM.

```
$ SET DEFAULT CITSCR:[TJP]
$ MODMAP
BEAM       = 2.5
FREQ       = 5011
MODFILE    = CIT:[PDB.JUL80.0851]0851.MOD
MAPFILE    = 0850_JUL80.MAP
LRTB       = -17,13,13,-17
XYINT      = 0.3
SOURCE     = "0850+581"
/
$ MAPPLOT
PLEV       = 1
LEVS       = -1,-0.5,0.5,1,2,4,8,16,32,64
MAPFILE    = 0850_JUL80.MAP
PLOTFILE   = "0850plot/VPS"
LINEWIDTH  = 4   LTYPE = 4
BEAM
CHarsize   = 1.2
NOTITLE  Topleft = "1980.53"
/
$ EXIT
```

Note in particular:

1. DCL commands have a $-sign at the beginning of the line; otherwise, the commands are precisely the same as interactive commands.
2. If the program normally asks questions when it is run interactively, then the answers to the questions must be included in the command procedure in the right places.
3. The SET DEFAULT command at the beginning makes sure the procedure is executed using the correct default directory; batch jobs start out in the normal login directory.
4. The EXIT command (optional) terminates the procedure.

   To run this procedure PLOT.COM interactively, type

   ```
   $ @PLOT
   ```

To run it as a subprocess, saving the output in a log file:

```
$ SPAWN/NOWAIT/NOTIFY/INPUT=PLOT.COM/OUTPUT=PLOT.LOG
```

To run it as a batch job, starting no earlier than 8 p.m. this evening:

```
$ SUBMIT/NOPRINT/NOTIFY/LOG=SYS$SCRATCH:PLOT.LOG/AFTER=20:00:00 -
  PLOT.COM
```

In the last two cases, the log file (PLOT.LOG) can be examined while the job is executing by using the TYPE command (TYPE PLOT.LOG). For more information, see the VMS manuals or use the HELP command (HELP SUBMIT, HELP SPAWN).

UNIX example

The following `csh` example runs two programs, MODMAP to make an image file from a model file, and MAPPLOT to make a contour map of the resulting image; let us suppose the command file is called `plot`. (Warning: don't use the name of one of the Caltech programs as the name of a command file.)

```
cd /scr/tjp
modmap <<EOD
  BEAM       = 2.5
  FREQ       = 5011
  MODFILE    = /mnt/pdb/jul80/0851/0851.mod
  MAPFILE    = 0850_jul80.map
  LRTB       = -17,13,13,-17
  XYINT      = 0.3
  SOURCE     = "0850+581"
  /
EOD
mapplot <<EOD
  PLEV       = 1
  LEVS       = -1,-0.5,0.5,1,2,4,8,16,32,64
  MAPFILE    = 0850_jul80.map
  PLOTFILE   = '"/scr/0850plot"/vps'
  LINEWIDTH  = 4   LTYPE = 4
  BEAM
  CHarsize   = 1.2
  NOTITLE  Topleft = "1980.53"
  /
EOD
exit
```

Note in particular:

1. The programs read parameters from the standard input stream. In `csh`, the `<<` operator reads data from the command file up to the terminator (`EOD` in this example) and passes the data to the program through its standard input.
2. If the program normally asks questions when it is run interactively, then the answers to the questions must be included in the command procedure in the right places.
3. The `cd` command at the beginning makes sure the procedure is executed using the correct default directory; batch jobs start out in the normal login directory.
4. The `exit` command (optional) terminates the procedure.

   To run this procedure interactively, type

   ```
   csh plot
   ```

To run it as a background subprocess, saving the output in a log file:

```
csh plot >plot.log &
```

2.7 Saving files on magnetic tape

Data should be archived on magnetic tape using `BACKUP` (VMS) or `tar` (Unix). Binary data files can be transported to another computer running the same operating system, but cannot be transported in this way from, say, VMS to Unix, or from one variety of Unix to another.

For transporting data, both Merge (visibility data) files and images can be archived on standard FITS-format tapes which can be read by the NRAO AIPS image processing system and many other programs. See the chapter on "FITS files" for more information.

VMS BACKUP

The following is an example of the commands required to copy all the files belonging to user EXT1 from two disks, CIT and CITSCR:

```
$ ALLOCATE MTA1:
```

(mount your tape on MTA1:)

```
$ MOUNT/FOREIGN MTA1:
$ BACKUP/REWIND/DENSITY=1600/VERIFY/LIST=SYS$PRINT:/LOG -
  CIT:[EXT1...]  MTA1:CITEXT1.BCK
$ BACKUP/NOREWIND/VERIFY/LIST=SYS$PRINT:/LOG -
  CITSCR:[EXT1...]  MTA1:SCREXT1.BCK
$ DISMOUNT MTA1:
$ DEALLOCATE MTA1:
```

The /REWIND qualifier on the first BACKUP command causes BACKUP to overwrite any existing files on the tape. Specify /NOREWIND if you don't want to do this. See the VMS manuals for more information about BACKUP.

UNIX tar

Different versions of UNIX implement `tar` differently, so read the UNIX documentation carefully before using it. The following is an example of the commands required to copy all the files in a subdirectory `jun86` of the current directory under Convex OS v8.1.

```
tpmount -d 6250
```

(This allocates a tape drive; mount your tape on the drive allocated.)

```
tar cv jun86
tpunmount tape
```

Take care when using `tar`; with the `c` option it starts writing at the beginning of the tape, overwriting any existing files. Do not specify the "root" directory in the path to the files you want to backup (e.g., `/scr/tjp/jun86`); if you do, you can only restore the files to the same location.

Chapter 3

GLOBAL FRINGE FITTING

This is an updated version of a document prepared by Ann E. Wehrle and Steven T. Myers. It is intended as a users guide for getting data from the Caltech/JPL correlator through the fringe-fitting routines in AIPS to the AIPS or Caltech VLBI mapping routines. It should be read in conjunction with Chapter 11 of the *AIPS Cookbook* (15-Oct-90) and the appropriate AIPS Explain Files. The instructions apply to the VLBI tasks in the 15OCT88 and later versions of AIPS; previous versions of some tasks have fatal errors.

You may wish to read *all* the Explain files for the AIPS tasks UVLOD, DBCON, UVFIX, UVAVG, ANCAL, MULTI, INDXR, CALIB, CLCAL, LISTR, SPLIT, EXTDEST, and TACOP before you start fringe-fitting. The most important of these are CALIB, CLCAL, and SPLIT. Also, print and read the EXPLAIN CALIBRAT file, which is different from EXPLAIN CALIB. If you are a novice VLBI-er, you will need considerable help from a pro.

This chapter includes a RUN file for AIPS on IKAROS which has produced good results on strong sources. Some modification will be needed to fringe-fit weak sources bracketed by calibrator observations. We strongly suggest you use a RUN file because it is easy to make changes in it and you can run it repeatedly.

The input parameters in this chapter have been used successfully to fringe-fit data at 6 cm. Solution intervals and similar parameters will have to be adjusted for other wavelengths or as suits the user. We must stress that most of these parameters will be need to be tuned to the specific dataset being fit and user experimentation is suggested. This is *not* a cookbook.

Note on preferred procedure: When confronted with a virgin dataset, the best procedure is to run a quick UVLOD–CALIB to make sure that everything is OK with the data. No UVFIX, UVFLG, calibration is needed and it can be run directly on the single-source UVFITS file (watch out as it will produce a SN table). Don't do anything with the CALIB output file, just look at the solutions for non-detections and funny delays. Pay careful attention to missing antennas, choice of REFANT, and times where the SNR was too low—check againt the experiment logsheets. This will find most problems in correlation and allow setting up of the UVFLG RUN file, and selection of REFANT, ANTWT, and windows in CALIB. If you find a station with a large delay or rate and do not wish to recorrelate, then use these preliminary delays in step 3.10 below. This test-fitting should be performed as soon at the B2FITS data is in hand so as to determine if any recorrelation is needed. Don't wait several years and then decide that the clock was wrong on station X and could we please try again.

3.1 The correlator

The Caltech–JPL Block II Correlator can process data from up to 16 stations in one pass. If your data were correlated in several passes, you may have redundant correlations. If so, you may combine the data using the program B2FITS. B2FITS uses correlator output ("PCR files") to make FITS format files ("uvf files"). B2FITS will choose the best data

for duplicate passes for you. When running B2FITS, check that multiple correlations of the same scan used the same clock offsets and the same station positions; if not, you must recorrelate because you will not be able to do global fringe-fitting using all your data. You should use epoch 1950. [Absentee users of the correlator are supplied with uvf files from B2FITS, not the PCR files.]

If data from different passes were run separately through B2FITS, you will need to use task UVAVG in AIPS to remove redundant correlations.

The Caltech correlator is an 8-lag correlator in normal operation. The correlated output is hardware FFT'd and 4 frequency bins are kept. The relatively few delays correlated means that errors are introduced into the fringe-fitted amplitudes and phases (non-closing errors!): about 2% amp, no phase at -125 ns residual delay to 5% amp and 0.8 degrees phase at $+550, -800$ ns; then it rapidly gets worse. The correction factors can be calculated for this effect but note that with an even number of lags the error is not symmetric about zero delay. This means that since the station order at correlation is not consistent (and more importantly, not passed into AIPS), a correction cannot be made for 8-lag data. Currently the correlator uses only 7 lags to avoid this problem. It is also possible to run the correlator with 15 lags, but only for experiments with 10 or fewer stations. This will keep errors down to much less than 1% and 1 degree out to 1000 ns.

Tests performed by us (STM, SCU) at Caltech also show bandpass mismatch effects at the 1–3% level over the $+500, -750$ ns range in residual delay. Theoretically it is possible to determine the errors from cross-power spectra (giving us the phase error vs. frequency), but more testing on this is needed.

3.2 Get data onto a computer for fringe-fitting

The FITS files are then transferred via tape or a net to a computer running AIPS. The FITS files are read into the computer using AIPS task UVLOD.

3.3 Calculate u, v, w for your data

Task UVFIX will calculate the uvw coordinates which were not provided by B2FITS. Be sure to set UVFIXPRM(1)=1 or you will get incorrect baselines.

3.4 Combine data from the same scan

If you have several passes of data from the same scan, correlated with the same reference date, the same epoch (1950), and the same antenna list (order of antennas is important), combine them using task DBCON with DOARRAY=TRUE. Then remove the duplicate correlations using task UVAVG. If your several passes of data from the same scan do *not* fit the preceding criteria, it *may* be possible to modify DBCON to deal with this more general situation, in which case you should contact the AIPS group.

3.5 Change data format

If you used B2FITS, your FITS files from the correlator (and from UVFIX) are in "single-source format". You should convert the data to "multi-source format" to use the fringe-fitting program CALIB (CALIB does other things besides fringe-fit VLBI data, as you will discover) to take advantage of the table manipulation ability. Run task MULTI to convert to multi-source format. In MULTI, use APARM(1) = 1 minute, or choose some time which is no larger than the desired delay-rate solution interval.

3.6 Index data

Now you need to set up an indexing system so that the AIPS tasks can access your data. Run task INDXR with default settings to create an index table.

3.7 Protect CL table

Copy your original CL table version 1 to version 2 to protect version 1 from modification by subsequent tasks. Use TACOP.

3.8 Do amplitude calibration with task ANCAL

At this point, you can do amplitude calibration using task ANCAL which allows the system temperature and gain curve information from your station logs to be applied to the data: you'll get correlated flux densities from "correlation units". Read EXPLAIN ANCAL for instructions on creating the calibration file in the $RUN area. Modify CL table 2, not (the original) table 1.

An alternative is to defer amplitude calibration until after fringe-fitting, and use the Caltech program CAL. This has the advantage that it requires much less work to redo the calibration (which is often an iterative procedure).

3.9 Flag known bad data

Use task UVFLG to produce a FG table for time intervals when your station logs noted problems and when each antenna was "off-source", *e.g.*, from the scan start time to the time the antenna began tracking your source. Although this is tedious, it will ensure better solutions, faster-running tasks, and minimal time spent editing later. Running this task from a RUN file is strongly recommended. Warning: AIPS uses International Atomic Time (IAT or TAI), not UTC; so times from the antenna logs have to be modified.

3.10 Edit CL table and insert delay offsets

This optional step is particularly useful if one or more of the antennas has a significant (> 200 ns) clock offset as this will allow the use of smaller search windows in the fringe fitting step. Use TABED with INEXT = 'CL'; INVER = 2; OUTVER = 2 (CL 2), OPTYP = 'REPL' (replace current values), APARM(1) = 21 (column 21 = delay), APARM(5) = 4 (column 4 = antenna), APARM(6) = n (for antenna n), APARM(7) = 0.5 (tolerance), KEYVAL = delay, 0 (put delay offset here, units are *seconds*). Run this once for each antenna to change. See EXPLAIN TABED for details.

3.11 Use CALIB for fringe-fitting

First, a word about tables. AIPS creates a series of tables which contain information about the calibration. These tables are later used to create a calibrated dataset which may be mapped in AIPS or transferred to the Caltech VLBI package for mapping. The goal in using CALIB and related programs is to produce and apply the appropriate tables to your data. The calibration parameters derived from some model of the source (*e.g.*, a point source) include antenna phase, antenna amplitude, antenna group delay (frequency derivative of antenna phase), antenna phase delay rate (time derivative of antenna phase). There can be many versions of the tables. CALIB makes a solution table, SN.

The tables are:

AN antennas, created by UVLOD from data in FITS header

SU source, created by MULTI

CL calibration table, gets modified as you go along

NX index table, tells AIPS how to access data fast, created by INDXR

CH channel, assigns frequencies to IF's.

SN solution, created by CALIB

BL baseline

BP bandpass

FG flag, identifies flagged data, i.e., marked as bad, created by UVFLG.

INPUT parameters for CALIB:

CALSOUR = ' ' do all sources and divide by fluxes in SU table

DOCALIB = 1 apply calibration produced by ANCAL

SMODEL = 0 don't divide by model (unnecessary for point source)

DODELAY = 1 delay, delay rate and phase solutions requested

GAINUSE = 2 CL table requested

FLAGVER = 0 use most recent version of the FG table

REFANT = *this is very important*; pick a reliable antenna present for most of the scan for preferred reference antenna; if CALIB finds no fringes to your REFANT when it is present it will choose the next highest-weighted antenna for a REFANT. If it can't find fringes to that antenna, it gives up.

SOLINT = 6 solution interval in minutes; choose greater than your coherence time (5–15 min usually OK)

APARM(1)=3 choose minimum number of antennas present for a solution

APARM(2)=0 input data has NOT been divided by a model previously

APARM(3)=0 don't average RR, LL (99% of VLBI data is LL only)

APARM(4)=0 don't average frequencies in IF's

APARM(5)=0 make separate solutions for IF's

APARM(6)=1 do some printout including antenna signal-to-noise ratio

APARM(7)=9 The minimum allowable signal to noise ratio; choose this number high (*e.g.*, 9 or 10) to avoid false detections if search window is large. *Data with SNR lower than this will be flagged and discarded.*

APARM(8)=0 From the HELP file: "If there is no antenna (AN) table with the input file, then the maximum antenna number in the file should be entered in APARM(8)."

Amplitude and phase control parameters (ignored for fringe fitting):

SOLTYPE=' ' solution type is normal least-squares method.

SOLMODE=' ' solution mode is phase-only for single source data.

CPARM = 0 does not matter at this point

Delay-rate control parameters (DPARMs):

DPARM(1)=1 Number of baseline combinations to use in initial, coarse fringe-search. Only used in FFT solution, which does provide the first guess to the global LSQ routine.

DPARM(2)=1000 Use 1000 nsec delay window (centered on 0) to search. Use a bigger window if you can't find fringes.

DPARM(3)=200 Use 200mHz rate window (centered on 0) to search. Use a bigger window if you can't find fringes.

DPARM(4)=2 The integration time of the data is 2 seconds for standard correlator output.

DPARM(5)= 0 Do least squares solution. The FFT is non-global and gridded into quarter-lag (62.5 ns) bins. The LSQ is necessary if you want a true global solution but may go out to lunch if the source is weak on many baselines. Beware.

DPARM(6)= 0 Average all frequencies in each IF (there are usually 4 frequencies per IF in Mark-II data).

ANTWT The antenna weights should be properly set if you ran ANCAL to calibrate your data. If you wish to force CALIB to weight a particular antenna higher than it deserves, set its weight high, *e.g.*, 5 or 10 while the rest of the ANWTs are 1. Never choose an antenna weight=0; AIPS will replace this by 1, which you probably did *not* want. If you want to downweight an antenna, use 0.001 for its weight, or some number very small compared to the other weights. Antenna weighting is a tricky business and there is some room for user experimentation here. I (STM) have had good luck using uncalibrated data (weights determined by number of bits correlated) in one case with a poor model.

Note for those interested : The weight given an antenna to determine order of preference for use as REFANT is the sum over all baselines involving the antenna in question of the product of the uv data weight, the model visibility amplitude, and the ANTWTs of the two antennas.

3.12 Examine results of CALIB with LISTR

After you have run CALIB, you will want to first look at the FFT results shown in the printout (use PRTMSG) on the printer (SAVE these pages until you are content with your final map). You should have good signal to noise ratios on all antennas. The signal to noise ratio printed out by CALIB is the SNR found on the FFT search. To look at the real solutions after running CALIB, run LISTR three times with:

DOCRT = −1 for printout; OPTYP = 'GAIN'; INEXT = 'SN'; INVER = 0 (to look at latest SN table), DPARM(3) = 6 (6 figures in output). DPARM(1) = 6 for delay. Change DPARM(1) = 7 for delay rate, and DPARM(1)= 1 for phase.

The delay and rate should be found within your window in the FFT. The windows are not hard-and-fast for the least-squares solution, as there is simply a penalty term equal to the delay over the delay window (similar in rate), and it may be able to find fringes outside the box. Be sure you understand the reason if fringes were not found for each solution interval of your data (*e.g.*, antenna off source, etc.), otherwise, CALIB has failed to fringe-fit all your data. One problem that has been encountered (in about 30% of the data we have fringe-fitted in the last six months) is finding the sidelobe of the main fringe right on the edge of your window; CALIB will be happy with the bump it found and stop there. Use the printout to identify cases where the fringes were found at the edge of the window; if you have any, increase the size of the window and rerun CALIB (after deleting the old tables).

If you decide you have done the calibration all wrong and wish to start over, it is easy to clean up. To start over, delete all CL tables with version numbers > 1 but NOT CL table 1, your original. Then delete all SN tables using EXTDES because CLCAL will apply them ALL to your data.

3.13 Apply the solution table

Applying the solution table (SN 1) to the CL table (CL 2) is done with task CLCAL. It will create a new CL table (CL 3). Choosing OPCODE CALI means that you want ALL the SN tables MERGED and all the phase-like parameters (phase, delay, and delay rate) referenced to a common reference antenna, as well as smoothing the solutions and applying them to the CL table. REFANT specifies which antenna to use for the reference; if you do not specify it, CLCAL will choose the one used in the most solutions. Choose:

OPCODE='CALI'

INTERPOL='BOX'

INTPARM= usually 3 times your solution interval in hours, *e.g.*, for 6 cm example, INTPARM=.3,.3,.3. You may need a longer interval for weak sources. In general, use whatever is needed to smooth over the occasional bad solutions.

SMOTYPE='VLBI' smooth delay and rate

GAINVER=2 CL table in

GAINUSE=3 CL table out.

3.14 Produce calibrated data

Now you finally get to make a calibrated dataset. This is done with task SPLIT, which applies the calibration and splits the database into files containing data on one source each. Set DOCALIB = 1, GAINUSE=3, for applying the calibration from the most recent version of the CL table (CL 3). Also set CPARM(1)=1 to average in frequency and CPARM(2)=2 for 2-second records. Set BLVER=−1 if you have no baseline-dependent calibration to apply (most VLBI experiments are unable to deal with baseline dependent errors; see Wilkinson's article in the Toulouse Conference on VLBI (1983) for a discussion of this fundamental source of errors in VLBI). There is one trick to this task: the outname of the file created is the name of the source; you cannot control this parameter. You can, however, choose the outclass: we suggest you use some memnonic such as the scan number (or stoptime), *e.g.*, '1300', if you are fringe-fitting each hour separately, to keep your data identifiable. Notice that SPLIT produces a single-source file with only HI and AN tables.

3.15 Run CALIB to correct the phases

CALIB in self-cal mode should be run to align the phases for subsequent averaging. We do not recommend letting CALIB edit data based on residuals at this point; large blocks of bad data were discarded when you ran UVFLG earlier while the model used here is not usually good enough to justify discarding points based on the solution. In any event, the bad points in your data set can be removed much more appropriately with IED and VLBEDIT in the Caltech package or TVFLG in AIPS. Note that if your model is horribly wrong, *e.g.*, if your source is a strong equal double and you are using a point source model, self-calibrating will likely result in CALIB throwing out a lot of your data as it tries to reconcile non-zero closure phases.

The phase self-cal could also be performed on the multi-source file with the generation of an extra set of tables and proper juggling of extension version numbers, however the CL table records are longer than the data records themselves (they have slots for as-yet unimplemented features). Phase solutions on the 2sec data records would require 2sec CL table entries (specified in MULTI) and the resulting MULTI file would be large.

In CALIB, set:

CALSOUR = ' '

SMODEL = 1,0

DOCALIB = −1

GAINUSE = 0

DODELAY = −1

SOLTYPE = ' '

SOLMODE = 'P!A' solve for phase

SOLINT = (2.0/60.0) Solve for phase in 2-second intervals This should probably be set as low as the strength of the source will allow, 2 sec for strong sources and up to minutes for weak ones. For short wavelengths, your solution interval may need to be rather short, *e.g.*, at 1.3 cm, a good averaging time may be 30 seconds. The symptom of too long an interval may be seen with (AIPS) VBPLT or (Caltech) VPLOT; the amplitudes show scatter over a region much larger than the error bars. *The value of this parameter must be chosen with care. We strongly recommend trying several values and comparing the results with VPLOT.*

APARM(1) = 3 Require 3 antennas present for solution

APARM(6) = 0 no printout

APARM(7) = 1 SNR needs to be low since it is calculated as phase difference from model which can be large.

ANTWT = 1 use weights from calibration For the purposes of phase alignment, it is appropriate to use the weights ANTWT = 1 which allows the noise in the solution to be put on the noisiest baselines. However, this is deadly if CALIB is being used for self-calibration in the hybrid mapping sense as the weights which reflect sensitivities will have a range of 100000 (!) and the resulting map will be effectively 1–2 baseline. In these cases it is necessary to use the routine WTMOD to change the weights (by replacing with wt * 1.E9 **0.25 for example) to reduce the range to a factor of 10–100.

3.16 What to do with Good Data

Your data are now ready for point-by-point editing and hybrid mapping.

First, print the history file for each final dataset and save it. Second, write your data to tape using FITTP. We suggest you do this for each scan in case you find signs of trouble later. You may also want to keep some of the intermediate datasets, but it is often easier to keep just the original data (uvf files) and the RUN files used for fringe-fitting. If you are going to use AIPS to reduce your data, see the AIPS cookbook for instructions.

If you intend to use the Caltech VLBI package to reduce your data, you need to convert your AIPS-format data to Caltech format. Run task FITTP to export your data in FITS format. Then use the Caltech programs FITSMERGE to convert to FITS format (specify NOHISTORY to discard rthe AIPS history file) and AVERAGE to average the 2-second points to your coherent integration time (*e.g.*, 60 seconds), using METHOD=S, TIME=60, to reduce your dataset to a manageable size. On a strong source, you may want to edit with IED before averaging.

If you wish to do your mapping in AIPS and take advantage of the TV routines and features of MX, then the proper averaging is done in UVAVG and it will be necessary to run WTMOD. The relevant calibration/mapping cycle is CALIB–UVSRT–MX, all in single-source mode. STM has had success with this method, except for editing, and results very similar to those produced by the Caltech package were obtained (albeit for snapshot data, AIPS should be superior for very high S/N cases).

Under some circumstances, you will want to re-fringe-fit your data, in which case read the next section.

3.17 When should you fringe-fit your data twice?

Several circumstances may lead you to fringe-fit twice. (*a*) If you have a weak source, you may run CALIB twice. See the help files for instructions in this case. It has been suggested that if your source is very weak, that you smooth solutions over 1–2 hours, then apply your solutions to CALIB another time. This will presumably allow the use of narrow windows. Calibrator sources could also be used to provide a first guess. (*b*) If some of your baselines were so weak that you did not detect anything on them the first time you fringe-fitted, you may be able to pull a detection out when you put in a good model for fringe-fitting. (*c*) If you have a strong source, you may need fringe-fit only once. However, if you find,

after mapping your source, that it is a strong double or otherwise not well-represented by a point source, you will wish to use your map as the input to CALIB in a second iteration, if only for the phase-alignment stage (the delays are not as sensitive to the choice of model, though rates might have a problem). You can use a map produced in AIPS through the CALIB–MX hybrid procedure or use IMLOD to read in a CLEAN map file from the Caltech package. Be sure that UVFIX was run (correctly) or you will get nothing useful.

If you have a very weak source, possibly the only hope is to use calibrators or to restrict oneself to the FFT solution (you will have only a few baselines strong enough anyway so forget about the global part).

If you fringe-fit using a model (*e.g.*, clean components from a map or a Gaussian), your data *must* have the same coordinates as the image file containing the model. Currently B2FITS places 1950 coordinates in the header but before October 1988 it used coordinates of date. For most applications the absolute position is irrelevant, and it is legitimate to change the position in the header of the image file to match that in the uv data, using a GETHEAD–PUTHEAD sequence to copy the coordinates from the data (usually CRVAL4, CRVAL5) into the map header (CRVAL1, CRVAL2). If your data has coordinates of date, for consistency's sake one can use the AIPS verb PUTHEAD to replace EPOCH, OBSRA, OBSDEC, CRVAL4, and CRVAL5 with the appropriate 1950 values (double precision, of course!).

The same fringe-fitting procedure as above is followed when a non-point-source model is used, except for the details of CALIB in both fringe-fitting mode and self-calibration mode:

Fringe-fitting: For a clean-components model, use IN2NAME = CLEAN map, IN2CL='MA', and so forth. Specify CALSOUR = 'sourcename' NMAPS = 1; NCOMP=0 (to use all CC's), DPARM(1) = 3 (3 baseline solution), the rest of the inputs should be the same as above.

Self-calibration: CALSOUR probably need not be specified but set SMODEL = 0, IN2NAME, etc. for the clean map, and NMAPS = 1; NCOMP = 0 (or to the number desired). Otherwise the same as above.

Acknowledgements: We are grateful to John Reynolds, Dayton Jones, Steve Unwin, Bob Preston, Dave Meier, and of course Bill Cotton for identifying and correcting bugs in the AIPS routines; and to Steve Unwin, John Benson, Bill Cotton, Phil Diamond, Tony Readhead, Tim Pearson, Marshall Cohen, John Biretta and Bob Garwood for comments. Toni Zensus developed the original RUN file on which this was based. Alan Bridle provided encouragement for documenting the black art of fringe-fitting. Much thanks to Bill Cotton and NRAO for supplying the information on AIPS and putting up with one of the authors (STM) for a week.

3.18 An AIPS run-file for fringe-fitting

```
***  EXPT P98G Jun 1990 6cm, Tim Pearson
*-----------------------------------------------------------------------
* Command procedure: global fringe fitting
*-----------------------------------------------------------------------
priority=0; prnumber=-1; prtask=' '; prtime=0; clrmsg
restore 0
antennas =  1,2,3,4,5,6,7,8,9,10,11,12
indisk 1; outdisk 1
baddisk 2,3
douvcomp 0
*-----------------------------------------------------------------------
* Compute uv coordinates (UVFIX): x.UVF -> x.UVFIX
TASK 'UVFIX'
inname 'P98J1_0804_A'; incl 'UVF'; inseq 0
outn inname; outs 0; outcl 'UVFIX'
uvfixprm(1) 1
go; wait
*-----------------------------------------------------------------------
* Make multisource file; 1 minute entries in CL table (MULTI): -> x.MULTI
TASK 'MULTI'
incl 'UVFIX'; inseq 0
outn inname; outs 0; outcl 'MULTI'
aparm 1, 0
go; wait
*-----------------------------------------------------------------------
* Default indexing (INDXR): modifies x.MULTI
TASK 'INDXR'
incl 'MULTI'
go; wait
*-----------------------------------------------------------------------
* Delay/rate fitting (CALIB): modifies x.MULTI, makes SN.1
TASK 'CALIB'
clr2n
calsour     = ' '
docalib     = -1
gainuse     = 1
flagver     = 0
smodel      = 0
dodelay     = 1
* refant = Bonn
refant      = 2
* solution interval 10 min
solint      = 10
* min 3 antennas; min SNR 7
aparm       = 3, 0, 0, 0, 0, 1, 7, 0
soltype     = ' '
solmode     = ' '
cparm       = 0
dparm       = 1, 1000, 200, 2, 0, 0
*              S    B   W   L  No   K   G  KP  PT  LA   O  Y1
antwt       =  5, 100, 25, 35, 35, 25, 40, 30, 30, 30, 40, 32
*
go; wait
```

```
*------------------------------------------------------------------------
* Apply calibration (CLCAL): modifies x.MULTI, creates CL.2
TASK 'CLCAL'
opcode      = 'cali'
interpol    = 'box'
smotype     = 'vlbi'
* Smoothing time in hours
intparm     = .3,.3,.3
gainver     = 1
gainuse     = 2
*
go; wait
*------------------------------------------------------------------------
* Make calibrated dataset; apply CL.2 (SPLIT): x.MULTI -> source.TMP
TASK 'SPLIT'
outcl = 'A1TMP'; outseq = 0
docalib     = 1
gainuse     = 2
flagver     = 0
aparm       = 1, 2, 0
*
go; wait
*------------------------------------------------------------------------
* Align phases (CALIB): source.TMP -> source.FIT
TASK 'CALIB'
inname '0804+499'; incl 'A1TMP'; inseq 0
outnam=inname; outcl 'A1FIT'; outs 0
calsour     = ' '
docalib     = -1
gainuse     = 0
flagver     = 0
* Point-source model
smodel      = 1, 0
dodelay     = -1
* Reference antenna = Bonn
refant      = 2
* Solution interval = 1 minute
solint      = 1.0
* Require 3 antennas; min SNR=1
aparm       = 3, 0, 0, 0, 0, 0, 1
soltype     = ' '
solmode     = 'P!A'
cparm       = 0, 1, 0
dparm       = 0
* Default weighting
antwt       = 0
*
go; wait
*------------------------------------------------------------------------
* Export data in FITS format (FITTP).
TASK 'FITTP'
incl='A1FIT'
outfil='SCR:P98J1_0804_A.FIT'
*
go; wait
*------------------------------------------------------------------------
priority=0; prnumber=0; prtask=' '; docrt=false; prtime=0; prtmsg
```

Chapter 4

HYBRID MAPPING

This chapter (based on a memorandum by Ann Wehrle) outlines the steps performed in a typical sequence of self-calibration and imaging with continuum VLBI data. Questions aimed at ensuring high image quality are posed in each section. Some criteria of good images are discussed.

4.1 Preliminaries

The first step is to examine your data for quantity and quality. After running FITSMERGE, run AVERAGE to average your data to 60-second intervals. Note that the averaging time is the coherent integration time. You may need to reduce or increase the averaging interval for frequencies other than 5 GHz. In AVERAGE, choose TIME=60, METHOD=S. The latter requests a coherent integration with the error bars obtained from the scatter in the data points in each averaging interval; this is the only valid choice for data from AIPS. You may wish to average data on a strong baseline with various averaging times and plot the resulting amplitudes as a function of integration (averaging) time if you wish to explore how long an averaging time you can get away with. There is usually no reason to do an incoherent average for making a map—only for model-fitting (where it can reduce the cpu time required considerably).

It is always a good idea to look at the raw data at this stage. Use VPLOT to examine the 2-second data points (amplitude and phase) on your calibration sources and perhaps also on the program sources. Specify NOERRORS to suppress the error bars. Apart from low or noisy points at the beginning of each scan, the data from a strong calibrator should show a gaussian distribution about a constant amplitude with zero phase. Some instrumental problems may show up: *e.g.*, periodic errors in amplitude or phase. Coherence problems will show up as correlated fluctuations in phase.

The following is a sample Unix csh script to convert a FITS file into Merge format and average it. It requires one argument: the name of the FITS file from AIPS, which is assumed to have an extension '.FIT'. It creates two merge files, with extension '.mrg' and '.avg'.

```
#/bin/csh
echo $1
rm -f $1.mrg $1.avg $1.ftls
fitsmerge <<EOD
  input $1.FIT
  output $1.mrg
  nohist
  list = $1.ftls
  /
EOD
```

```
average <<EOD
  input = $1.mrg
  output= $1.avg
  time=60
  method=s
  /
EOD
```

Collect all the data for each source

Run MERGE selecting all stations and combining all data sets; it should calculate new closure phases for you consistent with the averaging time you used. If you have only one data set already, you don't need to run MERGE, just run ADDCLO to calculate the closure phases which you only need for plotting and model-fitting anyway. (Reminder: although the closure phases calculated and attached to the data points are *not* used in hybrid mapping, the closure phases are recalculated each time you run AMPHI so that the phases can be adjusted and recalculated for the new AMPHI data set. While it is easier to think that the closure phases are calculated once and for all and carried around as baggage, in fact, the closure relations are used to recalculate the closure phases every time you run AMPHI.) Then run LIST with HEADER specified to find out what you've got. Are all antennas present? Are their coordinates reasonable (not zero, at least)? Are the coordinates in epoch 1950? If not, examine the inputs to FITTP and MERGE to see what went wrong. If the coordinates are not epoch 1950, you *may* need to go back and run B2FITS again if you decide you want to fringe-fit with a model.

Inspect the data

Now run VPLOT or VISPLOT to get a good look at the data. This is tedious but absolutely essential. Plot amplitude, phases, and closure phases. For amplitudes and phases, use a time axis of UTM or UT so you can identify suspicious times right away. Save the output. Do the phases undergo any sharp jumps, or worse yet, do the closure phases show sharp jumps? If so, your fringe-fitting was bad. (Note: A compact double also produces sharp jumps in the closure phases.) Does data exist at each interval at each baseline where your antennas were tracking? If not, look at your fringe-fitting output and the antenna logs to figure out what happened.

Edit the data

Next, use VLBEDIT to remove bad data at the start of each scan when the antennas were doing calibration or pointing. If you were efficient and used UVFLG to get rid of bad data in AIPS, but then allowed all flagged data to be passed by FITTP, you have preserved the integrity of your solutions, but you still need to get rid of bad data! You can do this by going back to AIPS and setting the parameter that allows flagged data to be passed or you can just delete your bad data with VLBEDIT; going back to AIPS is probably better. I usually delete the first five or six minutes of each scan with VLBEDIT after using LIST to identify the record numbers of data at the start of each scan. This takes about 10 minutes. Then use IED to remove individual bad data points; it will take you between 15 and 60 minutes for the average-quality data set. When you are finished, run VPLOT or VISPLOT again and save the output.

Calibrate the data

If you have not calibrated your data with ANCAL, use CAL now. (Some people prefer to run CAL before editing.) Keep careful records of your calibration; you will probably have to modify the calibration parameters and re-run CAL after looking at the calibration sources. Examine the plots produced by CAL to check that you made no gross errors in entering the calibration parameters. (If you used ANCAL before fringe-fitting and find errors at this stage, you will have to go back and redo the fringe-fitting.)

Inspect the data again

Now check your *uv*-coverage with VPLOT (or UVPLOT). This may be a considerable surprise to you after all your scheduling efforts months ago! You can also use VPLOT (or UVDIST) to get an overview of your data from which you may be able to infer something about the basic structure of your source. UVPLOT can produces a one-page plot of all your data with flux density versus radial *uv*-distance; bad points will stand out like sore thumbs. It is often useful to superimpose a simple gaussina model (*e.g.*, a single Gaussian of FWHM 1.0 mas (for global arrays at 6 cm) and flux density equal to the total flux in your source).

4.2 Mapping

Now begin the AMPHI–INVERT–CLEAN cycle. It is probably a good idea to treat the first iteration through hybrid mapping as an experiment, to get things right; don't feel inhibited about going back to the original data and starting over.

Although it is possible to start with a point source model, there are very good reasons for doing model fitting at this stage to get a good starting model.

AMPHI

Your goal in first run of AMPHI is to straighten out the phases as best you can, knowing nothing else, a priori, about the source. You only want to let AMPHI adjust the PHASES this time around, so leave CLAMP and SCALE off. SCALE should be turned on after a couple iterations of the mapping cycle. If you have SCALE'd your data, the relative gains of each station will be listed in the output of AMPHI. They should be of order 10%. If they are larger than 10%, your calibration is screwed up. Check the gain curves. CLAMP is useful near the end of the mapping cycle. If you have a very good model, use a short timescale (1–10 minutes) to correct rapidly-changing gain errors (mispointing, atmosphere); otherwise use a long timescale (> 60 minutes). Be careful: if you have few antennas, or if you are not careful, AMPHI can overcorrect the data and force them to fit the model.

A powerful way to increase your dynamic range is to use the gain factors derived from one SCALE operation on an unresolved "calibrator source" to improved the calibration on your interesting (not unresolved) source. You can use the gain factors derived with the SCALE option as new fudge factors in CAL (in CAL, these are referred to as FT's—note that the FT factors should be modified by the squares of the factors determined by AMPHI) or as multiplicative factors in VLBEDIT (easier and quicker). You cannot use this technique if you have no calibrator source unless you have many stations for which you can use the closure amplitudes.

In addition, UVCROSS can be used to determine multiplicative correction factors that can be applied with VLBEDIT, if your dataset has crossing points or closely parallel baselines.

INVERT

Make a dirty map and dirty beam. I plot the beam and the map to the same scale by setting PLOTBEAM and PLOTMAP to the same values; this is a matter of style and not mandatory. Make a 256×256 map (the default) to start with, unless you know the source requires a bigger map. IDIM is twice the size of the map in pixels or cells. Choose the gridding XYINT, also known as the cellsize or pixel size, to make sure you have at least 2.5 or 3 cells across the beam at half-power point, in both x and y directions. If you choose a smaller number you will get aliasing and *your maps will be complete garbage.* Check the INVERT printout to be sure—the oversampling factor should never be ignored. MAXP should be set to some large number like 5000 or 10000; at least as many as there are data points (see LIST or UVPLOT to get this number). Too small a number for MAXP will mean you are excluding some of your data, a bad idea. The contours default to reasonable values, but I like to put in two contours at 50% and 52% so the important full width at half maximum is easier to measure with a ruler.

When you get your listing from INVERT, get out your ruler and protractor and measure the dirty beam. It is difficult to calculate the dirty beam in VLBI due to undersampling the plane. Do not trust the dirty beam characteristics printed out by INVERT; they are nearly always wildly wrong for low-declination sources. If you find several peaks over 50%, you are going to have a tough time deciding what to use as a CLEAN beam; there is no "right" answer. You will just have to resign yourself to noting the funny dirty beam shape in your publication and describing the resulting ambiguities in your image (see discussion below about reliability). Maybe Maximum Entropy deconvolution will help your image.

[The dirty map and dirty beam produced by INVERT are not in FITS format and thus cannot be displayed; eventually this will be changed. An interim program NEWINVERT produces map and beam in FITS format.] I'd recommend that you load the dirty map and the dirty beam into AIPS using IMLOD and use TVBLINK to compare the two of them. The goal is to decide if your source looks like a point source or looks more complicated. How many peaks are in your dirty map? Be extremely suspicious if you find you have a double in your dirty map when you know your source is virtually unresolved at this frequency. Your first command file stops here so you can examine the results with a critical eye and your ruler.

CLEAN

CLEAN is probably the area with the most room for creativity in the imaging-self-calibration sequence. For a TITLE, choose something descriptive, including the iteration number. I like to generate a CLEANMAPFILE (which is in FITS format) which I can display on TV and look at the residuals. Looking at the residuals and the clean component model on the TV will save much time if you can identify patterns of sidelobes that are not being cleaned up properly in your CLEANing. A TV image is much more informative than a contour plot; use both if you have a TV as well as a printer. Set the BEAM to the FWHM parameters you measured on the contour plot of the dirty beam. For a first CLEAN, try setting NITER to 500 iterations and LOOPGAIN to 0.1. Set NOSQUASH to avoid squashing all the components together, since you will want to use only the components found before the first negative in your next AMPHI. Print out the first 50 components or so with NCPRINT. The contours on the plot should be set as above or your own favorite combination.

For a first iteration, use no window. Later on, you can experiment with windows; CLEAN allows multiple windows which are good for extended sources like core-jet structures. Plotting the window is a good idea. Most of the creativity in CLEAN comes from the choice of windows. Should windows be drawn tightly around the components you think are most real or should you be unbiased about where the source is? From my experience, the answers to these questions depend on how much *uv*-coverage you have (snapshots—tight window), the declination of the source (experiment), the structure of the source (if simple, use MODELFIT to find good model, use model as input to AMPHI, then use tight windows; if complex, use multiple windows), the quality of your calibration (bad calibration, use tight windows to start) and the amount of extended structure (fuzz) your source has (lots of fuzz, use tight window to start, enlarge window as your maps improve). You will just have to experiment; CLEAN is fast and you can try many modes of windowing in a day. A suggestion: you can locate the main components with a map of low resolution (and appropriately large pixels) by using UVCUT in INVERT to restrict the *uv*-range to the central part of the *uv*-plane. Then you can set the windows around those components in your high-resolution map. Some experienced VLBI people use MODFIX to pick out "good" components by specifying LRTB; this is a form of ex post facto windowing. Using the CLEAN parameter PLOTD(eltas) will plot exactly where the CLEAN components are located.

Are your windows optimized? If windows exclude real, significant flux density, you will not be able to make a good map. If your windows are too large, you will find it impossible to eliminate spurious symmetric and antisymmetric structures. If you have trouble fitting windows around a source, you may want to ROTATE the image in INVERT to make the source fit better in an upright rectangle.

If you get a really terrible map (e.g., first component negative, components all over the map) something has gone wrong. Things to check include oversampling, was the dirty beam file made from exactly the same dataset as the dirty map, do the windows include the strongest peaks, did AMPHI do something sensible (check the before and after chisquareds). Possibly your phases are wrong (see note on averaging and closure phases above).

Examine the Fit

After each CLEAN finishes, you should *always* plot up the fit of the model to the amplitude and phase data using VPLOT or VISPLOT, especially after using SCALE or CLAMP in AMPHI. Does your model visibility go through the error bars on all the data? Perfection is not possible, but your model should fit the maxima and minima of the data quite nicely. If some baselines just are not fit well, ask yourself if they are the shortest baselines; if so, you may have a source which is quite extended—try making a map with data from just the central part of the *uv*-plane. Another useful strategy is to remove your weakest (noisiest) baselines for the first few iterations; you can go back to the original dataset after you've got a reasonable fit. Use VLBEDIT to delete all data on selected baselines or stations; you can also use BASEDIT to make the resulting file smaller if you want to. QFIT is a useful program for deciding which baselines are most suspect.

Last, you should plot the best map's clean components with the *original* data to see if your map is way off track. I must stress that no VLBI map should be presented as a final product until you have checked to see how well it represents the data.

4.3 What is a good map?

The ambiguity of VLBI images is legendary. While it would be desirable to have some definitive objective criterion which would allow a stamp of "approved" on your image, no such criterion exists. We do have some ad hoc criteria. Unlike VLA images, most VLBI maps are limited not by the thermal noise, but by dynamic range; that is, their ultimate quality depends on the reliability of the calibration, instrumental effects like mismatched bandpasses being beyond the mapmaker's control. You should look at the thermal noise estimates produced by INVERT, which are not, unfortunately, the thermal noise per beam area, but which are based on the size of the error bars in the data, to see if you are expecting unreasonably good results from your data. Reminder: amplitude errors produce symmetric patterns while phase errors produce antisymmetric patterns. See Wilkinson's article in the Toulouse Conference for an extensive discussion on "The Attainment of Higher Quality Maps from VLBI".

Experience has shown that the most reliable maps are characterised by the following:

1. Simplicity. Most VLBI sources mapped to date have been a few blobs easily modelled by Gaussians. "Ears" that appear on the blobs are usually artifacts that can be removed with tighter windows. Sources with long skinny jets naturally aren't simple; you may strike it rich and get one of these to map!
2. Lack of symmetry. Symmetric patterns are a symptom of errors in the amplitude calibration; they can be removed with careful use of clean components in AMPHI and one-sided windowing in CLEAN. The appearance of a "jet and counterjet" symmetric around a central source is very common in early stages of mapping what turns out to be a one-sided core-jet, but we have never yet found a genuine two-sided VLBI jet! If your source is at declinations greater than 30 degrees or so, your map will have roughly circular sidelobes; this makes circular features in your map very suspicious.
3. Absence of deep negative contours. Negative contours are unphysical. If your source has extended structure which is not sampled well by the *uv*-coverage of the observations, you will probably have some negative clean components. They should not exceed a few percent of the peak in the image. If you have strong negative components, there are problems in your amplitude or phase calibration. Plot the original visibilities with your current model and look for places where the model misses the data by significant amounts. Look at your gain curves—are there any typos? Try plotting up your gain curves with PLOTGAIN.
4. Beauty-is-truth. An "ugly" map has lots of positive and negative contours spread around the map. These represent flux density which is not going into the "right" components but instead is being spread around the map by the mapping process. Use a smaller window in CLEAN, run more iterations of the AMPHI–INVERT–CLEAN cycle, and CLEAN deeper. It is easier to define an "ugly" map than a "pretty" map.
5. Absence of features with the same spacing as your sidelobes. The algorithms have trouble with dirty beams with several peaks over 70% or so. If your source is at low declinations and you have Hartebeesthoek, you will have to be very careful of features with the same spacing as the peaks in the dirty beam. CLEAN has a tough time deciding what is real and where the flux really ought to go. You will have to be much more critical in picking out the clean components you think are real than if you had a northern source and no Hartebeesthoek.

4.4 Advice on finishing up

At this point you may feel somewhat discouraged because your maps are not as good as you would like. It is appropriate to reflect on what you are trying to achieve with your maps. Are you trying to obtain a dynamic range of 1000:1 with five antennas? This is not realistic. For images with 1000:1 dynamic range, you need excellent calibration and many antennas. You cannot produce a high dynamic range image with two hours of data! If your CLEAN components do not fit the visibilities on the short baselines, compare a map made at some lower frequency to see where the "extra flux" is. You may wish to make a lower-resolution image from the same data for more information on the source. Are you bothered that your CLEAN components do not add up to the total flux? Aside from the fact that the flux density scale is uncertain by 10% or so, your source may contain lots of flux density at arcsecond (or larger!) scales. Compare your map to a VLA, WSRT or MERLIN map. If you are new to VLBI, take a look in the literature at some recent VLBI maps- they may surprise you. You could easily spend months mapping each source and produce dozens of maps which differ from your first good map by only microscopic details (none of which will be believed by your VLBI colleagues, anyway!)—this is a waste of time.

If you have had trouble with the calibration, check to see if you have the best gain curves for the antennas. Perhaps the antenna surfaces have been adjusted since the gain curve was last computed. The friends-of-VLBI will have sent you the most up-to-date information on their antennas—do not ignore their advice and caveats. Try contacting someone who observed in the same VLBI run and at the same frequency as your experiment to see if they have solved problems similar to yours.

Finally, ask yourself if your best map addresses the physics problem you posed in your proposal. If it does, stop mapping, do the calculations to analyze the nature of the radio source and write your paper!

4.5 Sample command file

The following example (in Unix csh syntax) was used for the first iteration of hybrid mapping of a source observed at 5 GHz with the EVN. The shell variable ITER is used to change file names in each iteration.

```
#/bin/csh
cd /scr/tjp/jun86
set ITER=1
set PREV=0

amphi <<EOD
  INPUT   = 0954_556.edt
  MODFILE = test12.mod
  OUTPUT  = 0954_556.adj$ITER
  LISTFILE= 0954_556amp.lis
  /
EOD
```

```
invert <<EOD
  INPUT   = 0954_556.adj$ITER
  MAPFILE = 0954_556.map
  BEAMFILE= 0954_556.bmm
  PLOTFILE= 0954_556inv.plt/PR
  LISTFILE= 0954_556inv.lis
  XYINT   = 1.0  MILLIARCSEC
  FLUX    = 1.3
  PRINTUV
  PLOTMAP
  PLOTBEAM = 20
  IDIM=512
  /
EOD
clean <<EOD
  SIZE     = 256
  MAPFILE  = 0954_556.map
  BEAMFILE = 0954_556.bmm
  PLOTFILE = 0954_556cln.plt/PR
  LISTFILE = 0954_556cln.lis
  BEAM     = 5.0
  PRINTRESIDUALS
  NITER    = 500
  LRTB     = -8,9,40,24, -11,14,27,5, -10,25,10,-18
  MODEL    = 0954_556.mdl$ITER
  CLEANMAP = 0954_556.clm$ITER
  PLOTWINDOW
  PLOTMAP
  CONTOURS = -2, -1, 1, 2, 4, 8, 16, 32, 64
  TITLE = "Iteration $ITER ($0)"
  /
EOD
rm 0954_556.map 0954_556.bmm
print -r 0954_556amp.lis 0954_556inv.lis 0954_556cln.lis
lpr -v -r 0954_556inv.plt
lpr -v -r 0954_556cln.plt
exit
```

Chapter 5

MODELFITTING

5.1 The Need for Model Fitting

In conventional aperture synthesis, with N stations and $N(N-1)/2$ baselines, the imaging steps are:

1. Fourier transform the $N(N-1)/2$ complex visibilities to produce a "dirty map";
2. Deconvolve the dirty map with the dirty beam by means of CLEAN or maximum entropy.

In VLBI, however, instead of $N(N-1)/2$ reliably measured phases we have N arbitrary phase offsets, one for each antenna. One of these can arbitrarily be set to zero, so that there are $N-1$ undetermined phases. Another way of looking at this is that there are $(N-1)(N-2)/2$ measured closure phases, and therefore phases have to be invented on $N-1$ baselines in order to determine the complete set of $(N-1)/2$ phases. The $N-1$ undetermined phases *can in principle be set to any arbitrary value* and still yield an image that will fit the data to any level of accuracy you may choose to specify: *i.e.*, there are infinitely many brightness distributions that fit the data.

The trick is to find that solution which best approximates the true distribution subject to the observations. We do this be making use of information that is ignored in conventional mapping: (1) The sky brightness cannot be negative (apart from noise); (2) The source has a finite size, or finite "support". In addition, experience has shown that, for compact extragalactic objects, a simple structure that fits the data is more likely to be correct than a complex structure that fits the data equally well.

It should be clear that VLBI will always require a lot more care and judgement than VLA mapping, because there is an element of subjectivity in VLBI mapping that is entirely absent from VLA mapping. In order to guide our judgement it is therefore very important to get a feel for what the data demand of the source, i.e., overall size, orientation, shape, number of components, etc. Model fitting provides a powerful technique for doing this.

In hybrid mapping, there is no substitute for accurate calibration and a good starting model. The quality of the final map nearly always depends on these two factors.

The reasons for model fitting can be summarized:

1. It gives the astronomer a good feel for the source structure, which is very important in the subsequent mapping stages.
2. It provides a powerful test of the calibration—if it is possible to find a simple model that fits all the amplitudes and closure phases well (agreement factor < 2) then the calibration is probably reasonably good. Problems with calibration (*e.g.*, a 20% offset in the amplitude calibration of one telescope, or an inconsistent pair of visibility measurements at a crossing point) often show up in model fitting. Thus the calibration cannot be regarded as finished until reasonable models have been made of all the sources.

3. It enables the astronomer to tailor the mapping step to suit the source, and choose appropriate values for the map cell size (XYINT) and array size.
4. It speeds up the convergence of the hybrid mapping and often improves the quality of the final map. In other words, it provides some confidence that we have found an acceptable approximation of the true sky distribution subject to the observations.
5. It provides an invaluable parameterization of the brightness distribution. This is useful both for comparing structural changes between epochs, and for carrying out physical calculations. (An alternative way of determining physically useful parameters is to fit gaussian components in the image domain, using AIPS.)
6. Depending on the complexity of the source, there are a number of further arguments for model fitting.
   (*a*) For *very compact sources*, mapping with a point source start will clearly give a good map. The value of the model fitting is that it enables you to search for features smaller than the beam size, and provides a straightforward test for weak, extended structure. In the most compact sources inspection of, and model fitting to, the closure phases and closure amplitudes can be particularly useful.
   (*b*) In *double sources* with a separation less than the beam width, model fitting can be used in conjunction with deconvolution using CLEAN or MEM to test the reliability of the derived separation. It is advisable to use all three methods.
   (*c*) Model fitting in *extended sources* can be particularly useful in locating weak compact features embedded in much brighter, resolved features.

The only reason that can be advanced against careful modelling is that it appears to be time-consuming. However this is nearly always not the case—you will often spend a lot more time hybrid mapping, and trying to convince yourself that you have the best possible map, if you don't pull out all the stops on model fitting, so that the time spent on model fitting is generally more than repaid in the subsequent mapping stages.

5.2 Practical Model Fitting

The actual practice of model fitting is simple for most sources, but there is a small subset ($\sim 10\%$) for which it is tedious, and there is no simple practical recipe that guarantees a good model. The usual procedure involves three steps.

Locating the maxima and minima in the (u, v) plane. Program UVPLOT can be used with the LABEL option to display tracks in the (u, v) plane and identify the baselines. You can mark maxima and minima in this plane by comparison with the run of amplitude versus time produced by VPLOT or VISPLOT. It is most useful to plot amplitude versus interferometer hour angle (IHA) in VISPLOT. Remember that there is an absolute maximum at the origin.

Source complexity and estimation of parameters. If the source is simple (a single component) the maxima and minima will be separated by about 6 hours. If the source is more complex there will usually be more frequent maxima and minima. For a single component, it is straightforward to measure the width (FWHM) in the visibility plane, and hence derive the dimensions of the component in milliarcseconds (see Figure).

In the case of a complex source, we next try to fit a double. The maxima for a simple double source lie on a series of parallel straight lines in the (u, v) plane with constant separation. Again the component separation can be deduced from the Figure. A fairly common feature is a series of parallel loci of maxima whose spacing decreases with distance from the origin—indicative of a double with hot spots (more compact components) at the outer edges of the main components.

For simple doubles the relative flux of the components can be deduced from the relative heights of the maxima and minima. However, for more complex doubles this is non-trivial since resolution and orientation affect the amplitudes. Make you best guess of the relative fluxes and sizes of the components, and let the model-fitting program find the best values.

For a really complex source there may be evidence in the locations of the maxima and minima of a double that is only visible on intermediate baselines, for example. The separation can then be found, but it may be difficult to position this double relative to the rest of the source.

Model fitting with MODELFIT. The model-fitting program is very powerful, but it is finding minima in a multidimensional parameter space. There are generally many minima in this space, and it is quite easy for model fitting to get stuck in a local minimum that is a long way from the absolute minimum that you are searching for. The model-fitting program therefore needs all the help it can get.

Keep the number of components small. Only introduce a new component when you have exhausted the possibilities of MODELFIT and have found a hard local minimum.

Check your progress frequently by plotting out the fit to the data (VPLOT). MODELFIT may be pushing the solution in a direction that you know is wrong for some reason. Also plot a contour map of the model (MODPLOT) fairly often to be sure that you are getting a plausible model, consistent with your original deductions from the maxima and minima.

The hardest part of model fitting is finding the general region of the absolute minimum. Once you are in the right region, MODELFIT will force you to a good solution very rapidly. However, to begin with (generally with agreement factors above about 4), you will have to be selective about the parameters that you vary. Note that some parameters are highly correlated, and should be allowed to vary simultaneously (e.g., flux and component diameter). You will find that your ability to drive MODELFIT improves rapidly with practice.

The visibility functions for various brightness distribution models; from "Interferometry and Aperture Synthesis" by E. B. Fomalont and M. C. H. Wright (1974), in *Galactic and Extra-Galactic Radio Astronomy*, ed. G. Verschuur and K. Kellermann, p. 287.

Chapter 6

DEFINITIONS AND CONVENTIONS

6.1 Station Coordinates

Station coordinates are specified in a left-handed geocentric Cartesian coordinate system (X, Y, Z):

X – directed towards the Greenwich meridian (positive for stations with longitude between $-90°$ and $+90°$),

Y – directed towards longitude 90° West (positive for all American stations, negative for most European stations),

Z – directed towards the North pole (positive for northern hemisphere stations).

The *geocentric* latitude ϕ and longitude λ of the station are related to X, Y, Z by:

$$\begin{aligned} X &= R\cos\phi\cos\lambda, \\ Y &= -R\cos\phi\sin\lambda, \\ Z &= R\sin\phi. \end{aligned}$$

Note that longitude is conventionally positive when measured towards the East. The *geodetic* latitude is measured relative to the geodetic vertical (normal to a spheroidal approximation to the oblate Earth) and thus differs from the geocentric latitude.

6.2 Baseline Vector

The baseline vector $\overleftarrow{AB}$ is directed from station B to station A, and its components are expressed in the same system as the station coordinates:

$$\begin{aligned} \mathbf{b} = \overleftarrow{AB} &= (b_X, b_Y, b_Z) \\ &= (X_A - X_B, Y_A - Y_B, Z_A - Z_B). \end{aligned}$$

The baseline vector $\overleftarrow{BA}$ has the opposite sign to vector $\overleftarrow{AB}$.

In a MERGE-format file containing data from the ordered set of stations A, B, C, D, ..., the baseline vectors are $\overleftarrow{AB}$, $\overleftarrow{AC}$, $\overleftarrow{BC}$, $\overleftarrow{AD}$, $\overleftarrow{BD}$, $\overleftarrow{CD}$, ... (i.e., the vector is always directed from the higher-numbered station to the lower-numbered one), and the baselines usually appear in the file in this order, although this is not a requirement. It is usually convenient to order the stations by longitude.

This definition of baseline vector is opposite to the conventional one (e.g., Fomalont and Wright, "Interferometry and Aperture Synthesis", in Verschuur and Kellermann, *Galactic and Extragalactic Radio Astronomy*).

6.3 Interferometer Hour Angle (IHA)

Interferometer Hour Angle is defined by

$$IHA = H - h - \pi/2$$

where H is the Greenwich Hour Angle of the source ($H = GAST - RA$, where $GAST$ is the Greenwich Apparent Sidereal Time and RA is the Right Ascension), and h is the Greenwich Hour Angle of the baseline, defined by

$$h = \arctan(b_Y/b_X);$$

h takes all values between 0 and 2π. If the IHA of a source observed on baseline $\overleftarrow{AB}$ is θ, the IHA on baseline $\overleftarrow{BA}$ is $\theta + \pi$.

6.4 Baseline Coordinates (u, v, w)

u is the East component and v is the North component of the projected baseline vector seen from the source, expressed in wavelengths. w is the component of the vector in the direction of the source:

$$\begin{aligned} u &= \lambda^{-1} b_{\rm eq} \cos(IHA) \\ v &= \lambda^{-1} \left(b_Z \cos\delta + b_{\rm eq} \sin\delta \sin(IHA)\right) \\ w &= \lambda^{-1} \left(b_Z \sin\delta - b_{\rm eq} \cos\delta \sin(IHA)\right) \end{aligned}$$

where $b_{\rm eq} = \sqrt{b_X^2 + b_Y^2}$. Note that u, v, and w change sign if the baseline is reversed.

6.5 Sky Coordinates (x, y)

x and y are coordinates relative to the phase reference position (α_0, δ_0). x is a displacement to the East and y a displacement to the North. For VLBI maps the sky may be assumed to be flat. In terms of radius and position-angle coordinates (r, θ) with position angle measured from North through East,

$$\begin{aligned} x &= r \sin\theta, \\ y &= r \cos\theta. \end{aligned}$$

6.6 Phase

Visibility is related to sky brightness by

$$\begin{aligned} V(u,v) &= \int dx \int dy\, I(x,y) \exp\left(+2\pi i(ux+vy)\right), \\ I(x,y) &= \int du \int dv\, V(u,v) \exp\left(-2\pi i(ux+vy)\right). \end{aligned}$$

These formulae define the convention for sign of phase adopted by the Caltech VLBI programs. The first equation is used whenever the response to a specified source distribution is calculated (programs VISPLOT, MODELFIT, MODPLOT, AMPHI, etc.), and the second equation is used for aperture-synthesis map-making (program INVERT).

6.7 Closure Phase

The closure phase observed with a triangle of three telescopes A, B, C is defined as the algebraic sum of the visibility phases around the triangle:

$$\psi(ABC) = \phi(\overleftarrow{AB}) + \phi(\overleftarrow{BC}) + \phi(\overleftarrow{CA}).$$

Because $\phi(\overleftarrow{BA}) = -\phi(\overleftarrow{AB})$, ψ is unchanged by an even (cyclic) permutation of the telescope labels,

$$\psi(CAB) = \psi(ABC),$$

but it is inverted by an odd permutation:

$$\psi(BAC) = -\psi(ABC).$$

6.8 Orbital Parameters

When orbiting stations are used, the station X, Y, Z coordinates are functions of time; these functions are decribed by the *orbital parameters*.

There are six orbital parameters plus the coordinate system in which they are measured (X', Y', Z') and the date on which they are measured (the epoch of the orbit):

- A – the semi-major axis of the ellipse
- e – the eccentricity
- i – the inclination, i.e., the angle between the orbital plane and the $X'Y'$ plane; 0–90 degrees: *prograde* orbit; 90–180 degrees: *retrograde* orbit.
- Ω – longitude of ascending node; the ascending node is the place where the orbit intersects the $X'Y'$ plane and where the projection of the spacecraft velocity vector on the Z' axis is positive; longitude is measured eastward from the X' axis.
- ω – argument of periapsis (in this case, perigee); this is the angle measured in the orbital plane in the direction of orbital motion from the ascending node to the point of *closest* approach to the earth.
- M – mean anomaly; the *true* anomaly T is the angle of the spacecraft from the periapsis in the orbital plane in the direction of orbit. Except for circular orbits, this does not vary linearly with time. The *mean* anomaly M is also an angular coordinate that varies from 0 to 360° over one orbit, but it *does* vary linearly with time. (It is in fact proportional to the area swept out by the spacecraft vector.)

The primes on the above coordinate frame are used to distinguish it from the non-inertial, left-handed system in which station coordinates are expressed. It is convenient if the X', Y', Z' frame is the celestial coordinate system (earth equator = $X'Y'$ plane and equinox = X') on the date of the experiment, but it is not necessary.

To fully specify the orbit, one also needs the date and time at which the above parameters have their stated values (Ω and ω change slowly with time and M varies rapidly during one period). The epoch date of the orbit is not necessarily the same as the epoch date of the coordinate system in which the elements are measured. However, to compute u and v for a given experiment, the orbit must be *propagated* to the experiment date, using suitable orbit propagation routines, and the coordinate system must be precessed if it is not earth equator and equinox of date.

Chapter 7

MODEL FILES

7.1 Model file format

Model files are text files that can be typed or printed directly; they can be modified or created using the standard text editors. A model file consists of one line for each component of the model, with up to 7 numbers on each line (in free format):

1. Component flux density S (Jy).
2. Distance of center of component from origin r (milliarcsec).
3. Position angle of center of component with respect to the origin θ (degrees, North through East).
4. Major axis of component a (milliarcsec). For all component types, this is the full width, not the half-width; for circular components, it is the diameter.
5. Axial ratio α (minor/major, *i.e.*, < 1).
6. Position angle of major axis ϕ (degrees, North through East).
7. Type:
   - 0 or 1: delta function (if major axis is 0) or elliptical gaussian (major axis is FWHM);
   - 2: elliptical disk of uniform brightness;
   - 3: optically thin ellipsoid;
   - 4: elliptical ring of zero thickness;
   - 5: rectangle of uniform surface brightness (major axis is length of longer side; axial ratio is ratio of sides).

If less than 7 numbers are specified, the omitted numbers are taken to be zero. Thus for gaussians and delta-functions, the Type can be omitted; for delta-functions, the major-axis, axial-ratio, position-angle and type can be omitted. (Not all the programs understand types 2–5.) The *origin* is an arbitrary phase-reference point. The maximum number of components varies depends on the individual program.

Blank lines and lines beginning with an exclamation mark (!) in a model file are ignored.

The following is an example of a model file. A comment is included to label the columns. There are two gaussian components, with the stronger one situated at the phase center and the weaker one 4.29 milliarcsec away:

```
!  Flux     Radius    Theta       Axis      Ratio      Phi  Type
 1.6816     0.0000     0.00     0.7985    0.11917     4.88     1
 0.1840     4.2899   194.47     1.4077    0.57033   121.14     1
```

7.2 Model computation in the (u,v)-plane

Models are defined in the sky plane or (x, y)-plane. Given a model brightness distribution $f(x, y)$, the model visibility $F(u, v)$ is computed by Fourier transformation:

$$F(u,v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y) \exp[2\pi i(ux + vy)]\, dx\, dy. \tag{1}$$

A model is a sum of *components*,

$$f(x,y) = \sum_i S_i f_i(x - x_i, y - y_i), \tag{2}$$

where S_i is the flux density of the ith component, (x_i, y_i) is its nominal position in the sky plane, and f_i is a normalized function,

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f_i(x,y)\, dx\, dy = 1. \tag{3}$$

To evaluate equations 1 and 2, we use the *linearity* of the Fourier transform and the *shift theorem*:

$$F(u,v) = \sum_i S_i F_i(u,v) \exp[2\pi i(ux_i + vy_i)]. \tag{4}$$

We now consider the form of $f_i(x, y)$ and $F_i(u, v)$ for each of the standard component types, beginning with circularly symmetric components. We can transform from circular to elliptical components by using the *similarity theorem*,

$$\mathrm{FT}\{f(ax, by)\} = \frac{1}{|ab|} F\left(\frac{u}{a}, \frac{v}{b}\right). \tag{5}$$

e.g., if $F(u, v)$ is the visibility of a standard circular component, the visibility of a component with axial ratio (minor/major) α and major-axis position angle 0 is $\alpha F(\alpha u, v)$. For a rotated component, with major-axis position angle ϕ, the visibility is the same as that of an unrotated component measured at (u', v'), where

$$u' = u\cos\phi - v\sin\phi,$$
$$v' = u\sin\phi + v\cos\phi.$$

The following paragraphs define the functional form of the standard model components. Note that because $f(x, y)$ is normalized, $F(0, 0) = 1$. For circularly symmetric components, we write $f_i(x, y) = f_i(r)$ and $F_i(u, v) = F_i(\rho)$, where $r = \sqrt{x^2 + y^2}$ and $\rho = \sqrt{u^2 + v^2}$. It is straightforward to show that for an elliptical component with axial ratio α and position angle ϕ, $F(u, v) = F(\rho')$, where

$$(\rho')^2 = \alpha^2(u\cos\phi - v\sin\phi)^2 + (u\sin\phi + v\cos\phi)^2.$$

0. Delta function The major axis parameter a is 0 (any component with a=0 is evaluated as though it were a delta function).

$$f(x,y) = \delta(x,y),$$
$$F(u,v) = 1.$$

1. Gaussian a = full width to half-maximum intensity (FWHM).

$$\begin{aligned} f(r) &= \frac{1}{\sqrt{4\pi/\ln 2}a} \exp\left(\frac{-4\ln 2\, r^2}{a^2}\right) \\ F(\rho) &= \exp\left(\frac{-(\pi a\rho)^2}{4\ln 2}\right) \\ &= 1 - \frac{z^2}{4\ln 2} + \frac{z^4}{2(4\ln 2)^2} - \ldots \end{aligned}$$

where $z = \pi a\rho$.

2. Uniformly bright disk a = diameter.

$$\begin{aligned} f(r) &= \begin{cases} 4/(\pi a^2), & \text{if } r \le a/2 \\ 0, & \text{otherwise} \end{cases} \\ F(\rho) &= \frac{2J_1(\pi a\rho)}{\pi a\rho} \\ &= 1 - \frac{z^2}{8} + \frac{z^4}{192} - \ldots \end{aligned}$$

where $z = \pi a\rho$.

3. Optically thin sphere a = diameter. This is sometimes called a 'tapered disk'; the brightness at each point is proportional to the path length through the sphere.

$$\begin{aligned} f(r) &= \begin{cases} 6/(\pi a^2)\sqrt{1-(2r/a)^2}, & \text{if } r \le a/2 \\ 0, & \text{otherwise} \end{cases} \\ F(\rho) &= 3\sqrt{\pi/2}J_{3/2}(\pi a\rho)(\pi a\rho)^{-3/2} \\ &= \frac{3}{(\pi a\rho)^3}\left[\sin(\pi a\rho) - \pi a\rho\cos(\pi a\rho)\right] \\ &= 1 - \frac{z^2}{10} + \frac{z^4}{280} - \ldots \end{aligned}$$

where $z = \pi a\rho$.

4. Ring a = diameter; the brightness is zero except on the circumference.

$$\begin{aligned} f(r) &= \frac{1}{\pi a}\delta(r - a/2), \\ F(\rho) &= J_0(\pi a\rho), \\ &= 1 - \frac{z^2}{4} + \frac{z^4}{64} - \ldots \end{aligned}$$

where $z = \pi a\rho$.

5. Uniformly bright square a = edge length.

$$\begin{aligned} f(x,y) &= \begin{cases} 1/a^2, & \text{if } |x| \le a/2,\ |y| \le a/2 \\ 0, & \text{otherwise} \end{cases}, \\ F(u,v) &= \frac{\sin(\pi au)}{\pi au}\frac{\sin(\pi av)}{\pi av}. \end{aligned}$$

Comparison of standard components It is difficult to distinguish between a gaussian, a disk, an optically thin sphere, and a ring from short-baseline visibility measurements. All have a quadratic dependence on baseline. If the respective 'diameters' are a_G, a_D, a_S, a_R, then the series expansions given above show that the short-baseline visibilities will be equal if

$$\frac{a_G^2}{4\ln 2} = \frac{a_D^2}{8} = \frac{a_S^2}{10} = \frac{a_R^2}{4}$$

i.e.,

$$a_D = 1.70 a_G$$
$$a_S = 1.90 a_G$$
$$a_R = 1.20 a_G.$$

It is also useful to compare the baselines at which the component visibility drops to 0.5. These are:

$$\pi a_G \rho = 1.386 \quad \text{(gaussian)}$$
$$\pi a_D \rho = 2.215 \quad \text{(uniform disk)}$$
$$\pi a_S \rho = 2.498 \quad \text{(optically thin sphere)}$$
$$\pi a_R \rho = 1.521 \quad \text{(ring)}$$

The 0.5 visibility point will occur at the same baseline if

$$a_D = 1.60 a_G$$
$$a_S = 1.80 a_G$$
$$a_R = 1.10 a_G.$$

References:

Purcell, G. H., Jr., *The structure of compact radio sources at 609 MHz*, Ph.D. thesis, California Institute of Technology (1973).

Bracewell, R., *The Fourier transform and its applications*, New York: McGraw-Hill (1965).

7.3 Model computation in the sky plane

Calculation of a model in the sky plane (*i.e.*, a model image) is more difficult than calculating its Fourier transform. There are two problems to worry about: (*a*) what is usually required is a model convolved with a beam, normally a gaussian; (*b*) each pixel in an image represents the total flux in a small region of the image. A completely analytical calculation involves (*a*) analytical convolution of each model component $f_i(x, y)$ with the beam, and (*b*) integration of the resulting function over the area of each pixel.

The first step, analytical convolution, is easy for a gaussian beam and delta-function or gaussian model components (see next section). I don't know any analytical result for the other component types, however. It is also questionable whether it is appropriate to use a gaussian (which extends to infinity) for the beam. The second step, integration, is usually approximated by evaluating the convolved model at the center of each pixel. This requires high oversampling (many pixels per beamwidth) for accurate results.

If numerical convolution is used, it is necessary to sample the model function on a grid before convolution. Unless a very fine grid is used (again, many pixels per beam), there will be aliasing problems ('jaggies') such as occur in computer graphics.

7.4 Convolution of elliptical gaussians

Programs such as MODPLOT need to convolve elliptical gaussian model components with an elliptical gaussian beam pattern. Here we state the formulas used for this computation; they can also be applied to the problem of deconvolution.

If F_i is a two-dimensional elliptical gaussian with major axis a_i, minor axis b_i, and major-axis position angle ϕ_i, then the convolution of n such gaussians,

$$F_0 = F_1 \star F_2 \star \cdots \star F_n,$$

is another gaussian with parameters a_0, b_0, ϕ_0 which can be computed as follows. Let

$$s_i = {a_i}^2 + {b_i}^2,$$
$$d_i = {a_i}^2 - {b_i}^2,$$

so that

$$a_i = \sqrt{(s_i + d_i)/2},$$
$$b_i = \sqrt{(s_i - d_i)/2}.$$

Then

$$s_0 = \sum_{i=1}^{n} s_i,$$
$$d_0 \sin 2\phi_0 = \sum_{i=1}^{n} d_i \sin 2\phi_i,$$
$$d_0 \cos 2\phi_0 = \sum_{i=1}^{n} d_i \cos 2\phi_i.$$

Reference: J. P. Wild, *Aust. J. Phys.*, **23**, 113–115 (1970).

Convolution of other model components (*e.g.*, a disk or ring) with an elliptical gaussian is more difficult, and is usually done numerically (*e.g.*, R. Padman *et al.*, *Mon. Not. R. astr. Soc.*, **214**, 251, 1985).

Chapter 8

FITS FILES

8.1 Introduction

The Caltech VLBI programs use the standard "FITS" formats for images (maps). The (u, v) data files ("MERGE" files) can be converted into the standard "UVFITS" format for transport to other computer systems, such as AIPS.

FITS is a widely-used format for the interchange of astronomical images and other data on magnetic tape. Binary data in 16-bit or 32-bit integer "pixels" are accompanied by a human-readable "header" containing auxiliary parameters. Although the general FITS format is standardized, there is considerable flexibility in the way data may be arranged within the file and in the choice of header parameters. To maximize portability, the Caltech VLBI programs try to be consistent with AIPS usage.

Strictly, the FITS standard applies only to files on 9-track magnetic tape. The Caltech VLBI programs operate on FITS disk files, which are direct byte-for-byte copies of standard FITS tape files. These files can also be read and written by AIPS programs. In UNIX, a FITS disk file is a continuous byte stream. In VAX/VMS, the bytes are arranged in fixed-length 2880-byte records in the same way as on tape.

8.2 FITS images (maps)

The following Caltech VLBI programs produce FITS images:

1. CLEAN: the output "clean map" is a standard FITS image, with the list of clean components attached as an extension table, compatible with AIPS.

2. MODMAP: MODMAP convolves a model (consisting of delta functions and/or elliptical Gaussians) with a Gaussian beam, creating an image in FITS format.

3. VLBMEM: creates dirty map, dirty beam, and maximum-entropy image in FITS format.

At present, INVERT creates images which are not in the FITS format (the dirty beam and dirty map).

FITS images can be examined with programs MAPPLOT (contour or grey-scale display) and FITSLIST (listing of header).

8.3 UVFITS data files

The following programs can be used to convert visibility data in one of the Caltech formats into UVFITS format compatible with AIPS.

1. MERGEFITS: create a UVFITS file from an input MERGE-format file.

2. B2FITS: create a UVFITS file from one or more input PCR files (output files from the Block-II correlator).

3. PHFITS: create a UVFITS file from one or more data files created by GPHASOR (i.e., unfitted data from the Block-0 correlator) or PHASOR (fitted data).

To convert UVFITS files to MERGE format, use program FITSMERGE. The file headers of UVFITS files can be examined with program FITSLIST.

8.4 Use of FITS files with AIPS

The usual method of transporting data from the Caltech VLBI programs to AIPS is to write a FITS tape containing either images or uv data. The FITS tape can be read using the AIPS tasks IMLOD (for images) and UVLOD (for uv data), and it can also be examined with task PRTTP.

FITS disk files can be read by task IMLOD (for images) and UVLOD (for uv data): specify the adverb INFILE to point to the file. Note that the name of the input file must be all upper case; AIPS cannot cope with lower-case file names.

AIPS can also create FITS disk files: use task FITTP, specifying OUTFILE.

8.5 Copying FITS files from tape to disk (VMS)

First, mount the tape with the FOREIGN qualifier, specifying a block size of 28800 bytes and a record size of 2880 bytes:

```
$ MOUNT/FOREIGN/BLOCK_SIZE=28800/RECORD_SIZE=2880  MTA0:
```

(This allows you to read tapes with a blocking factor of up to 10.)

Second, position the tape at the start of the desired file; e.g., to read the third file from the tape, skip two files:

```
$ SET MAGTAPE/SKIP=FILES:2  MTA0:
```

(You can skip forward or backward, or rewind the tape: see HELP SET MAGTAPE).

Third, copy the next file on tape to disk, specifying the disk file name you want to use:

```
$ COPY  MTA0:  FILE17.FITS
```

The COPY command leaves the tape positioned at the start of the next file.

Finally, when you have copied all the files you want, dismount the tape:

```
$ DISMOUNT MTA0:
```

8.6 Copying FITS files from tape to disk (Convex)

First, allocate a tape drive, specifying `+r` to make sure that the tape is not rewound after each operation:

```
tpmount +r -d 6250
```

Second, position the tape at the start of the desired file, using the `mt` command (see manual, mt(1)); e.g., to read the third file from the tape, skip two files:

```
mt fsf 2
```

Third, copy the next file on tape to disk, specifying the disk file name you want to use:

```
cp /dev/rmt17 file17.fits
```

(Specify the tape drive allocated to you in place of `/dev/rmt17`.) The `cp` command leaves the tape positioned at the start of the next file.

Finally, when you have copied all the files you want, dismount and deallcate the tape:

```
tpunmount
```

8.7 Copying FITS files from disk to tape (VMS)

First, mount the tape with the FOREIGN qualifier, specifying a block size and a record size of 2880 bytes, and a density of 1600 bpi (you can use 6250 bpi instead, but such tapes are not "FITS standard"):

```
$ MOUNT/FOREIGN/BLOCK_SIZE=2880/RECORD_SIZE=2880/DENSITY=1600  MTA0:
```

To make a *blocked* FITS tape, specify a blocksize that is a multiple of 2880, e.g.,

```
$ MOUNT/FOREIGN/BLOCK_SIZE=28800/RECORD_SIZE=2880  MTA0:
```

Second, advance the tape to the position you want to start writing. Usually this will be either the beginning:

```
$ SET MAGTAPE/REWIND  MTA0:
```

or after the last file recorded on the tape:

```
$ SET MAGTAPE/SKIP=END  MTA0:
```

(Be very careful to avoid overwriting any files you want to keep: make sure you understand the SET MAGTAPE command.)

Third, copy the disk file to tape, specifying the disk file name you want to use:

```
$ COPY  FILE17.FITS  MTA0:
```

The COPY command leaves the tape positioned correctly to write another file if you wish.

Finally, when you have copied all the files you want, dismount the tape:

```
$ DISMOUNT MTA0:
```

8.8 Copying FITS files from disk to tape (Unix)

The Unix utilities `mt` and `dd` can be used to manipulate FITS tapes. Read the man files for these utilities and make sure you understand what you are doing before proceeding. The following example is for the Convex and uses the Convex-specific utility `tpmount`. First allocate a *non-rewinding* tape drive:

```
tpmount +r -d 6250
```

Use `mt` to position the tape where you want to start writing, e.g.,

```
mt rew
```

Then use `dd` to copy the FITS files one by one. Specify block size 2880 for standard FITS files, or a multiple of 2880 (up to 20 times, e.g., 28800) if you want to create a "blocked" FITS tape:

```
dd of=tape bs=2880 if=K24_3C380_A.FITS
dd of=tape bs=2880 if=K24_3C380_B.FITS
```

Finally use `mt` again to rewind the tape if you want to check it with MTANAL or FITSLIST, or use `tpunmount` to dismount the tape.

8.9 Checking the contents of a FITS tape

Use the FITSLIST program to read the headers of all the files on a tape (you can also use other programs, e.g., the AIPS task PRTTP). In VMS:

```
$ ALLOCATE MTA0:
$ FITSLIST
  tape = mta0:
  list = fitslist.lis
  /
$ DISMOUNT MTA0:
$ DEALLOCATE MTA0:
$ TYPE FITSLIST.LIS
```

In UNIX (interactive):

```
tpmount -a /dev/rmt17
fitslist
  tape = /dev/rmt17
  list = fitslist.lis
  /
tpunmount
more fitslist.lis
```

FITSLIST lists all the headers of all the files on the tape, including extension files. For a briefer listing, PRTTP (AIPS) is recommended. For a very brief listing, use the INDEX parameter in FITSLIST.

FITSLIST can also be used to list the headers of one or more disk FITS files. In VMS:

```
$ FITSLIST
  input = citscr:[tjp]*.fits
  list = fitslist.lis
  /
$ TYPE FITSLIST.LIS
```

In UNIX:

```
fitslist
  input = /scr/tjp/*.fits
  list = fitslist.lis
  /
cat fitslist.lis
```

8.10 References

The FITS image format is defined in "FITS: a flexible image transport system" by D. C. Wells, E. W. Greisen and R. H. Harten, *Astron. Astrophys. Suppl.*, **44**, pp. 363–370 (1981).

The FITS format for visibility data is defined in "An extension of FITS for groups of small arrays of data" by E. W. Greisen and R. H. Harten, *Astron. Astrophys. Suppl.*, **44**, pp. 371–374 (1981).

Further extensions to the FITS standard are defined in "Generalized extensions and blocking factors for FITS" by P. Grosbøl, R. H. Harten, E. W. Greisen, and D. C. Wells, *Astron. Astrophys. Suppl.*, **73**, pp. 359–364 (1988); and "The FITS tables extension" by R. H. Harten, P. Grosbøl, E. W. Greisen, and D. C. Wells, *Astron. Astrophys. Suppl.*, **73**, pp. 365–372 (1988).

Chapter 9

KEYIN

This chapter defines the syntax of the "free-format parameter input" used by the Caltech VLBI programs. This is implemented in a Fortran subroutine called KEYIN. It provides a mechanism for a user to assign values to program parameters by name. A basic assignment is

```
PARAMETER = VALUE
```

The assignments can occur in any order in the input stream, with zero, one or more assignments per line. A single assignment may not extend over more than one line. The parameter name is separated from its value by one or more blank-spaces and/or an equals sign (=). The *value* part is optional: some parameters do not require values and may be regarded as switches, which if present turn on a particular option of the program. A tab-character may be used wherever a blank is allowed.

9.1 Parameters

The parameter names are determined by the particular program; typically a name will consist of between 1 and 8 letters or digits, starting with a letter. Only the first 8 characters are significant; characters after the 8th are ignored. Parameters can be typed in either upper or lower case, and can be abbreviated (minimum match).

9.2 Values

Numerical values A numerical value consists of the following parts, any of which may be omitted: *sign, integer-part, fractional-part, scale-factor.*

- *sign* is + or −, and may be separated from the rest of the value by one or more blanks. Blanks are not allowed anywhere else within the value.
- *integer-part* is 0 or more decimal digits.
- *fractional-part* is a decimal-point (.) followed by 0 or more decimal digits.
- *scale-factor* is the letter E or e followed by an optional + or − sign and one or more digits, representing a power of 10. A value may not consist only of an scale-factor.

  Examples of acceptable values:

```
0  1  -13  13.35  0.1335E+2  1.3351e1
```

Time and angle values One or more colons (:) are allowed in the integer-part of a numerical value. The effect of a colon is to multiply the preceding part by 60 and add it to the following part. This allows the notation `hh:mm:ss` or `ddd:mm:ss`. For example,

```
04:13 = 4*60 + 13 = 253
12:00:05 = (12*60 + 0)*60 + 5 = 43205
```

Fractional-parts are allowed, e.g.,

```
RA = 23:13:42.6378   DEC = -32:14:15.1
```

Note: If you include only one colon when the program expects hh:mm:ss, it will be treated as mm:ss. This is different from the AIPS convention, which would treat the number as hh:mm. The program CAL is anomalous in this respect: it expects times to be entered as hh:mm, and you will get the wrong result if you include a seconds field.

Textual values A parameter may have a textual value (any sequence of characters), indicated by quotation marks, e.g.,

```
NAME="Flann O'Brien"
TELESCOPE='JODRELL BANK 250-FT'
```

The text can be enclosed in either double-quotes or apostrophes. The first example shows how an apostrophe can be included in a double-quoted string; similarly a double-quote character can be included in an apostrophized string. To simplify typing, the quotation marks may be omitted, but only if (*a*) the equals sign (=) is not omitted, and (*b*) the character string does not include any blank spaces or commas and does not start with a digit.

Multiple values Some parameters may be allowed to have more than one value, e.g.,

```
OPTS = 1,2,7,9,15
```

A comma after a value indicates that another value follows; do not put a comma after the last value. An error message will be printed if too many values are assigned to a parameter. Values may also be assigned to a multi-valued parameter by an "array" notation, e.g.,

```
OPTS(1)=1   OPTS(4)=9   OPTS(2)=2,7
```

(The effect of the last example is to assign OPTS(2) = 2 and OPTS(3) = 7.) The "subscript" consists of an unsigned integer enclosed in parentheses, with no spaces either before or within it. Any of the different types of value can be used in a multiple assignment, i.e., the values need not be all numeric or all textual. When typing long lists of values, you can start a new line after a comma, e.g.,

```
PLEV = -8, -4, -2, -1, -0.5, 0, 0.5, 1, 2,
       4, 8, 16, 32, 64
STATIONS = BONN, WSRT, BOLOGNA, JODRELL, ONSALA, HAYSTACK,
       NRAO-GB,FDVS, VLA-27, PIETOWN, KITT-PEAK, OVRO, HAT-CREEK
```

Arithmetic expressions An arithmetic expression can be used instead of a numerical value, e.g.,

```
X = (56+13.6/2-(15.6+3))
XYINT = (1/3)
```

Arithmetic expressions are composed of numbers (numerical values, as defined above) and the operators + - * / ( ) and must begin with (. Note that the entire expression must be enclosed in parentheses. Blanks are not permitted within the expression. The normal rules of algebra apply, except that / and * have equal precedence, so that for example $(1/5 * 6) = 1.2$.

9.3 Comments

Any characters following an exclamation mark (!) up to the end of the line are ignored. These comments can be used to annotate a set of parameters.

9.4 End marker

Some programs will reserve a special word to mark the end of the free-format input. Usually a solidus (/) is used.

9.5 Special commands

Certain words are recognized as special commands. Each of these should be typed on a line by itself:

- `HELP` will type out the help file for the current program, if one exists [VMS version only].
- `HELP PROG` will type out the help file for the program called PROG, if one exists in the VLB help library [VMS version only].
- `SHOW` will print a list of the parameters and their current values.
- `SHOW PARM` will show the current value of parameter PARM.
- `SAVE file` will save the current values of the parameters in a file (see below).
- `@ file` will take parameter values from a file (see below).

9.6 File inclusion and parameter saving

To save typing, a program can be told to take input from a named file by use of the `@` symbol, eg

```
@/scr/tjp/X.DAT
```

would cause the program to read and interpret the file X.DAT. After reading the file, the program reverts to the previous input stream (unless the included file contains the end marker). One or more spaces may occur between the `@`-sign and the file-name. A command of this type should be the last item on a line.

The inverse operation to `@` is saving the current values of all the parameters used by the program in a file. This is done with the special command SAVE, e.g.,

```
SAVE /scr/tjp/X.DAT
```

would create a file X.DAT and write the names of all the program parameters and their values in this file in a format that can be recovered by the `@` command. Thus SAVE can be used in one run of a program and then `@` can be used in a later run to get back the parameters of the first run.

If either `@` or SAVE is used without a file name, a file name of the form *program*`.par` is used, where *program* is the name of the program being executed. Note that SAVE overwrites the file; it doesn't create a new version.

DCL symbol substitution

The VMS version of KEYIN provides a mechanism for substituting DCL-defined symbols in the input stream. This is sometimes useful in comlicared or iterative command procedures. As KEYIN reads each input line, it looks for constructions of the form {SYMBOL}, where SYMBOL is a defined DCL symbol name. If it finds such strings, it substitutes the value of the DCL symbol for {SYMBOL}.

Example:

```
$ INPUT = "CIT:[TJP.JUNK]SLUDGE.MRG"
$ PLOT  = "/VERS"
$ LINEWIDTH = 3
$ MORE = "FONT=3"
$ UVDIST
PLOTFILE = "{PLOT}" INPUT = "{INPUT}"
LINEWIDTH = {LINEWIDTH}
{MORE}
/
$ EXIT
```

The input stream in this example is equivalent to

```
PLOTFILE = "/VERS" INPUT = "CIT:[TJP.JUNK]SLUDGE.MRG"
LINEWIDTH = 3
FONT=3
/
```

Take note of the quotation marks. This is mostly useful in iterative procedures like hybrid mapping; it allows one to use the conditional statements of DCL to select different parameters for different iterations. An error occurs if the string within the braces is not a defined DCL symbol.

The same mechanism is not required in Unix, because the command interpreter `csh` provides it, albeit with a different syntax. The equivalent `csh` procedure would be:

```
set input = /mnt/tjp/junk/sludge.mrg
set plot  = /VERS
set linewidth = 3
set more = "font=3"
uvdist
  plotfile = "${plot}"
  input = "${input}"
  linewidth = ${linewidth}
  ${more}
  /
exit
```

Chapter 10

MOVING FILES BETWEEN VMS AND UNIX

Data files come in three types: (1) Text files: models, catalogs, programs, etc; (2) MERGE files; (3) FITS files (images and uvfits). Text files are easy to transport, and require no translation. They can be transferred using DECnet (COPY and netcp), TCP/IP (ftp), or tape (BACKUP or tar). MERGE files require internal format changes when transferred between VMS and UNIX, or when transferred between unlike UNIX machines.

10.1 Note on using netcp

netcp is a Convex program for performing DECnet copies. Before using `netcp` for the first time, define the user name you want to use on the remote (VMS) node in a `setenv` command:

```
setenv NET_USER TJP
```

netcp will prompt for you to supply the password associated with this account on the remote node. The `setenv` command can be put in your `.login` file.

10.2 Transferring TEXT files from VMS to UNIX

Text files include command files, scripts, and model files. They can be transferred directly using ftp (TCP/IP protocol) or netcp (DECnet protocol), and require no additional translation. Example (using netcp):

```
netcp -ln 'DEIMOS::CITSCR:[TJP]1823.MD4' 1823.md4
```

Example (using ftp):

```
ftp deimos
<prompts for username and password>
get CITSCR:[TJP]1823.MD4 1823.md4
bye
```

10.3 Transferring TEXT files from UNIX to VMS

Use netcp or ftp, e.g.,

```
netcp -l  3C273.model 'PHOBOS::CITSCR:[TJP]3C273.MODEL'
```

The copy can also be initiated from the VMS end (note that this cannot be used for binary files):

```
$ COPY/LOG IKAROS"username password"::"/mnt/tjp/3C273.model" -
CITSCR:[TJP]3C273.MODEL
```

10.4 Transferring FITS files from VMS to UNIX

FITS files are transferred in the same way as text files, but specifying verbatim mode (in netcp) or binary mode (in ftp). Example (using `netcp`: note the `v` flag!):

```
netcp -lv 'DEIMOS::CIT:[VLB]SUSIE.MAP' susie.map
```

Example (using ftp):

```
ftp deimos
<prompts for username and password>
binary
get CIT:[VLB]SUSIE.MAP susie.map
bye
```

10.5 Transferring FITS files from UNIX to VMS

On the UNIX machine, copy the file to the VMS machine using `netcp` in verbatim mode or `ftp` in binary mode:

```
netcp -lv 3C273.map 'DEIMOS::SCR:[TJP]3C273.MAP'
```

```
ftp deimos
<prompts for username and password>
binary
put 3C273.map  SCR:[TJP]3C273.MAP
bye
```

`ftp` can also be used from the VMS machine with a `get` command. After copying the file, it is necessary to *fix up the VMS file header*. On the VMS machine, run program `FILESET` to set the attributes of the file (this doesn't change the file itself, but tells VMS that the file has variable-length records instead of fixed length):

```
$ SET COMMAND VLB:FILESET
$ FILESET SCR:[TJP]3C273.MAP -
  /ATTRIB=(TYPE=FIXED,RECORD=2880,MAX_RECORD=2880)
```

Check that FITSLIST or MAPPLOT can read the resulting file.

10.6 Transferring MERGE files from VMS to Convex

MERGE files are transferred like FITS files in verbatim or binary mode; but after transferring the file, you must use program `convax` to translate the data into Convex format. `convax` asks you whether you want to translate from VMS to Convex or vice versa, and then asks for the input and output file names. Note that this is a *Convex-specific* procedure, and no equivalent exists for transferring merge files between VMS and Sun.

10.7 Transferring MERGE files from Convex to VMS

This is a tedious operation which requires all of the following steps to be performed correctly. Suppose the original Convex file is called `3C273.mrg` and we want the VMS file to be called `3C273.MRG` in the scratch directory.

1. On the Convex, use program `convax` to convert from Convex format to VMS format:

```
convax
1
3C273.VMS
3C273.mrg
```

Specify the direction (Convex to VMS) correctly and take care not to mix up the input and output file (if you, do, you will destroy the input file; this bug should be fixed!).

2. On the Convex, copy the converted file to the VMS machine using netcp in verbatim mode or ftp in binary mode:

```
netcp -lv 3C273.VMS 'PHOBOS::CITSCR:[TJP]3C273.MRG'
```

3. On the VMS machine, run program `FILESET` to set the attributes of the file (this doesn't change the file itself, but tells VMS that the file has variable-length records instead of fixed length):

```
$ SET COMMAND VLB:FILESET
$ FILESET CITSCR:[TJP]3C273.MRG -
  /ATTRIB=(TYPE=VARIABLE,MAX_RECORD=32000)
```

4. Check that LIST or VISPLOT can read the resulting file.

Chapter 11

FORMAT OF MERGE FILES

This chapter describes the format used for Merge files, and is intended for programmers. The program MRGCOPY is a template program for reading and operating on a Merge file; the program should read in conjunction with the format description.

The MERGE-format data files are binary files and must be read or written using Fortran unformatted I/O: i.e., READ(9) or WRITE(9). This results in a VBS-format file on the IBM 370 or a Sequential Segmented file on VAX/VMS.

Each file consists of a number of *header* records, followed by *data* records.

The following notation is used:

I4: 4-byte integer (INTEGER*4)
R4: 4-byte floating-point (REAL*4)
R8: 8-byte floating-point (REAL*8)
Cn: n characters (CHARACTER*n)

(Note that the binary format of these quantities is different on different computers; e.g., IBM characters are EBCDIC, VAX characters are ASCII. A translation program such as `convax` must be used when moving Merge files from one type of computer to another.)

11.1 Header Records

Book-keeping (20 I4 words)

1. Format version number: this must be either 1 or 2. All the VLB programs recognize format 1, but some may not recognize format 2. In format 2, the (u, v)-coordinates of each data point are stored in the file, while in format 1 the (u, v)-coordinates must be calculated from the information in the header.
2. Number of "history" records following.
3. Total number of header records (excluding this record).
4. Number of stations (antennas), N_s. The maximum number of stations is 99, but few programs can cope with more than 20.
5. Number of baselines, N_b; $N_b \leq N_s(N_s - 1)/2$.
6. Number of closure-phases, N_t; $N_t \leq N_s(N_s - 1)(N_s - 2)/6$; $N_t = 0$ if there are no closure-phases.
7. Length of data records in 4-byte words, L: $L = 1 + 4N_b + 2N_t$ (format 1); $L = 1 + 7N_b + 2N_t$ (format 2).
8. Type of data: 0 = uncorrected correlation coefficient; 1 = correlated flux density (Jy).
9. Polarization (Stokes parameter): this follows the AIPS convention: $-1 = RR$, $-2 = LL$, $-3 = RL$, $-4 = LR$, $1 = I$, $2 = Q$, $3 = U$, $4 = V$; 0 = undefined.

The remaining words in this record are unused, and should be set to zero.

Baseline Index

An indication of which baselines are included in the file: N_b I4 words, one for each baseline, equal to $100I_1+I_2$ where I_1 and I_2 are the numbers of the two stations comprising the baseline ($I_1 < I_2$). This record is absent if $N_b = 0$. If all the baselines are present in the "canonical" order, this record will contain 102, 103, 203, 104, 204, 304, 105, 205, 305, 405, etc. Any subset of the baselines is allowed, in any order.

Closure-triangle Index

An indication of which closure-phases are included in the file: N_t I4 words, e.g., $10000I_1 + 100I_2 + I_3$ for the triangle formed by stations I_1, I_2, I_3 ($I_1 < I_2 < I_3$). This record is absent if $N_t = 0$. If all the possible triangles are present in the "canonical" order, this record will contain 10203, 10204, 10304, 20304, 10205, 10305, 20305, 10405, 20405, 30405, etc.

History records

The number of history records is given in the first header record. Each record has format C80 (80 characters). Their contents are entirely arbitrary. They are intended to provide a readable record of the source of the data and its analysis. Normally one or more history records are added each time the file is modified. There is no limit on the number of history records which can appear in a file.

Source parameters (format C16 + 8R8)

1. Source name (C16, 16 characters).
2. Mean RA of 1950.0 (R8, radians).
3. Mean Dec of 1950.0 (R8, radians).
4. Apparent RA of date of observation (R8, radians).
5. Apparent Dec of date of observation (R8, radians).
6. Total flux density, Jy (R8).

The remaining words in this record are not used, and should be set to zero.

Station Parameters (N_s records)

In MERGE format 1, the station records are as follows (each C8 + 9R8):

1. Station name (C8, 8 characters).
2. X (R8, meters).
3. Y (R8, meters).
4. Y (R8, meters).

The remaining words in this record are unused, and should be set to zero. X, Y, Z form a left-handed geocentric coordinate system with Z directed towards the North Pole and X directed towards the Greenwich Meridian. The origin of coordinates is usually the center of the earth, but this is arbitrary.

In MERGE format 2, the station records are interpreted differently (each C8 + 9R8 + I4):

1. Station name (C8, 8 characters).

2–10. Nine R8 words (see below).

11. Station type (I4), = 1 for ground stations, =2 for orbiters.

For ground stations, words 2–4 are X, Y, Z as in format 1; words 5–10 are zero. For orbiters, words 2–7 have the following meaning, while words 8–10 are zero:

2. A, orbit semi-major axis (R8, meters).
3. e, orbit eccentricity (R8, unitless).
4. i, inclination of orbital plane to X',Y' plane (R8, radians).
5. Ω, longitude of orbit ascending node measured eastward from X' axis (R8, radians).
6. ω, argument (angle) of periapsis measured in orbital plane, in direction of orbit, from ascending node (R8, radians).
7. M, mean anomaly (mean angle from periapsis) of spacecraft (R8, radians).

X', Y', Z' form the *right-handed, inertial* coordinate system in which the orbital elements are measured. X' points toward the position of the vernal equinox at $0^{\rm h}$ UT on the start date of the experiment and Z' points toward the celestial north pole. Note that this is also the frame to which source coordinates are precessed and in which baselines are computed. The epoch of the ascending node, periapsis, and mean anomaly is $0^{\rm h}$ UT on the experiment start date.

Miscellaneous (2I4 + 9R8)

1. Year, e.g., 1978 (I4).
2. UT of some point during the run, e.g., $0^{\rm h}$ on the first day of the run; see below (I4).
3. Greenwich [Apparent] Sidereal Time at the specified UT, radians (R8).
4. Frequency of observation, Hz (R8).
5. Coherent averaging time, seconds (R8).
6. Incoherent averaging time, seconds (R8) (zero if not averaged).
7. Bandwidth, Hz (R8) (zero if unknown).

The remaining words in this record are unused, and should be set to zero.

The UT in this record and in the data records is an integer counting the number of sixtieths of seconds elapsed since the beginning of the year (range 0–1 897 344 000).

11.2 Data Records

Each data record contains L words of which the first is I4 and the remainder are R4. The first word is the UT at the center of the integration interval (integer number of sixtieths of seconds since the beginning of the year).

In format 1 the UT is followed by N_b groups of 4 (R4) words:

1. Amplitude (correlation coefficient or correlated flux density in Jy),
2. Error in amplitude (standard error),
3. Phase (degrees),
4. Error in phase.

In format 2 the UT is followed by N_b groups of 7 (R4) words:

1. Amplitude (correlation coefficient or correlated flux density in Jy),
2. Error in amplitude (standard error),
3. Phase (degrees),
4. Error in phase,
5. East-west baseline projection (u) in wavelengths,
6. North-south baseline projection (v) in wavelengths,

7. Baseline projection toward source (w) in wavelengths.

In both formats, the N_b groups are followed by N_t groups of 2 (R4) words:

1. Closure-phase (degrees),
2. Error in closure-phase.

Absence of data is indicated by a zero or negative error. Within each data record, the N_b baselines are arranged in the order specified in the "baseline index" header record and the N_t closure-phases are arranged in the order specified in the "closure-triangle index" header record. The data records can be in any order, but should preferably be in order of UT.

End Record End of data is indicated by a record with the same format as a data record but with a negative UT. The End Record can be omitted if the last data record is followed by end-of-file. Some programs will accept concatenated MERGE files, in which the data for one file are followed by the header of another. In such files, the end record is required.