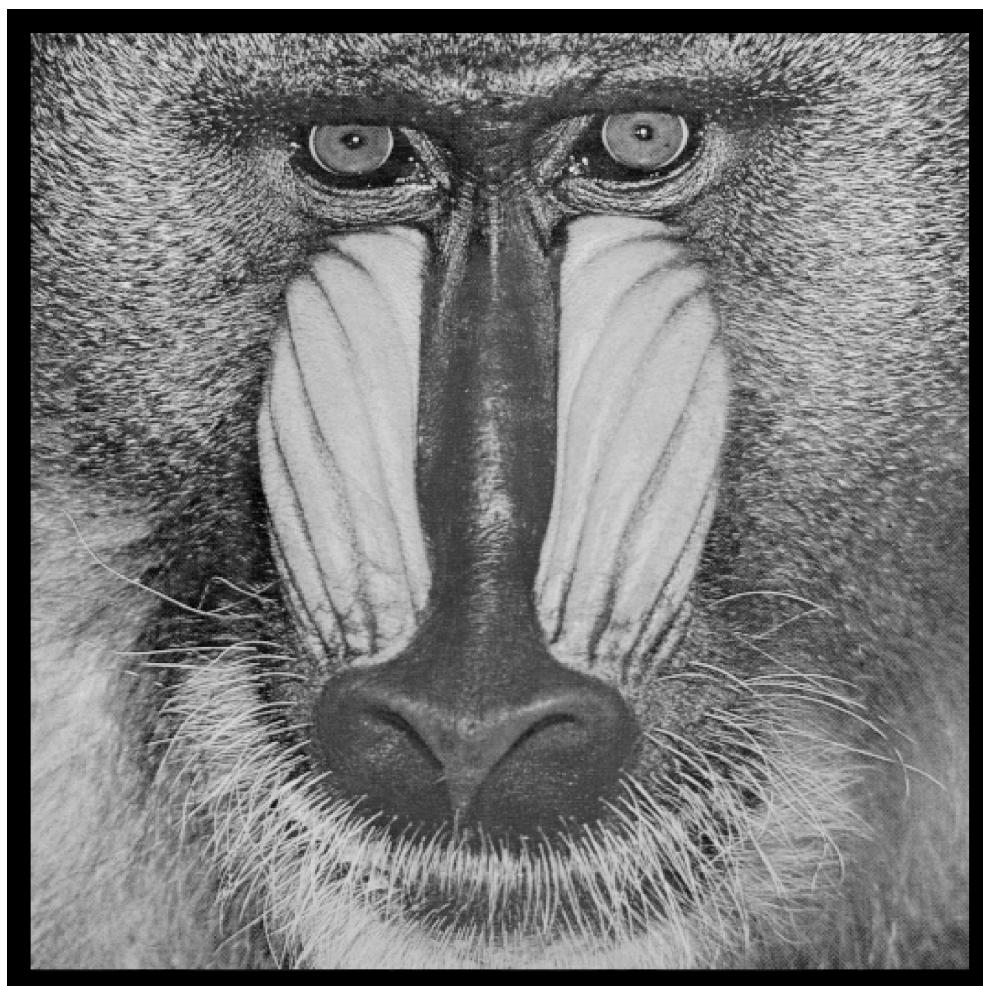


AIPS COOKBOOK

31-December-2017 and earlier



The National Radio Astronomy Observatory

Edgemont Road
Charlottesville, VA 22903-2475

A facility of the National Science Foundation
operated under cooperative agreement by
Associated Universities, Inc.

Acknowledgments

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

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

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

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

This *CookBook* is now available on the Internet via the World-Wide Web. The current Table of Contents together with a revision history for the full 31DEC14 *CookBook* is available at

<http://www.aips.nrao.edu/cook.html>.

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

Contents

1	Introduction	1 – 1
1.1	The NRAO <i>AIPS</i> Project — A Summary	1 – 1
1.2	The <i>CookBook</i>	1 – 3
1.3	Organization of the <i>CookBook</i>	1 – 4
1.3.1	Contents	1 – 4
1.3.2	Minimum match	1 – 5
1.3.3	Fonts and what they signify	1 – 5
1.4	General structure of <i>AIPS</i>	1 – 5
2	Starting Up <i>AIPS</i>	2 – 1
2.1	Obtaining access to an <i>AIPS</i> computer	2 – 1
2.2	Using the workstation	2 – 1
2.2.1	Logging in to the workstation	2 – 1
2.2.2	Control characters on the workstation	2 – 2
2.2.3	Starting the <i>AIPS</i> program	2 – 3
2.2.4	Typing commands to the <i>AIPS</i> program	2 – 10
2.3	Managing windows	2 – 10
2.3.1	General window management	2 – 10
2.3.2	Managing the <i>AIPS</i> TV window called XAS	2 – 11
2.4	Additional recipes	2 – 12
3	Basic <i>AIPS</i> Utilities	3 – 1
3.1	Talking to <i>AIPS</i>	3 – 1
3.1.1	<i>POPS</i> and <i>AIPS</i> utilities	3 – 1
3.1.2	Tasks	3 – 1
3.1.3	Verbs	3 – 2
3.1.4	Adverbs	3 – 2
3.2	Your <i>AIPS</i> message file	3 – 3
3.3	Your <i>AIPS</i> data catalog files	3 – 4
3.3.1	Speedy data file selection	3 – 5
3.3.2	Catalog entry status	3 – 6
3.3.3	Renaming data files	3 – 6
3.3.4	Header listings	3 – 6

Table of Contents

3.4 Your <i>AIPS</i> history files	3 – 8
3.5 Saving and restoring inputs	3 – 8
3.6 Monitoring disk space	3 – 10
3.7 Moving and compressing files	3 – 11
3.8 Finding helpful information in <i>AIPS</i>	3 – 11
3.9 Magnetic tapes	3 – 14
3.9.1 Hardware tape mount	3 – 14
3.9.2 Software mounting local tapes	3 – 15
3.9.3 Software mounting REMOTE tapes	3 – 15
3.9.4 Using tapes in <i>AIPS</i>	3 – 15
3.10 <i>AIPS</i> external disk files	3 – 16
3.10.1 Disk text files	3 – 16
3.10.2 RUN files	3 – 17
3.10.3 FITS-disk files	3 – 18
3.10.4 Other binary data disk files	3 – 19
3.11 The array processor	3 – 20
3.12 Additional recipe	3 – 20
4 Calibrating Interferometer Data	4 – 1
4.1 Copying data into <i>AIPS</i> multi-source disk files	4 – 2
4.1.1 Reading from VLA archive files using FILLM	4 – 3
4.1.1 Reading from a VLA archive files using FILLM	4 – 3
4.1.1.1 Editing and applying nominal sensitivities to VLA data	4 – 7
4.1.2 Reading data from FITS files with FITLD	4 – 7
4.2 Record keeping and data management	4 – 8
4.2.1 Calibrating data with multiple FQ entries	4 – 8
4.2.2 Recommended record keeping	4 – 8
4.3 Beginning the calibration	4 – 9
4.3.1 Initial editing	4 – 10
4.3.2 Primary flux density calibrators	4 – 11
4.3.3 First pass of the gain calibration	4 – 13
4.3.3.1 Using calibrator models	4 – 13
4.3.3.2 Flux calibration without calibrator models	4 – 14
4.4 Assessing the data quality and initial editing	4 – 16
4.4.1 Editing with LISTR and UVFLG	4 – 19
4.4.2 Editing with EDITA	4 – 20
4.4.3 Editing with TVFLG	4 – 23

4.4.4	Baseline corrections	4 – 29
4.5	Antenna-based complex gain solutions	4 – 31
4.5.1	Bootstrapping secondary flux-density calibrators	4 – 31
4.5.2	Full calibration	4 – 32
4.5.3	Final (?) initial global calibration	4 – 32
4.6	Polarization calibration	4 – 35
4.7	Spectral-line calibration	4 – 40
4.7.1	Reading the data	4 – 40
4.7.2	Spectral-line aspects of SETJY	4 – 40
4.7.3	Editing the data	4 – 41
4.7.4	Bandpass calibration	4 – 42
4.7.5	Amplitude and phase calibration	4 – 45
4.8	Solar data calibration	4 – 46
4.8.1	Reading solar data from the VLA archive	4 – 46
4.8.2	Using SNPLT and LISTR to assess the nominal sensitivities	4 – 47
4.8.3	Using SOLCL or SYSOL to apply the system-temperature correction	4 – 47
4.9	Completing the initial calibration	4 – 48
4.9.1	Using FITTP and FITAB to write multi-source data to tape	4 – 48
4.9.2	Creating single-source data files with SPLIT	4 – 49
4.9.3	Making images from multi-source data with IMAGR	4 – 50
5	Making Images from Interferometer Data	5 – 1
5.1	Preparing <i>uv</i> data for imaging	5 – 1
5.1.1	Indexing the data — PRTTP	5 – 1
5.1.2	Loading the data — FITLD and UVLOD	5 – 2
5.1.3	Sorting the data — UVSRT	5 – 3
5.2	Basic image making — IMAGR	5 – 4
5.2.1	Making a simple image	5 – 5
5.2.2	Imaging multiple fields and image coordinates	5 – 6
5.2.3	Data weighting	5 – 7
5.2.4	Cell and image size, shifting	5 – 10
5.2.5	Zero-spacing issues	5 – 10
5.3	Deconvolving images	5 – 11
5.3.1	Basic Cleaning with IMAGR	5 – 11
5.3.2	Multiple fields in IMAGR	5 – 14
5.3.3	Clean boxes and the TV in IMAGR	5 – 16
5.3.4	Experimental variations on Clean in IMAGR	5 – 19

5.3.4.1	Clean-component filtering	5 – 19
5.3.4.2	SDI modification of Clean in IMAGR	5 – 19
5.3.4.3	Multi-scale modification of Clean in IMAGR	5 – 20
5.3.4.4	Spectral-index corrections	5 – 20
5.3.5	Data correction options in IMAGR	5 – 21
5.3.5.1	Frequency-dependent primary-beam corrections	5 – 21
5.3.5.2	Frequency-dependent correction for average spectral index	5 – 21
5.3.5.3	Error in the assumed central frequency	5 – 21
5.3.5.4	Array mis-orientation effects	5 – 22
5.3.5.5	Non-coplanar effects	5 – 22
5.3.5.6	Units mismatch of residuals and Clean components	5 – 22
5.3.6	Manipulating Clean components	5 – 22
5.3.7	Image-plane deconvolution methods	5 – 24
5.4	Self-calibration	5 – 25
5.4.1	Self-calibration sequence and SCMAP or SCIMG	5 – 26
5.4.2	Self-calibration with CALIB	5 – 27
5.4.3	Considerations in setting CALIB inputs	5 – 29
5.4.4	Experimental extension of multi-field self-calibration	5 – 31
5.5	More editing of <i>uv</i> data	5 – 31
5.5.1	General remarks on, and tools for, editing	5 – 31
5.5.2	Baseline-based <i>uv</i> -data editing — EDITR	5 – 33
5.6	Imaging with OBIT	5 – 38
6	Displaying Your Data	6 – 1
6.1	Getting data into your <i>AIPS</i> catalog	6 – 1
6.1.1	IMLOD and FITLD from FITS-disk	6 – 1
6.1.2	Image coordinates	6 – 2
6.2	Printer displays of your data	6 – 3
6.2.1	Printing your visibility data	6 – 3
6.2.2	Printing your image data	6 – 4
6.2.3	Printing your table data	6 – 4
6.2.4	Printing miscellaneous information	6 – 5
6.3	Plotting your data	6 – 5
6.3.1	Plotting your visibility data	6 – 7
6.3.2	Plotting your image data	6 – 8
6.3.2.1	Contour and grey-scale plots	6 – 9
6.3.2.2	Row tracing plots	6 – 12

6.3.2.3	Miscellaneous image plots	6 – 14
6.3.3	Plotting your table data	6 – 15
6.3.4	Plotting miscellaneous information	6 – 16
6.4	Interactive TV displays of your data	6 – 16
6.4.1	Loading an image to the TV	6 – 17
6.4.2	Manipulating the TV display	6 – 18
6.4.3	Intensity and color transfer functions	6 – 19
6.4.4	Setting parameters with the TV	6 – 19
6.4.5	Reading image values from the TV	6 – 20
6.4.6	Labeling images on the TV	6 – 21
6.4.7	Comparing images on the TV	6 – 21
6.4.8	Slice files and the TV display	6 – 24
6.4.9	Other functions using the TV	6 – 24
6.4.10	Capturing the TV	6 – 25
6.5	Graphics displays of your data	6 – 26
6.5.1	Plotting data and setting values with the graphics display	6 – 27
6.5.2	Slice files and the graphics display	6 – 27
6.5.3	Data analysis with the graphics display	6 – 28
6.6	Additional recipes	6 – 28
6.7	Examples of color plotting	6 – 28
7	Analyzing Images	7 – 1
7.1	Combining two images (COMB)	7 – 1
7.1.1	Subtracting a continuum image from an image cube	7 – 1
7.1.2	Polarized intensity and position angle images	7 – 1
7.1.3	Other image combination options	7 – 2
7.1.4	Considerations in image combination	7 – 3
7.2	Combining more than two images (SUMIM , SPIXR , STACK)	7 – 3
7.3	Image statistics and flux integration	7 – 4
7.4	Blanking of images	7 – 5
7.5	Fitting of images	7 – 6
7.5.1	Centroid fits (IMCENTER) and parabolic fit to maximum (MAXFIT)	7 – 6
7.5.2	Two-dimensional Gaussian fitting (IMFIT)	7 – 7
7.5.3	Source recognition and fitting (SAD , TVSAD)	7 – 8
7.5.4	Gaussian fits to slices (SLFIT)	7 – 8
7.5.5	Spectral parameter fitting (TVSPC , XGAUS , ZEMAN , RMFIT , FARS)	7 – 10
7.6	Image analysis	7 – 12

Table of Contents

7.6.1	Geometric conversions	7 – 12
7.6.2	Mathematical operations on a single image	7 – 13
7.6.3	Primary beam correction	7 – 13
7.6.4	Changing the resolution of an image	7 – 15
7.6.5	Filtering	7 – 15
7.7	Modeling in the image and uv planes	7 – 16
7.7.1	Analysis in the uv plane	7 – 17
7.8	Additional recipes	7 – 17
8	Spectral-Line Software	8 – 1
8.1	Data preparation and assessment	8 – 1
8.2	Editing and self-calibration	8 – 4
8.3	Continuum subtraction	8 – 5
8.4	Imaging	8 – 6
8.5	Display and manipulation of data cubes	8 – 7
8.5.1	Building and dismantling data cubes	8 – 7
8.5.2	Transposing the cube	8 – 9
8.5.3	Modifying the image header	8 – 9
8.5.4	Displaying the cube	8 – 10
8.6	Analysis	8 – 14
8.7	Additional recipes	8 – 17
9	Reducing VLBI Data in <i>AIPS</i>	9 – 1
9.1	VLBI data calibration recipe	9 – 2
9.2	Pipeline for the VLBA	9 – 5
9.3	Loading, fixing and inspecting data	9 – 6
9.3.1	Loading data from the VLBA correlator	9 – 6
9.3.1.1	Running FITLD	9 – 6
9.3.1.2	Calibration transfer	9 – 10
9.3.1.3	Repairing VLBA data after FITLD	9 – 11
9.3.1.4	Sorting and indexing VLBA correlator data	9 – 11
9.3.1.5	Subarraying VLBA correlator data	9 – 11
9.3.1.6	Indexing VLBA correlator data	9 – 12
9.3.1.7	Concatenating VLBA correlator data	9 – 13
9.3.1.8	Labeling VLBA correlator polarization data	9 – 13
9.3.1.9	Ionospheric corrections	9 – 14
9.3.1.10	Corrections to the Earth Orientation Parameters	9 – 16

9.3.1.11	Preparing the OB table for SVLBI data	9 – 16
9.3.1.12	Loading the time corrections file for SVLBI data	9 – 17
9.3.2	Loading data from a MkIII/MkIV correlator	9 – 17
9.3.2.1	Running MK3IN	9 – 17
9.3.2.2	Sorting MkIII/IV data	9 – 19
9.3.2.3	Concatenating MkIII/IV data	9 – 19
9.3.2.4	Merging MkIII/IV data	9 – 20
9.3.2.5	Correcting MkIII/IV sideband phase offsets	9 – 20
9.3.2.6	Indexing MkIII/IV data	9 – 21
9.4	Tools for data examination	9 – 21
9.4.1	Textual displays	9 – 21
9.4.2	Spectral displays: POSSM	9 – 22
9.4.3	Time displays: VPLT, CLPLT, and CAPLT	9 – 23
9.4.4	EDITR	9 – 24
9.4.5	SNPLT	9 – 24
9.4.6	COHER	9 – 25
9.4.7	FRPLT	9 – 26
9.5	Calibration strategy	9 – 26
9.5.1	Incremental calibration philosophy	9 – 27
9.5.1.1	Smoothing and applying corrections in SN and CL tables	9 – 28
9.5.1.2	Running CLCAL for phase referencing observations	9 – 30
9.5.2	Processing observing log and calibration information	9 – 30
9.5.2.1	Automatic formatting of VLBA and VLBA-like log files	9 – 31
9.5.2.2	Manual formatting of log files	9 – 32
9.5.2.3	Loading calibration log information	9 – 32
9.5.2.4	Generating VLA amplitude calibration with VLAMP	9 – 33
9.5.3	Data editing	9 – 34
9.5.4	Amplitude and instrumental delay calibration	9 – 35
9.5.4.1	Parallactic angle correction	9 – 35
9.5.4.2	Digital sampler bias corrections for VLBA correlator data	9 – 36
9.5.4.3	Instrumental phase corrections	9 – 36
9.5.4.4	“Manual” instrumental phase corrections	9 – 38
9.5.4.5	Bandpass calibration	9 – 40
9.5.4.6	Continuum amplitude calibration	9 – 41
9.5.5	Spectral-line Doppler correction	9 – 43
9.5.6	Spectral-line amplitude calibration	9 – 44

9.5.7	Phase calibration	9 – 45
9.5.7.1	Special considerations: SVLBI	9 – 46
9.5.7.2	Special considerations: spectral-line	9 – 47
9.5.7.3	Special considerations: polarization	9 – 47
9.5.7.4	Special considerations: phase-referencing	9 – 47
9.5.7.5	Correcting for atmospheric delays	9 – 48
9.5.7.6	Finding multi-band delays	9 – 49
9.5.7.7	Antenna-based fringe-fitting	9 – 49
9.5.7.8	Baseline-based fringe-fitting	9 – 56
9.5.7.9	SVLBI-specific techniques	9 – 58
9.5.7.10	Spectral-line fringe-fitting	9 – 59
9.5.7.11	Polarization-specific fringe-fitting	9 – 61
9.5.7.12	R-L delay calibration	9 – 61
9.5.7.13	Feed D-term calibration	9 – 62
9.5.8	Complex Bandpass	9 – 63
9.5.9	Baseline-based errors	9 – 64
9.6	After initial calibration	9 – 65
9.6.1	Applying calibration	9 – 65
9.6.2	Time averaging	9 – 65
9.6.3	Verifying calibration	9 – 67
9.7	Self-calibration, imaging, and model-fitting	9 – 68
9.7.1	CALIB	9 – 68
9.7.2	IMAGR, SCIMAG, and SCMAP	9 – 70
9.7.3	Non-conventional methods of imaging	9 – 71
9.8	Summary of VLBI calibration tables	9 – 72
9.9	Additional recipes	9 – 73
10	Single-Dish Data in <i>AIPS</i>	10 – 1
10.1	<i>AIPS</i> format for single-dish data	10 – 1
10.1.1	On-the-fly data from the 12m	10 – 2
10.1.1.1	Listing OTF input files	10 – 2
10.1.1.2	Reading spectral-line OTF files into <i>AIPS</i>	10 – 3
10.1.1.3	Reading continuum OTF files into <i>AIPS</i>	10 – 4
10.1.2	Other input data formats	10 – 4
10.2	Single-dish data in the “uv” domain	10 – 5
10.2.1	Using PRTSD, UVPLT, and POSSM to look at your data	10 – 5
10.2.2	Using UVFLG, SPFLG, and EDITR to edit your data	10 – 7

10.2.3 Using CSCOR and SDCAL to calibrate your data	10 – 11
10.2.4 Using SDLSF and SDVEL to correct your spectral-line data	10 – 12
10.2.5 Using SDMOD and BSMOD to model your data	10 – 13
10.3 Imaging single-dish data in <i>AIPS</i>	10 – 13
10.3.1 Normal single-dish imaging	10 – 13
10.3.2 Beam-switched continuum imaging	10 – 15
10.4 Analysis and display of single-dish data	10 – 19
10.4.1 Spectral baseline removal	10 – 20
10.4.2 Using WTSUM and BSAVG to combine images	10 – 21
10.4.3 Spectral moment analysis	10 – 22
10.4.4 Source modeling and fitting	10 – 22
10.4.5 Image displays	10 – 23
10.4.6 Backing up your data	10 – 24
10.5 Combining single-dish and interferometer data	10 – 24
11 Exiting from, and Solving Problems in, <i>AIPS</i>	11 – 1
11.1 Helping the <i>AIPS</i> programmers	11 – 1
11.2 Exiting from <i>AIPS</i>	11 – 1
11.2.1 Backups	11 – 2
11.2.2 Deleting your data	11 – 2
11.2.3 Exiting	11 – 3
11.3 Solving problems in using <i>AIPS</i>	11 – 3
11.3.1 “Terminal” problems	11 – 3
11.3.2 Disk data problems	11 – 4
11.3.3 Printer problems	11 – 5
11.3.4 Tape problems	11 – 5
11.4 Additional recipe	11 – 6
12 <i>AIPS</i> for the More Sophisticated User	12 – 1
12.1 <i>AIPS</i> conventions	12 – 1
12.1.1 AIPS shortcuts	12 – 1
12.1.2 Data-file names and formats	12 – 2
12.2 Process control features of <i>AIPS</i>	12 – 3
12.2.1 RUN files	12 – 3
12.2.2 More about GO	12 – 4
12.2.3 Batch jobs	12 – 5
12.3 AIPS language	12 – 6

12.3.1 Using <i>POPS</i> outside of procedures	12 – 8
12.3.2 Procedures	12 – 9
12.3.3 Writing your own programs with <i>POPS</i>	12 – 12
12.3.3.1 Special facilities for use in procedures	12 – 13
12.4 Remote use of <i>AIPS</i>	12 – 14
12.4.1 Connections via X-Windows	12 – 14
12.4.2 Connections to a terminal	12 – 15
12.4.3 Remote data connections	12 – 16
12.4.4 File transfer connections	12 – 17
12.5 Moving data to a new computer	12 – 18
12.6 Adding your own tasks to <i>AIPS</i>	12 – 18
12.6.1 Initial choices to make	12 – 19
12.6.2 Getting started	12 – 19
12.6.3 Initial check of code and procedures	12 – 20
12.6.4 Modifying an <i>AIPS</i> task	12 – 21
12.6.5 Modifying an <i>AIPS</i> template task.	12 – 21
12.6.6 Further remarks	12 – 23
12.7 Additional recipes	12 – 23
13 Current <i>AIPS</i> Software	13 – 1
13.1 ADVERB	13 – 1
13.2 ANALYSIS	13 – 10
13.3 AP	13 – 13
13.4 ASTROMET	13 – 14
13.5 BATCH	13 – 14
13.6 CALIBRAT	13 – 15
13.7 CATALOG	13 – 20
13.8 COORDINA	13 – 23
13.9 EDITING	13 – 23
13.10 EXT-APPL	13 – 24
13.11 FITS	13 – 25
13.12 GENERAL	13 – 25
13.13 HARDCOPY	13 – 28
13.14 IMAGE-UT	13 – 28
13.15 IMAGE	13 – 29
13.16 IMAGING	13 – 28
13.17 INFORMAT	13 – 32

13.18 INTERACT	13 – 33
13.19 MODELING	13 – 35
13.20 OBSOLETE	13 – 36
13.21 ONED	13 – 36
13.22 OOP	13 – 37
13.23 OPTICAL	13 – 38
13.24 PARAFORM	13 – 38
13.25 PLOT	13 – 39
13.26 POLARIZA	13 – 42
13.27 POPS	13 – 42
13.28 PROCEDUR	13 – 45
13.29 PSEUDOVOE	13 – 47
13.30 RUN	13 – 47
13.31 SINGLEDI	13 – 48
13.32 SPECTRAL	13 – 49
13.33 TABLE	13 – 50
13.34 TAPE	13 – 51
13.35 TASK	13 – 52
13.36 TV	13 – 62
13.37 TV-APPL	13 – 64
13.38 UTILITY	13 – 66
13.39 UV	13 – 67
13.40 VERB	13 – 72
13.41 VLA	13 – 78
13.42 VLBI	13 – 79
13.43 Additional recipes	13 – 81
A Easy Continuum UV-Data Calibration and Imaging	A – 1
A.1 VLARUN	A – 1
A.2 Basic calibration	A – 2
A.3 Polarization calibration	A – 5
A.4 Backup and imaging	A – 5
A.5 Additional recipes	A – 6
B A Step-by-Step Guide to Spectral-Line Data Analysis in AIPS	B – 1
B.1 Editing and calibrating spectral-line data	B – 1
B.1.1 Loading the data	B – 1

B.1.2	Inspecting and editing the data	B – 2
B.1.3	Calibrating the data	B – 3
B.2	Making and Cleaning image cubes	B – 5
B.3	Moment analysis and rotation curve of galaxies	B – 6
B.4	Multi-frequency observations	B – 7
B.4.1	General frequency information	B – 7
B.4.2	Multi-frequency <i>uv</i> files	B – 7
B.5	Additional recipes	B – 8
C	A Step-by-Step Recipe for VLBA Data Calibration in <i>AIPS</i>	C – 1
C.1	Quick Guide	C – 1
C.2	Table Philosophy	C – 2
C.3	Data set assumed in this Appendix	C – 2
C.4	VLBA Utilities	C – 2
C.5	VLBA Pipeline	C – 3
C.6	Data Loading and Inspection	C – 4
C.7	Amplitude and Delay Calibration	C – 5
C.8	Rate and Phase Calibration	C – 8
C.9	Final Calibration Steps	C – 9
C.10	Incorporating non-VLBA antennas	C – 10
C.10.1	Loading T_{sys} and Gain Curves	C – 10
C.10.2	Pointing Flags	C – 11
C.11	Pre-EVLA VLA data	C – 11
C.11.1	Single VLA Antenna	C – 11
C.11.2	Phased VLA	C – 11
C.12	Summary for non-VLBA antennas	C – 12
C.13	Some Useful References	C – 13
C.14	Additional recipes	C – 13
D	Hints for Reducing High-Frequency VLA Data in <i>AIPS</i>	D – 1
D.1	Additional recipes	D – 7
E	Special Considerations for EVLA Data Calibration and Imaging in <i>AIPS</i>	E – 1
E.1	Getting your data into <i>AIPS</i>	E – 1
E.2	Initial calibration — VLANT, FRING	E – 4
E.3	Initial editing	E – 5
E.4	Basic calibration	E – 6
E.5	Detailed flagging	E – 9

E.6 Calibration with the SysPower table	E – 11
E.7 Polarization calibration — RLDLY, PCAL, and RLDIF	E – 12
E.8 Target source data — edit and SPLIT	E – 16
E.9 Spectral-line imaging hints	E – 17
E.10 Continuum imaging hints	E – 20
E.11 Additional recipes	E – 22
E.12 Concluding remarks, early science	E – 23
L Special Considerations for EVLA P-band Data Processing in AIPS	L – 1
L.1 P-band calibration and editing in AIPS	L – 1
L.2 P-band imaging in AIPS	L – 10
L.3 Additional Recipes	L – 12
F File Sizes	F – 1
F.1 Visibility (<i>uv</i>) data sets	F – 1
F.1.1 <i>uv</i> database sizes	F – 1
F.1.2 Compressed format for <i>uv</i> data	F – 2
F.2 Image files	F – 3
F.3 Extension files	F – 3
F.4 Storing data on tape	F – 4
F.4.1 DAT and Exabyte tapes	F – 5
F.5 Very large data sets	F – 6
F.6 Additional recipes	F – 6
Z System-Dependent AIPS Tips	Z – 1
Z.1 NRAO workstations — general information	Z – 1
Z.1.1 The “midnight” jobs	Z – 1
Z.1.2 Generating color hard copy	Z – 1
Z.1.2.1 Color printers	Z – 1
Z.1.2.2 Software to copy your screen	Z – 2
Z.1.3 Using the tape drives	Z – 2
Z.1.3.1 Mounting tapes on Exabyte and DAT drives	Z – 3
Z.1.4 Gripe, gripe, gripe,	Z – 3
Z.1.5 Solving problems at the NRAO	Z – 4
Z.1.5.1 Booting the workstations	Z – 4
Z.1.5.2 Printout fails to appear	Z – 4
Z.1.5.3 Stopping excess printout	Z – 4
Z.1.5.4 CTRL Z problems	Z – 5

Table of Contents

Z.1.5.5 “File system is full” message	Z - 5
Z.1.5.6 Tapes won’t mount	Z - 5
Z.1.5.7 I can’t use my data disk!	Z - 6
Z.2 <i>AIPS</i> at the NRAO AOC in Socorro	Z - 6
Z.2.1 Reserving public-use workstations at the AOC	Z - 7
Z.2.2 Using Linux workstations at the AOC — introduction	Z - 7
Z.2.3 Starting <i>AIPS</i>	Z - 7
Z.2.3.1 Starting <i>AIPS</i> on another machine	Z - 8
Z.2.3.2 On-line FILLM	Z - 8
Z.3 <i>AIPS</i> at the NRAO in Charlottesville and Green Bank	Z - 8
G Glossary	G - 1
I INDEX	I - 1

List of Figures

1.1 Basic AIPS flow diagram	1 – 6
4.1 3C48 at X-band	4 – 15
4.2 EDITA display	4 – 22
4.3 TVFLG display	4 – 26
5.1 Affect of ROBUST parameter on beams.	5 – 9
6.1 UVPLT displays	6 – 7
6.2 Contour and grey-scale plots of an image	6 – 11
6.3 Row plots of an image	6 – 13
6.4 Statistical plots of an image	6 – 14
6.5 Slice and table plots	6 – 15
6.6 KNTR with pseudo-coloring, contours and polarization vectors in bright and dark lines	6 – 29
6.7 KNTR shows true-color image	6 – 30
6.8 PCNTR with polarization position angle shown with color	6 – 31
6.9 PCNTR with contours colored by velocity	6 – 32
6.10 KNTR with contours colored with RGBLEVS	6 – 33
6.11 SNPLT with IFs/polarizations colored	6 – 34
7.1 Geometric and other functions on an image.	7 – 14
8.1 KNTR contours of spectral channels	8 – 11
8.2 TVCUBE display of transposed cube.	8 – 12
8.3 PLCUB display of spectra from a cube	8 – 13
8.4 Images of line-cube moments.	8 – 14
9.1 POSSM and VPLOT displays of uncalibrated VLBI data	9 – 52
9.2 Spectrum before and after calibration	9 – 56
9.3 An example of a fringe rate image	9 – 71
10.1 UVPLT displays of single-dish data.	10 – 6
10.2 POSSM display of single-dish data.	10 – 7
10.3 SPFLG display.	10 – 10
10.4 Convolving function Fourier transforms.	10 – 16
10.5 Images at various beam throw rotations.	10 – 19
10.6 Images at various beam throw corrections.	10 – 20
A.1 Visibility data before and after calibration	A – 4
D.1 VPLOT of “bad” phases	D – 2
D.2 VPLOT of “okay” phases	D – 3
E.1 Example antenna D term spectrum	E – 15
E.2 Orion hot core at K band	E – 24

List of Recipes

2.4.1	Banana daiquiri	2 – 12
2.4.2	Banana pick-me-up	2 – 12
3.12.1	Banana curried chicken	3 – 20
3.12.2	Bananes rôties	3 – 20
6.6.1	Banana mallow pie	6 – 28
6.6.2	Sopa de Plátano	6 – 28
7.8.1	Bananes rôties	7 – 17
7.8.2	Almond fudge banana cake	7 – 17
7.8.3	Coriander banana nut bread	7 – 18
7.8.4	Easy banana bread	7 – 18
7.8.5	Orange gingered bananas	7 – 18
8.7.1	Hot banana soufflé	8 – 17
8.7.2	Curried bananas	8 – 17
8.7.3	Banana crunch cake	8 – 18
8.7.4	Panecillos de Plátano	8 – 18
8.7.5	Banana-Rhubarb Crisp	8 – 18
9.9.1	Banana relish	9 – 73
9.9.2	Cranberry banana bread	9 – 73
9.9.3	Churros de Plátano	9 – 74
9.9.4	Roasted turkey quesadillas with banana	9 – 74
9.9.5	Banana Bombay salad	9 – 74
11.4.1	Banana coffeeeate	11 – 6
12.7.1	Going bananas with bananas	12 – 23
12.7.2	Banana-chocolate tea bread	12 – 24
12.7.3	Banana caramel pie	12 – 24
12.7.4	Curried shrimp	12 – 24
13.43.1	Sautéed sole tobago with bananas, pecans and lime	13 – 81
13.43.2	Banana sweet potato puff casserole	13 – 81
13.43.3	Cream of banana soup	13 – 82
13.43.5	Hawaiian banana cream pie	13 – 82
A.5.1	Banana colada	A – 6
A.5.2	Breaded chicken and bananas	A – 6
A.5.3	Banana-pineapple bread	A – 6
B.5.1	Banana breeze pie	B – 8
B.5.2	Banana cutlets	B – 8

C.14.1	Golden mousse	C – 13
C.14.2	Mexican chicken vegetable soup with bananas	C – 13
C.14.3	Orange baked bananas	C – 14
C.14.4	Delightful banana cheesecake	C – 14
C.14.5	Chicken salad with banana mayonnaise and grapes	C – 14
D.1.1	Banana bran muffins	D – 7
D.1.2	Banana-pineapple rum bread	D – 8
D.1.3	Dulce Zacatecaño	D – 8
D.1.4	Virginia's instant banana pie	D – 8
D.1.5	Chocolate chip banana bread	D – 8
E.11.1	Columbian fresh banana cake with sea foam frosting	E – 22
E.11.2	Chewy banana split dessert	E – 22
E.11.3	Banana July cocktail	E – 23
E.11.4	Banana Dream Pizza	E – 23
L.3.1	Banana mandarin cheese pie	L – 12
L.3.2	Banana poundcake	L – 12
L.3.3	Frozen Push-Ups	L – 12
F.6.1	Banana stuffing	F – 6
F.6.2	Banana nut bread	F – 6

1 Introduction

1.1 The NRAO *AIPS* Project — A Summary

The NRAO Astronomical Image Processing System (*AIPS*) is a software package for interactive (and, optionally, batch) calibration and editing of radio interferometric data and for the calibration, construction, display and analysis of astronomical images made from those data using Fourier synthesis methods. Design and development of the package began in Charlottesville, Virginia in 1978. It presently consists of over 1,561,000 lines of code, 182,000 lines of on-line documentation, and 402,000 lines of other documentation.¹ It contains over 530 distinct applications “tasks,” representing very approximately 85 man-years of effort since 1978. The *AIPS* group, now solely in Socorro, has one near full-time and one part time scientist/programmers (total 1 FTE), and a few other scientific staff with some low-level responsibility to the *AIPS* effort. The group is responsible for the code design and maintenance, for documentation aimed at users and programmers, and for exporting the code to hundreds² of non-NRAO sites that have requested copies of *AIPS*. It currently offers *AIPS* installation kits for a variety of UNIX systems (mostly Linux, Solaris, and Mac OS/X), with updates available annually.³

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

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

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

¹Counted on 8-February-2016 and omitting the GNU copyrights, PostScript and PDF files, and obsolete areas.

²The 15JAN96 release alone was shipped to 225 sites and installed on well over 800 computers. The 15APR99 release was shipped to 344 non-NRAO sites. In 2009, more than 2400 different IP addresses installed or updated copies of *AIPS*.

³The TST version, currently 31DEC16, will be available continuously. It remains under development and sites may update their copies at will. Binary versions are available for Linux (32- and 64-bit), Solaris, and Max OS/X PPC and Intel and text versions for these and other operating systems.

computer architectures and under different operating systems. The NRAO has sought this level of hardware and operating system independence in *AIPS* for two main reasons. The first is to ensure a growth path by allowing *AIPS* to exploit computer manufacturers' advances in hardware and in compiler technology relatively quickly, without major recoding. (*AIPS* was developed in ModComp and Vax/VMS environments with Floating Point Systems array processors, but was migrated to vector pipeline machines in 1985. Its portability allowed it to take prompt advantage of the new generation of vector and vector/parallel optimizing compilers offered in 1986 by manufacturers such as Convex and Alliant. It was extended in simple ways in 1992 to take full advantage of the current, highly-networked workstation environment). The second is to service the needs of NRAO users in their home institutes, where available hardware and operating systems may differ substantially from NRAO's. By doing this, the NRAO supports data reduction at its users' own locations, where they can work without the deadlines and other constraints implicit in a brief visit to an NRAO telescope site. The exportability of *AIPS* is now well exploited in the astronomical community; the package is known to have been installed at some time on a large number of different computers, and is currently in active use for astronomical research at way more than 140⁴ sites worldwide. *AIPS* has been run on Cray and Fujitsu supercomputers, on Convex and Alliant "mini-supercomputers," on the full variety of Vaxen and MicroVaxen, and on a wide range of UNIX workstations and laptops including Apollo, Data General, Hewlett Packard, IBM, MassComp, Nord, Silicon Graphics, Stellar and SUN products. It is available for use on personal computers under the public-domain Linux operating system and, since 2003, On Macintosh OS/X computers. In late 1990¹, the total computer power used for *AIPS* was the equivalent of about 6.5 Cray X-MP processors running full-time.

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

The principal users of *AIPS* are **VLA**, VLBA, and VLBI Network observers. A survey of *AIPS* sites carried out in late 1990¹ showed that 61% of all *AIPS* data processing worldwide was devoted to **VLA** data reduction. Outside the NRAO, *AIPS* is extensively used for other astronomical imaging applications, however. 56% of all *AIPS* processing done outside the U.S. involved data from instruments other than the **VLA**. The astronomical applications of *AIPS* that do not involve radio interferometry include the display and analysis of line and continuum data from large single-dish radio surveys, and the processing of image data at infrared, visible, ultraviolet and X-ray wavelengths. About 7% of all *AIPS* processing involved astronomical data at these shorter wavelengths, with 7% of the computers in the survey using *AIPS* more for such work than for radio and another 7% of the computers using *AIPS* exclusively for non-radio work.

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

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

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

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

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

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

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

1.2 The CookBook

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

AIPS software is changing and growing continually. This edition of the *CookBook* describes the 31DEC16 (aka “*AIPS* for the Ages (Aged), version 17”) release of *AIPS*. Some chapters have information only from earlier releases. When something only applies to fairly recent versions of *AIPS*, a comment to that effect is made. Features remain in later releases even if the particular comment does not say as much. There were many changes in *AIPS* software between the seventh (15JAN94) and the sixth (15JUL90) edition of the *CookBook*. The chapter on the *AIPS* calibration package for continuum, spectral-line, solar and VLBI data (Chapter 4) was revised with the assistance of Rick Perley and Alan Bridle. It now has new material for improved editing and spectral-line calibration. The list of current *AIPS* tasks (Chapter 13) has been updated and reflects the extensive improvement and expansion of *AIPS* software in the 90’s. The chapters on imaging and improving images were merged as were the chapters on interactive and hard-copy displays. These mergers reflect in part the mergers of these operations. The chapter on spectral-line imaging (Chapter 8) has been revised with the assistance of Elias Brinks. Phil Diamond, John Conway, Athol Kemball, and Ketan Desai have rewritten the chapter on VLBI calibration and imaging (Chapter 9).

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

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

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

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

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

1.3 Organization of the *CookBook*

1.3.1 Contents

Chapter 2 of the *CookBook* describes in general terms how to get started in *AIPS* — signing up, logging in, mounting tapes, etc. Appendix Z gives details of these operations specific to NRAO’s *AIPS* sites. Your local *AIPS* Manager may be able to provide a version of this appendix appropriate to your system. Chapter 3 introduces the basic *AIPS* utilities. Chapter 4 leads you through the basics of reading in and calibrating your *uv* data. Chapter 5 explains the basic operations required to make and improve images. Appendices A, B, and C provide simpler recipe-like approaches to calibration and imaging which beginning users may wish to try. Chapter 6 introduces the basic *AIPS* tools for making interactive and hard-copy displays of images and other data and Chapter 7 describes tools for analyzing them. Chapter 8 and Appendix B contain hints and further *AIPS* tools of particular interest, but not restricted, to spectral-line users and other observers who have images of more than 2 dimensions. Appendix E is designed for users of the new and still changing **EVLA**. Similarly, Chapter 9 and Appendix C are aimed primarily at users of VLB interferometers. Chapter 10 deals with single-dish data reduction with *AIPS*. Chapter 11 describes how to help the *AIPS* programmers, to backup your data, and to exit from *AIPS*. It also suggests some cures for common hang-ups and miscellaneous “disasters” which seem to afflict *AIPS* users. No such list can

be made comprehensive or sufficiently general to cover all the computer systems now running *AIPS*. You will need to consult with your local *AIPS* Manager or other users if you encounter an unlisted problem.

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

1.3.2 Minimum match

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

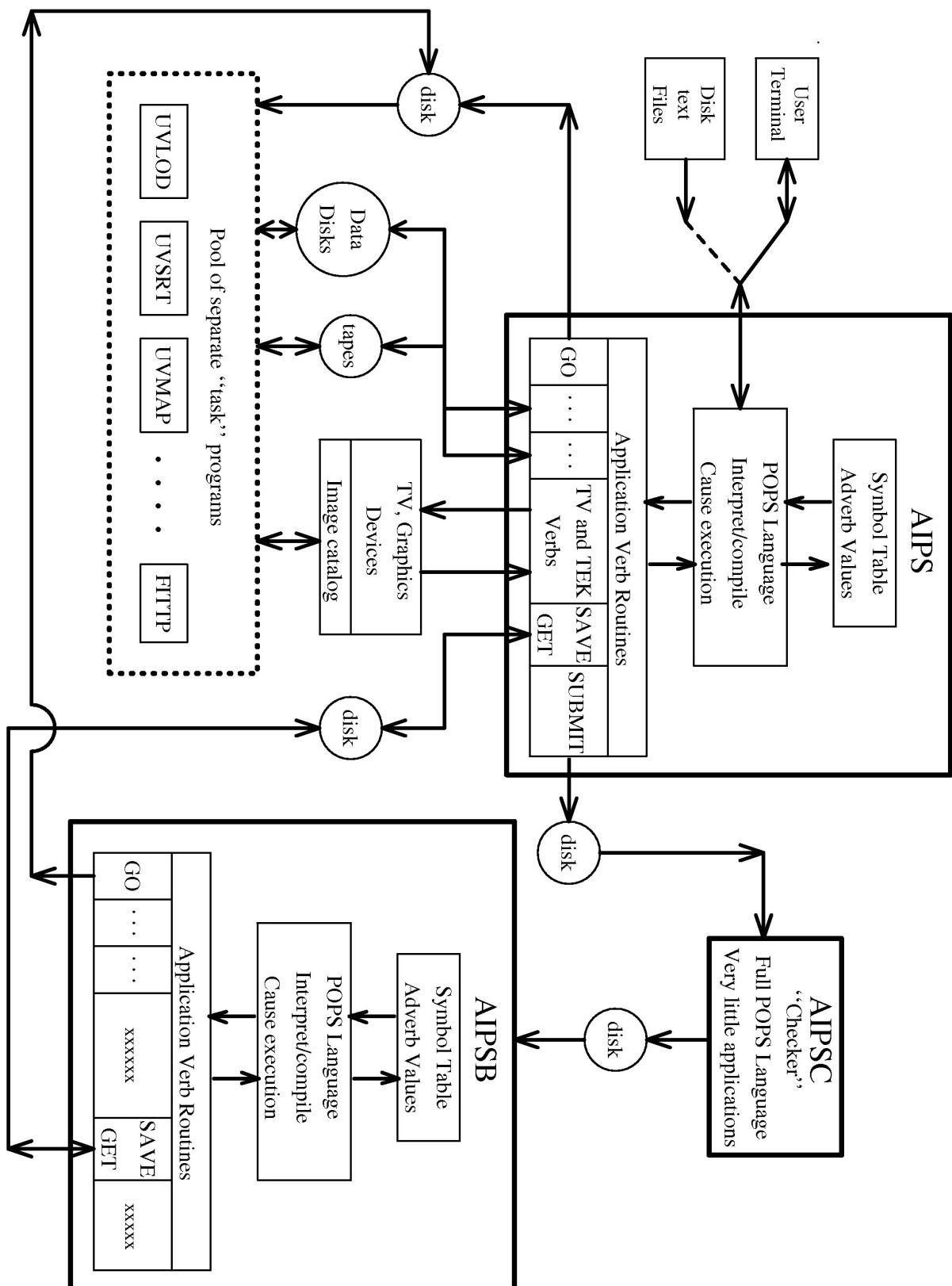
1.3.3 Fonts and what they signify

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

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

1.4 General structure of *AIPS*

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

Figure 1.1: Basic *AIPS* flow diagram

2 Starting Up *AIPS*

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

2.1 Obtaining access to an *AIPS* computer

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

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

2.2 Using the workstation

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

2.2.1 Logging in to the workstation

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

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

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

> `KLEENEX C_R`

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

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

2.2.2 Control characters on the workstation

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

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

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

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

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

2.2.3 Starting the AIPS program

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

To begin AIPS, enter

```
% aips CR with no options initially
```

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

```
You seem to be at a workstation called monkey
```

```
Starting local TV servers on monkey
```

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

```
Starting up 31DEC16 AIPS with normal priority
```

```
BEGIN THE ONE TRUE AIPS NUMBER n (release of 31DEC16) at priority 0
```

where 31DEC16 identifies the release of *AIPS* and *n* is a number between 1 and 6 (typically). If this is the only *AIPS* session on the computer, you should be assigned *n* = 1, with higher numbers used for additional sessions. If you start with *n* > 1, someone else may be using your computer remotely. AIPS will then tell which TV and graphics devices have been assigned to you:

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

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

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

```
AIPS n: Enter user ID number
```

```
? uuuu CR
```

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

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

```
XASERVERS: Start TV LOCK daemon TVSERV on monkey
```

```
TVSERVER: Starting AIPS TV locking, Inet domain
```

```
XASERVERS: Start XAS on monkey, DISPLAY monkey:0
```

```
XAS: ** TrueColor FOUND!!!
```

```
XAS: *** Using shared memory option for speed ***
```

```

XAS: Using screen width height 1270 924
      max grey level 8191 in 16 grey-scale memories
XASERVERS: Start graphics server TEKSRV on monkey, DISPLAY monkey:0
XASERVERS: Start message server MSGSRV on monkey, DISPLAY monkey:0

```

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

```

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

```

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

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

```

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

```

DESCRIPTION

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

OPTIONS

All command line options are case insensitive.

AIPS allows up to three versions to co-exist (disk space permitting) in one installation. They are identified either by date (e.g. 15OCT98) or name (OLD, NEW, or TST). On most installations, these will all be the same.

- old Start the OLD version of AIPS. For NRAO this is a frozen version which has been distributed worldwide.
- new Start the NEW version of AIPS. For NRAO this is the most recently released version and is frozen right at the time of initial public release.
- tst Start the TST version of AIPS. For NRAO this is the unreleased, development version. This is the default.

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

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

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

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

NOTE: The default TV behavior is to use INET or Internet based sockets, as the scripts have been doing since 1992. The "local" Unix socket based functionality does not change this.

For the default use of internet sockets, the full syntax of the TV= option is TV=tvdisp:tvhost, where tvhost is the name of the machine on which the TV display server (usually XAS), Tektronix graphics server (TEKSRV), message server (MSGSRV), and TV Lock server (TVSERV) are to run, and tvdisp indicates the machine to which the DISPLAY environment variable should point for XAS. Do NOT specify TV=hostname:0.0! Both TVHOST and TVDISP can be different from the machine that AIPS itself is running on. See the section on X Window System servers below for more information on how to control the servers.

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

TV=tvhost tvdisp defaults to tvhost.

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

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

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

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

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

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

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

AIPS has a limit of 71 disks in any one session. The limit on the number of disks that can be defined for any given site is 512. Disk 1 is special in that it stores the AIPS message and save/get files. The system is designed so that one particular required disk will almost always be assigned as disk 1. For performance reasons, this may be undesirable if the filesystem in question is mounted via NFS. See the description of personal .dadevs files below, as it can be used to customize the list of possible user data areas.

Selecting DA=ALL will try to include every area defined in the startup file, up to the session limit. Bear in mind that most AIPS tasks only have 10 slots for "BADDISK". Selecting DA=DEFAULT will completely bypass the configurable data areas and choose only those data areas preconfigured by the AIPS manager; THIS IS NOT NORMALLY ENABLED, CHECK WITH YOUR AIPS MANAGER BEFORE USING DA=DEFAULT.

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

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

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

\$NET0/DADEVS.LIST which is the site-wide data area configuration file.

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

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

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

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

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

TP=host[,host,...]

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

remote hosts.

- TPOK Do NOT check or launch the TPMON tape daemons on the local host. The default is to check if they are running and to launch them if not found.
- PR=# Select printer number (e.g., pr=2). If this option is not specified, the user will be presented with a menu of available printers and prompted to enter a choice. If there is only one printer configured, no menu will be presented. You may change the selected printer within AIPS via the PRINTER adverb.

REMOTE or REM or TEK

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

DEBUG[=prog] [:aips]

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

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

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

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

X WINDOW SYSTEM SERVERS

If you are running under the X Window System, AIPS will open up to three windows: a TV window (normally XAS), a message window (MSGSRV) and a graphics window (TEKSRV). If you specify the notv option on the command line, none of these will be started.

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

want to use cmdtool you would type

```
setenv AIPS_MSG_EMULATOR cmdtool
```

if you use the C or TC Shell, or

```
AIPS_MSG_EMULATOR=cmdtool; export AIPS_MSG_EMULATOR
```

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

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

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

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

ENVIRONMENT VARIABLES

In addition to the Message and Tek server customizations, you may choose to set a variable AIPSREMOTE to indicate your choice of remote shell command. It is strongly recommended that the secure shell (ssh) be used in place of the traditional Berkeley rsh or remsh command:

```
setenv AIPSREMOTE "ssh -n"
for csh or tcsh shells, or
    export AIPSREMOTE="ssh -n"
for bash, korn, zsh and other bourne-like shells.
```

If you do not specify a printer (by number) on the command line when starting AIPS, you will get a menu showing you all the alternative printers available. You should omit the PR option until you are familiar with the choices. The OLD version of *AIPS* is likely to be relatively free of bugs (provided the *AIPS* version in NEW does not prescribe format changes which prevent OLD from working), but the NEW version will contain improvements and will be mostly debugged. The TST version is a debugging area recommended for NRAO staff and those few users who may require the most recent software. (Note that this choice affects only the version of the AIPS program itself. You may choose TST, NEW or OLD versions of the *AIPS* reduction programs at a later time — see § 3.5.)

2.2.4 Typing commands to the AIPS program

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

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

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

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

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

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

The basic editing key bindings are:

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

DELETE and BACKSPACE work as expected.

2.3 Managing windows

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

2.3.1 General window management

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

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

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

2.3.2 Managing the AIPS TV window called XAS

On workstations, AIPS simulates a real TV display with a program called [XAS](#). The program starts when you start AIPS and comes up in an open form. Its icon shows a cute drawing of an ape along with words like AIPSTV UNIX for local display and AIPSTV INET for Internet-connectivity. In many ways, this is a normal window which can be resized, moved, iconified, and destroyed like any other. However, when the window is open and the cursor positioned inside the window, [XAS](#) offers some additional features. The cursor changes shape and color in the window to indicate this fact. To get [XAS](#) to treat the cursor position as a “TV cursor” position, you must hold down the left mouse button. This allows the cursor to fill two rôles

at nearly the same time, that of a workstation cursor and of a TV cursor. You do not have to hold the button down for long to register a TV position and, in fact, it is more efficient in interactive TV operations simply to click the left button at the desired locations. When you drag the cursor, numerous intermediate values are read with consequent extra computation. Note that the TV cursor position is read by **XAS** whenever the cursor is in the **XAS** window with the left button down. However, that position is only used when some verb or task reads it from **XAS** and uses it for some purpose, *e.g.*, to select image coordinates or to control image enhancement.

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

2.4 Additional recipes

2.4.1 Banana daiquiri

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

2.4.2 Banana pick-me-up

1. Slice ripe, peeled **bananas** into 3 cm chunks.
2. Wrap each chunk in strip blanched **bacon**.
3. Prepare mixture of **brown sugar** and **cinnamon** to taste.
4. Sprinkle mixture over banana chunks.
5. Bake at 350° F until the bacon is crisp and the sugar slightly caramelized.

3 Basic *AIPS* Utilities

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

3.1 Talking to *AIPS*

3.1.1 *POPS* and *AIPS* utilities

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

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

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

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

3.1.2 Tasks

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

A typical task setup will look like:

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

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

> **INP** C_R
 > **GO** C_R

to review the parameter values that you have set and
 to send the task into execution.

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

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

> **ABORT** C_R to kill the task named by **TASK**, or
 > **ABORT** *task.name* C_R to kill *task.name*.

This will stop the job quickly and delete any standard scratch files produced by it. However, input data files — and output data files that are probably useless — may be left in a “busy” state in your data catalog. The catalog file is described in § 3.3, including methods to clear the “busy” states and to delete unwanted files.

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

3.1.3 Verbs

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

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

3.1.4 Adverbs

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

> **CELL** 0.5 C_R to set a single scalar **CELL** to 0.5
 > **CELL** 1/2 C_R alternate for above with *POPS* in-line arithmetic
 > **IMSIZE** 512,256 C_R to set a two-element array **IMSIZE** to **IMSIZE(1)=512, IMSIZE(2)=256**
 > **IMSIZE** 512 256 C_R an alternate for the above if both values are positive
 > **IMSIZE** 256+256,256 C_R an alternate for the above using *POPS* in-line arithmetic

3.2. Your AIPS message file

3. Basic AIPS Utilities

> UVWTFN 'NA' C _R	to set a string variable UVWTFN to the value NA
> LEVS 0 C _R	to set all elements of the 30-element array LEVS to zero
> LEVS = -2,-1,1,2,3,4,5 C _R	to set LEVS(1)=-2, LEVS(2)=-1, LEVS(3)=1, etc. The = avoids in-line arithmetic that would otherwise subtract 2 from LEVS(1)
> LEVS = -2,-1 1 2 3 4 5 C _R	an alternate for the above; the comma avoids in-line arithmetic that would otherwise set LEVS(1) = -3

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

> INP C _R	to review the parameters for task TASK, or
> INP task_name C _R	to review the adverbs for task task_name.

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

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

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

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

3.2 Your AIPS message file

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

> HELP MSGSRV C_R for details.

At most AIPS installations, you get a message server by default. You may also control the size and appearance of the message server with parameters in the X-Windows .Xdefaults file. These parameters are also listed in by HELP MSGSRV C_R.

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

for the display and to let you control which messages will be displayed. For example, to set the minimum priority level for messages to be displayed, type:

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

> **DOCRT** -1 C_R to select the line printer or a text file under control of **OUTPRINT**.
> **DOCRT** 1 C_R to select the terminal at its current width ≥ 72 characters
> **DOCRT** *nc* C_R to select the terminal at width *nc* characters: $72 \leq nc \leq 132$.

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

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

> **CLRMSG** C_R to delete messages and to compress the message file.
CLRMSG supports adverbs like those of **PRTMSG**, except that the deletion is of messages older than **PRTIME** and the printing is of messages younger than **PRTIME** seconds ago. Old messages are automatically deleted from your message file when you **EXIT** from *AIPS*. (The time limit for "old" messages is set by your local *AIPS* Manager. Usually, it is about 3 days.)

3.3 Your AIPS data catalog files

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

> **INDI** 0 ; **MCAT** C_R to list all images on all disks, or
> **INDI** 0 ; **UCAT** C_R to list all *uv* data sets on all disks.

MCAT and **UCAT** list files based on the **INDISK** adverb. In 31DEC12, **M2CAT** and **U2CAT** list files based on the **IN2DISK** adverb. **M3CAT**, **U3CAT**, **M4CAT**, **U4CAT**, **OCAT**, and **UOCAT** are similar for the **IN3DISK**, **IN4DISK**, and **OUTDISK** adverbs, respectively.

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

> **CLRNNAME** C_R to reset **INNAME**, **INCLASS**, **INSEQ**, **INTYPE**, and **INDISK**,
> **CATALOG** C_R to generate the listing.

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

```
CATALOG ON DISK 1
CAT USID MAPNAME      CLASS   SEQ   PT      LAST ACCESS      STAT
18    76 3C166L50K    .IIM001.    1 MA 27-OCT-1996 22:30:18
19    76 3C166L50K    .IBM001.    1 MA 27-OCT-1996 23:02:14
22    76 3C166L50K    .IIM001.    2 MA 28-OCT-1996 15:30:45
CATALOG ON DISK 2
```

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

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

Note that this user has encoded useful information other than the source name into the image file names on disk 1. These images were of 3C166 at L band with 50 kilo-wavelength (*uv*) taper. Such information is also carried in *AIPS* history files (see § 3.4 below), but it is often useful to place it at a level where CAT can see it. The user also gave the **UVSRT** file in slot 2 on disk 3 a name that encodes the source name (3C138), the **VLA** configuration (A), and the observing band (C). Careful choice of *AIPS* filenames can save much other bookkeeping. The file name can be any valid string up to 12 characters long. Also note how SEQ numbers distinguish different versions of a file with the same name; this and the global variables in *AIPS* are helpful features when doing iterative computations such as self-calibration.

3.3.1 Speedy data file selection

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

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

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

> **IN2D** *n2* ; **GET2N** *ctn2* **C_R** to set the second set, and

> **IN3D** *n3* ; **GET3N** *ctn3* **C_R** to set the third set.

> **IN4D** *n4* ; **GET4N** *ctn4* **C_R** to set the fourth set.

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

3.3.2 Catalog entry status

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

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

3.3.3 Renaming data files

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

> INDI 2 ; INNA '1200+519' C_R	to select disk 2 and set the input (old) name.
> INCL 'IIM001' ; INSEQ 1 C_R	to set the rest of the input name adverbs, <i>i.e.</i> , to select the file in slot 22 on disk 2 in the example above.
> OUTN '1200+51 15K' ; OUTSEQ 2 C_R	to set desired output name and sequence number.
> INP RENAME C_R	to review the inputs.
> RENAME C_R	to rename the I image to '1200+51 15K' and reset its sequence number to 2.

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

3.3.4 Header listings

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

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

Multi-source *uv* data files may have many extensions (see Chapter 4). The most important are the HIstory file, the ANtennas file (subarray geometric data, date, frequency and polarization information, *etc.*), the BP (bandpass) file for bandpass calibration data, the CL (calibration) file for calibration and model information,

the FQ (frequency) file for frequency offsets of the different IFs, the FG (flag) file for editing information, the NX (index) file (which assists rapid access to the data), the SN (solution) file for gain solutions from AIPS calibration routines, and the SU (source) file with source-specific information such as name, position, and velocity. Chapter 4 describes the use of these extensions in some detail.

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

```
> IMHEAD C_R                                for a detailed listing, or
> QHEADER C_R                               for a shorter listing.
```

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

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

```
Image=3C345      (UV)          Filename=Z17G1_A      .MULTI .   1
Telescope=SBLNKGYO        Receiver=VLBI
Observer=FAP            User #= 1353
Observ. date=27-FEB-1991    Map date=13-JUN-1995
# visibilities     112813    Sort order TB
Rand axes: UU-L VV-L WW-L BASELINE TIME1 WEIGHT SCALE
                           SOURCE
```

Type	Pixels	Coord value	at Pixel	Coord incr	Rotat
COMPLEX	1	1.0000000E+00	1.00	1.0000000E+00	0.00
STOKES	4	-1.0000000E+00	1.00	-1.0000000E+00	0.00
FREQ	128	2.2228990E+10	63.50	5.0000000E+05	0.00
RA	1	16 41 17.608	1.00	3600.000	0.00
DEC	1	39 54 10.820	1.00	3600.000	0.00

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

Output from **IMHEAD** on an image file might look like:

```
Image=3C219      (MA)          Filename=3C219-BC-6 .ICL001.   1
Telescope=VLA           Receiver=
Observer=BRID          User #= 76
Observ. date=06-SEP-1992 Map date=18-APR-1994
Minimum=-1.89720898E-04 Maximum= 5.05501366E-02 JY/BEAM
```

Type	Pixels	Coord value	at Pixel	Coord incr	Rotat
RA---SIN	510	09 17 50.662	263.00	-0.300000	0.00
DEC--SIN	640	45 51 43.555	294.00	0.300000	0.00
FREQ	1	4.8726000E+09	1.00	2.5000000E+07	0.00
STOKES	1	1.0000000E+00	1.00	1.0000000E+00	0.00

Map type=NORMAL	Number of iterations=	50000
Conv size= 1.40 X 1.40	Position angle=	0.00
Observed RA 09 17 50.600	DEC 45 51 44.00	
Maximum version number of extension files of type HI is	1	
Maximum version number of extension files of type PL is	5	
Maximum version number of extension files of type SL is	1	

Both **QHEADER** and **IMHEAD** list the maximum version numbers of the table extension files associated with a data set. Because you may acquire many versions of such tables during calibration, these verbs are often invoked during calibration in *AIPS*. In 31DEC12, verbs **IM2HEAD**, **IM3HEAD**, **IM4HEAD**, and **IMOHEAD**, as well as **Q2HEADER**, **Q3HEADER**, **Q4HEADER**, and **QOHEADER** appeared to make comparable listings for the second, third, and fourth input and the output file naming adverb groups, respectively.

3.4 Your *AIPS* history files

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

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

> INDISK <i>n</i> ; GETN <i>ctn</i> C_R	to select the file to be displayed.
> PRTASK ‘ UVMAP ’ C_R	to examine only history information from UVMAP .
> DOCRT 1 C_R	to direct the display to your terminal, using its full width.
> PRTHI C_R	to print the UVMAP history.
> PRTASK ‘ ’ ; DOCRT FALSE C_R	to select all history cards and direct the output to the line printer.
> PRTHI C_R	to print the full history file.

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

3.5 Saving and restoring inputs

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

> INP <i>name</i> C_R	where <i>name</i> is the program name, or
> INPUTS <i>name</i> C_R	to write the input values to the message file.

> **QINP** C_R to resume the previous **INP** or **INPUTS** with the page last displayed.

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

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

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

> **DEFAULT** name C_R to reset the values for *name*.

> **DEFAULT** C_R to reset the values for the verb or task named in the **TASK** adverb.

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

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

> **GET** *aaaaa* C_R will restore these inputs later.

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

> **RESTORE** 0 C_R

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

> **STORE** 1 C_R to save your inputs in area 1, and

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

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

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

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

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

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

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

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

and retrieved by the command:

> **TGET** *name* C_R

TPUT and **TGET** allow you to avoid, to some extent, the global nature of the adverb values in AIPS. This is sometimes advantageous. Whenever a task (or a verb, for that matter) is executed by the verb **GO**, **TPUT** runs automatically. **TGET** will therefore recover the last set of input adverbs used to execute the task, unless you deliberately overwrite them with a **TPUT** of your own. Note that *AIPS* will complain if you try to **TGET** input adverbs for a task for which no **TPUT** has previously been run (either manually or automatically). You must “put” before you can “get.” **TGINDEX** will show you what tasks have been **TPUT** and when. **VPUT**, **VGET**, and **VGINDEX** allow you to save, recover, and list task-specific adverbs from up to 35 completely user-

controlled storage areas. In 31DEC14, all **TGET** and **VGET** files have a new format which includes adverb names. Then, when the adverbs to a task or verb change, **TGET** and **VGET** will still work. **PLGET** lets you recover the adverbs used to construct a user-selected plot file. Unfortunately, these cannot be known by name.

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

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

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

3.6 Monitoring disk space

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

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

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

> **USER** 32000 ; **INDISK** 0 C_R to get all disks and users.

> **DOALL** 0 ; **GO DISKU** C_R to run the *AIPS* disk user task.

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

> **USER** 0 ; **INDISK** *n* C_R where *n* is the mysteriously eaten disk

> **DOALL** *m* ; **GO DISKU** C_R to run the job, listing all catalog entries requiring more than *m* Megabytes.

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

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

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

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

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

3.7 Moving and compressing files

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

3.8 Finding helpful information in AIPS

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

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

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

> **HELP ADVERBS** C_R to get a list of all adverbs in the symbol table
 > **HELP ARRAYS** C_R to get a list of all array adverbs in the symbol table

> HELP REALS C _R	to get a list of all real adverbs in the symbol table
> HELP STRINGS C _R	to get a list of all character string adverbs in the symbol table
> HELP VERBS C _R	to get a list of all verbs, pseudoverbs, and procedures in the symbol table
> HELP PSEUDOS C _R	to get a list of all pseudo verbs in the symbol table
> HELP PROCS C _R	to get a list of all procedures in the symbol table

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

> **HELP HELP** C_R for help on **HELP**.

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

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

> APROPOS CLEAN C _R	to display all keyword and 1-line summaries of help files containing words beginning with "clean" (in upper and/or lower case), and
> APROPOS 'UV PLOT' C _R	note the quote marks which are required if there are embedded blanks, or
> APROPOS UV,PLOT C _R	to display all keyword and 1-line summaries of help files containing both words beginning with "uv" and words beginning with "plot."

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

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

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

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

ADVERB	POPS symbol holding real or character data
ANALYSIS	Image processing, analysis, combination
AP	Tasks using the "array processor"
ASTROMETry	Accurate position and baseline measurements
BATCH	Running AIPS tasks in AIPS batch queues
CALIBRATION	Calibration of interferometer uv data
CATALOG	Dealing with the AIPS catalog file
COORDINATES	Handling image coordinates, conversions
EDITING	Editing tables, uv and image data.
EXT-APPL	Access to extension files (tables)
FITS	FITS format for data interchange

GENERAL	General AIPS utilities
HARDCOPY	Creating listings and displays on paper
IMAGE-UTIL	Utilities for handling images
IMAGE	Transforming of images
IMAGING	Creation of images: FFT, Clean, ...
INFORMATION	General lists and user help functions
INTERACTiVe	Functions requiring user interaction
MODELING	Model fitting to uv or image data
OBSOLETE	Functions slated for removal
ONED	Functions for one-dimensional image slices
OOP	Tasks coded with object oriented principles
OPTICAL	Functions of interest for optical astronomy data
PARAFORM	Skeleton tasks for use in building new tasks
PLOT	Displays of image and uv data
POLARIZATION	Calibration, analysis, display of polarization
POPS	Aspects of the AIPS' user language POPS
PROCEDURE	Creation of and available procedures
PSEUDOVERb	Pseudoverbs in the POPS language and AIPS
RUN	Creation of and available RUN files
SINGLEDIsh	Functions of interest for single-disk radio data
SPECTRAL	Functions for spectra-line and other 3D data
TABLE	AIPS table extension files
TAPE	Use of magnetic tapes
TASK	AIPS tasks - available asynchronous functions
TV-APPL	Tasks using the TV display
TV	Basic functions on the TV display
UTILITY	Basic functions on tables, uv and image data
UV	Functions dealing with interferometer uv data
VERB	Synchronous functions inside the AIPS program
VLA	Functions of particular interest for the VLA
VLBI	Functions of particular interest for very long baseline data.

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

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

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

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

3.9 Magnetic tapes

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

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

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

In case you forget this list, the verb **TAPES** will show it to you. **TAPES** is even capable of going out on the Internet and asking what devices are available to an *AIPS* user at the computer specified by the **REMHOST** adverb (if it is running **TPMON**)!

3.9.1 Hardware tape mount

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

shows). Note the identification number *m* marked on the drive you are using, as you will need to provide that number to the software for mounting and dismounting the tape and for executing *AIPS* tasks which read or write tape.

3.9.2 Software mounting local tapes

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

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

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

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

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

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

3.9.3 Software mounting REMOTE tapes

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

- > **REMHOST** ‘RHECUS’ ; **REMTAPE** 2 C_R
- > **DENSITY** *dddd* C_R to set the density to *dddd* bpi if needed.
- > **INTAPE** *n* ; **MOUNT** C_R set local “tape” number and software mount

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

3.9.4 Using tapes in *AIPS*

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

- > **TASK** ‘PRTTP’ ; **INP** C_R to review the inputs.
- > **NFILES** 0 C_R to list all files on the tape.

- > **PRTLEV 0 C_R** to list the image headers but not the details — both more and less detailed listings are available.
- > **DOCRT FALSE C_R** to print on the line printer.
- > **GO C_R** to run the task.

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

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

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

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

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

3.10 AIPS external disk files

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

3.10.1 Disk text files

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

```
% setenv MYAREA 'pwd' CR
```

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

```
$ MYAREA='pwd'; export MYAREA C_R
```

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

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

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

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

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

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

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

3.10.2 RUN files

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

```
% cd $RUNFIL
```

to change to `RUN` area.

```
% emacs MAPIT.03D C_R
```

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

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

```
> VERSION = 'MYAREA' CR where MYAREA is your disk area, or
> VERSION = ' ' CR if $RUNFIL is to be used
> RUN FILE CR to execute the file named FILE.uuu
```

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

3.10.3 FITS-disk files

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

http://fits.gsfc.nasa.gov/fits_home.html

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

```
> DATAOUT 'filename' CR where filename is the name of the desired output file.
```

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

```
> DATAIN 'filename' CR
```

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

```
> DATAOUT = 'MYDATA:3C123.FIT' CR
```

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

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

```
> FOR I=1:10; GETN(I); FITDISK; END CR
```

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

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

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

FITS-disk files are written as Fortran files and hence are available also to user-coded programs. The Fortran specifications for the file are `ACCESS='DIRECT'`, `RECL=2880`, `FORM='UNFORMATTED'` in the `OPEN` statement for Unix systems. Most Fortrancs cannot read or write files larger than 2 Gigabytes, so *AIPS* now reads and writes these files with C subroutines. Users may also, of course, code programs to create such files to be read by `FITLD`, `IMLOD` or `UVLOD`. Consult *GOING AIPS*, Volume 2, Chapter 13 for details on how to do this.

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

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

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

An alternative to using `ftp` is to use the `rcp` (remote copy) Unix utility or to write the output file directory in the appropriate area on the other computer. In order to do this, you have to have accounts on both machines, and you should have set up a `.rhosts` file (see the Unix manual page on `rhosts` for instructions). Once you know this works (test it via, e.g., `rsh rhesus whoami`), the syntax for the remote copy is:

```
% rcp $FITS/MYFILE.FITS kiowa:/AIPS/FITS/MYFILE CR
```

(this shows how you would copy it from `rhesus` to `kiowa`). A secure copy (`scp`) would be better if you have set up the secure connection capability.

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

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

3.10.4 Other binary data disk files

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

<http://archive.cv.nrao.edu/>

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

3.11 The array processor

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

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

In 31DEC12, the pseudo-AP was changed to use only double precision integers and floats. This avoids serious overflow issues with the large number of data samples from modern telescopes and provides much greater accuracy especially in the gridding and Fourier transforming of visibility data.

3.12 Additional recipe

3.12.1 Banana curried chicken

1. Fry 2 chopped **onions** in 50 ml **cooking oil** until light brown.
2. Add 1/4 cup **cake flour** and mix well. Add 1 (cup?) **chicken stock** gradually while stirring.
3. Add 1 cup **raisins**, 1 teaspoon **salt**, 2 pounds cooked, boned **chicken**, 5 sliced **bananas**, 2 grated **apples**, 2 tablespoons grated **lemon rind**, 1 tablespoon **sugar**, 1 1/2 tablespoons **curry powder**, 1 **bay leaf**, 4 **peppercorns**.
4. Cover saucepan and simmer for 20 minutes.
5. Remove bay leaf. Add 1 cup **cream** and heat just before serving.
6. Serve on a bed of rice. Decorate with pineapples if preferred.

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

4 Calibrating Interferometer Data

This chapter focuses on ways to do the initial calibration of interferometric fringe-visibility data in *ATPS*. The sections which follow concentrate primarily on continuum calibration for connected-element interferometers, especially the [VLA](#). However, the information in these sections is useful to spectral-line, solar, and VLBI observers as well. For specific advice on the new [EVLA](#), consult Appendix E. For additional advice on spectral-line calibration, see § 4.7; for advice on calibrating observations of the Sun, see § 4.8; and for the gory details of VLBI, read Chapter 9. After the initial calibration has been completed, data for sources with good signal-to-noise are often taken through a number of cycles of imaging with self-calibration. See § 5.4 for information on these later stages of the reduction process. For accurate calibration, you must have accurate *a priori* positions and structural information for all your calibration sources and accurate flux densities for at least one of them. It is best if the calibration sources are unresolved “point” sources, but it is not required.

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

1. AN (antennas) for sub-array geometric data, date, frequency, polarization information, *etc.*,
2. FQ (frequency) for frequency offsets of the different IFs (IF pairs in [VLA](#) nomenclature),
3. [NX](#) (index) to assist rapid access to the data,
4. SU (source) for source specific information such as name, position, velocity, and
5. TY (temperature) for measured system temperatures and nominal sensitivities
6. CL (calibration) for calibration and model information,

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

7. BP (bandpass) for bandpass calibration,
8. FG (flag) for flagging (editing) information, and
9. SN (solution) for gain solutions from the calibration routines.
10. BL (baseline) for baseline-, or correlator-, dependent corrections,

All of these tables can be written to, and read back from, FITS files along with the visibility data. These, and any other, *ATPS* tables can be manipulated and examined using the general tasks [PRTAB](#), [TACOP](#), [TABED](#), [TAMRG](#), [TASRT](#), [TAFLG](#) and [TAPPE](#).

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

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

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

> **RUN VLAPROCS C_R** to compile the procedures.

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

> **RUN VLARUN C_R** to compile the procedures.

> **INP VLARUN C_R** to review the input adverbs and, when ready,

> **VLARUN C_R** to execute the pipeline.

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

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

1. For **VLA** data from the archive, use **FILLM** to read one or more disk files; see § 3.10.4. The **VLA** format was changed on January 1, 1988, but all older data were translated and archived in the modern format. On July 1, 2007, the ModComps were replaced with modern computers and the format had an essential change made to it. Use **FILLM** to read data from the post-ModComp era. Archive data are obtained from <https://archive.nrao.edu/archive/e2earchive.jsp>
2. For an *AIPS* multi-source data set written to a FITS tape or FITS disk during an earlier *AIPS* session, use **UVLOD** or **FITLD** to read the tape.
3. For single-source data sets that are already on disk and are very similar in structure, use **UV2MS** on one of them to create a multi-source data set, and then on each of the others to append them to that multi-source data set. Each of the input data sets should have the same number of polarizations, IFs, spectral channels, and “random parameters.” **UV2MS** also makes no corrections for differences in observed source positions or frequencies. After all are appended, use **UVSRT** to put the data in time-baseline order and **INDXR** to make an index and initial (null) calibration file.
4. For single-source data sets that are already on disk and are not sufficiently similar in structure for the method above, use **MULTI** on each single-source file to convert to multi-source format. Then use **DBCON** to concatenate the individual multi-source files into one big multi-source file. Finally use **UVSRT**, if needed, to put the data in time-baseline order and **INDXR** to make an index and initial (null) calibration file.

5. Data from the Australia Telescope may be loaded from disk files into AIPS using the task [ATL0D](#) which is now included with AIPS.

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

4.1.1 Reading from VLA archive files using FILLM

The NRAO Archive makes available, among other things, data from the [EVLA](#), which began observing in January 2010, and from the old [VLA](#) which ceased observing a few days into 2010. To load data from the [EVLA](#) into AIPS consult Appendix E for information about [BDF2AIPS](#) and other options. The following describes how to read old [VLA](#) data into AIPS.

The Archive now serves [VLA](#) data in the form of one or more “MODComp” format disk files. To load these into AIPS, enter

> TASK 'FILLM' ; INP C _R	to review the inputs needed.
> DATAIN 'MYDATA.AG230.' C _R	to read from the disk area pointed at by the logical MYDATA and the data from program ID AG230.
> NFILES 0 C _R	to start with file AG230_1. If your first file is e.g., AG230_4, set NFILES = 3.
> NCOUNT 4 C _R	to read four data files AG230_1 through AG230_4.
> OUTNA '' C _R	to take the default output file name.
> OUTDI 3 C _R	to write the data to disk 3 (one with enough space).
> DOUVCOMP -1 C _R	to write visibilities in uncompressed format. VLA files are small by modern standards, so saving space is not worth the costs.
> DOWEIGHT 1 C _R	Data weights will depend on the “nominal sensitivity” and should be calibrated along with the visibility amplitudes (DOCALIB = 1).
> CPARM 0 C _R	to do no averaging of the data in FILLM .
> CPARM(6) 1 C _R	to select VLA sub-array 1.
> CPARM(7) 2000 C _R	to have observations within 2 MHz be regarded as being at the same frequency.
> CPARM(8) 1 C _R	to use a 2-minute interval for the CL table; default is 5 min.
> CPARM(9) 0.25 C _R	to use a 15-second interval for the TY table; default is the input data interval.
> DPARM 0 C _R	to have no selection by specific frequency.
> REFDATE 'yyyyymmdd' C _R	to specify the year, month, and day of the reference date. This should be the first date in the data set (or earlier). All times in AIPS will be measured with respect to that date and must be positive. The default is the first date included by the data selection adverbs, which may not be the desired one. Note that REFDATE is only a reference point; it does not affect which data are loaded from the the files.

> **TIMERANG** *db , hb , mb , sb , de, he , me , se* C_R

to specify the beginning day, hour, minute, and second and ending day, hour, minute, and second (wrt **REFDATE**) of the data to be included. The default is to include all times.

> **INP** C_R

to review the inputs.

> **GO** C_R

to run the program when you're satisfied with inputs.

There are numerous adverbs including **BAND**, **QUAL**, **CALCODE**, **VLAOBS**, and **VLAMODE** to limit what data were loaded from magnetic tapes which could hold data from multiple projects. These adverbs still function, but are of little use today. Note that the values given above are illustrative and should not be copied verbatim in most cases.

Be careful when choosing the averaging time with **CPARM(1)**. If you have a large data set, setting this time too *low* will make an unnecessarily large output file; this may waste disk space and slow the execution of subsequent programs. Setting it too *high* can, however, (1) smear bad data into good, limiting the ability to recognize and precisely remove bad data, (2) smear features of the image that are far from the phase center, and (3) limit the dynamic range that can be obtained using self-calibration. If you need a different (usually shorter) averaging time for the calibrator sources than for your program sources, use **CPARM(10)** to specify the averaging time for calibrators. See Lectures 12 and 13 in *Synthesis Imaging in Radio Astronomy*¹ for general guidance about the choice of averaging time given the size of the required field of view and the observing bandwidth.

CPARM(2) controls a number of mostly esoteric options. If your data include the Sun or planets, you must set **CPARM(2)** = 16 to avoid having each scan on the moving source assigned a different name. The adverb **DOWEIGHT** = 1 has the same affect as **CPARM(2)** = 8 and both select the use of the nominal sensitivity to scale the data weights. When this is done, the weights will be $1/\sigma^2$ as they should for imaging, with σ in "Jy" in the same uncalibrated scale as the fringe visibilities. Having selected this option, you should apply any amplitude calibration to the weights as well as the visibilities. If you store the data in compressed form, only one weight may be retained with each sample. Any differences between polarizations and/or IFs in that sample will be lost. Uncompressed data require less cpu, but more real, to read but 2 to 3 times as much disk space to store.

CPARM(2)=2048 allows you to load data as correlation coefficients, which can be scaled to visibilities later with **TYAPL** (§ 4.1.1.1). **CPARM(3)** controls which on-line flags are applied by **FILM**, which now always writes an OF table containing information about these flags. That information can be viewed with **PTOF** and applied selectively to the data at a later time with **OFLAG**.

FILM writes a weather (WX) table to the output file. At the same time, it uses "canned" **VLA** antenna gain curves and a balance of the current with a seasonal model weather data to estimate opacity and gain corrections to be written into the first calibration (CL) table. These functions are controlled by adverbs **CALIN** and **BPARM** and may be turned off, although the default is to make the corrections. In subsequent tasks, set **DOCALIB** = 1 to use these initial calibration data. If, for some reason, the data weights do not depend on the nominal sensitivity, use **DOCALIB=100** to apply calibration.

Where possible, **FILM** will try to place all data in one file. However, in many cases this is not possible. For instance so-called "channel 0" data from a spectral-line observation will be placed in a separate file from its associated line data. Similarly, scans which have differing numbers of frequency channels will also be placed into separate files. Another case is observations made in mode LP, *i.e.*, one IF-pair is set to L band, the other to P band. In this case the two bands will be split into separate files. Yet another case arises when there are observations of different bandwidths. All of this should be relatively transparent to the user.

FILM and many *AIPS* tasks are able to handle multiple, logically different, frequencies within a multi-

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

source data set. **FILLM** does this by assigning an FQ number to each observation and associating a line of information about that frequency in the FQ file associated with the data set. Users should note that this concept can become quite complicated and that not all tasks can handle it in full generality. In fact, most tasks can only process one FQ number at a time. Polarization calibration works only on one FQ at a time since the antenna file format allows for only one set of instrumental polarization parameters. Therefore, it is *strongly* advised that you fill continuum experiments which involve multiple frequencies into separate data sets. **FILLM** will separate bands automatically, but you will have to force any remaining separation. To do this, (a) use the **QUAL** adverb in **FILLM**, assuming that you have used separate qualifiers in **OBSERVE** for each frequency pair; (b) use the **DPARM** adverb array in **FILLM** to specify the desired frequencies precisely; or (c) use the **UVCP** task to separate a multiple FQ data set into its constituent parts. Note that the first two options require multiple executions of **FILLM**, while the third option requires more disk space.

Spectral-line users and continuum observers using different frequencies in the same band should be aware of the FQ entry tolerance. Each frequency in a *uv* file will be assigned an FQ number as it is read from disk by **FILLM**. For spectral-line users, the observing frequency will normally change as a function of time due to Doppler tracking of the Earth's rotation, or switching between sources or between spectral lines; in general, this will cause different scans to have different FQ numbers. **FILLM** assigns an FQ number to a scan based on the FQ tolerance adverb **CPARM(7)** which defines the maximum change of frequency allowed before a new FQ number is allocated. If **CPARM(7) < 0**, the same FQ number is assigned to all data in spectral-line data sets. If **CPARM(7)** is positive, a scan's will be assigned to an existing FQ number if

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

where $\nu_{firstFQ}$ is the frequency of the first sample to which the particular FQ number was assigned. If no match is found, then a new FQ number is created and assigned and another line added to the FQ table file. Alternatively, if **CPARM(7)** is zero, then the FQ tolerance is assumed to be half of the maximum frequency difference caused by observing in directions 180 degrees apart (*i.e.*, $\Delta\nu = 10^{-4} \times \nu$).

An example: if an observer observes the 1612, 1665 and 1667 MHz OH masers in VY CMa and NML Cygnus, then presumably he would like his data to have 3 FQ numbers, one associated with each OH transition. However, running **FILLM** with **CPARM(7)** set to 0 would produce 6 FQ numbers because the frequency difference between the masers in VY CMa and NML Cygnus is greater than the calculated tolerance of 160 kHz. Therefore, in order to ensure that only 3 FQ numbers are assigned, he should set **CPARM(7)** to 1000 kHz. Setting **CPARM(7) < 0** would result in all data having the same FQ number, which is clearly undesirable.

For most continuum experiments the FQ number will be constant throughout the database. Normally any change in frequency should be given a new FQ number. To achieve this, **FILLM** treats **CPARM(7)** differently for continuum. If **CPARM(7) ≤ 0.0**, then **FILLM** assumes a value of 100 kHz. A positive value of **CPARM(7)** is treated as a tolerance in kHz as in the spectral line case.

Note: If your *uv* database contains several frequency identifiers, you should go through the calibration steps for each FQ code separately.

FILLM can still read from magnetic tape. Set **DATIN** to blanks, mount your tape (adverb **INTAPE**), index the tape with **PRTTP**, and use the adverbs to limit the data loaded to that portion of your project in which you are interested.

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

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

This should produce a listing such as:

```
Catalog on disk 3
```

Cat	Usid	Mapname	Class	Seq	Pt	Last access	Stat
1	103	25/11/88	.X BAND.	1	UV	05-FEB-1994 12:34:16	

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

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

This should produce a listing like:

```
Image=MULTI      (UV)          Filename=25/11/88      .X BAND.      1
Telescope=VLA                Receiver=VLA
Observer=AC238               User #= 103
Observ. date=25-NOV-1988     Map date=05-FEB-1994
# visibilities   191317     Sort order TB
Rand axes: UU-L-SIN VV-L-SIN WW-L-SIN BASELINE TIME1
              SOURCE FREQSEL WEIGHT SCALE
```

Type	Pixels	Coord value	at Pixel	Coord incr	Rotat
COMPLEX	1	1.0000000E+00	1.00	1.0000000E+00	0.00
STOKES	4	-1.0000000E+00	1.00	-1.0000000E+00	0.00
IF	2	1.0000000E+00	1.00	1.0000000E+00	0.00
FREQ	1	8.4110000E+09	1.00	1.2500000E+07	0.00
RA	1	00 00 00.000	1.00	3600.000	0.00
DEC	1	00 00 00.000	1.00	3600.000	0.00

```
Maximum version number of extension files of type HI is 1
Maximum version number of extension files of type AN is 1
Maximum version number of extension files of type NX is 1
Maximum version number of extension files of type SU is 1
Maximum version number of extension files of type FQ is 1
Maximum version number of extension files of type WX is 1
Maximum version number of extension files of type CL is 1
Keyword = 'CORRMODE'  value = ,
Keyword = 'VLAIFS'   value = 'ABCD'
```

This header identifies the file as a multi-source file (Image=[MULTI](#)) with 191317 floating-point visibilities in time-baseline (TB) order. There are two entries on the IF axis. These correspond to the [VLA](#)'s "AC" and "BD" IF-pairs respectively. The description of the frequency ([FREQ](#)) axis shows that the first IF ("AC") is at 8411 MHz and has 12.5 MHz bandwidth. The parameters of the second IF-pair ("BD") are determined from the data in the FQ table file and cannot be read directly from this header; these values are shown in the '[SCAN](#)' listing from [LISTR](#). The header shown above indicates that the data are in compressed format since the number of pixels on the COMPLEX axis is 1 and the [WEIGHT](#) and [SCALE](#) random parameters are present. Uncompressed data does not use these random parameters and has 3 pixels on the COMPLEX axis.

The term "IF" can be confusing. At the [VLA](#), IFs "A" and "C" correspond to right-hand and left-hand circularly polarized (RHC and LHC) signals, respectively, and are normally for the same frequency in an observing band. Such pairs, if at the same frequency, are considered to be one "IF" in [AIPS](#). An observation which was made in spectral line mode "2AC" is considered at the [VLA](#) to have two "IFs" whereas within [AIPS](#) this would be filled as one "IF" with two polarizations if they were both observed with the same frequency, the same number of channels, and the same channel separation. If these conditions do not hold, then they are filled into separate *uv* files, each with a single IF and a single polarization. The term "sub-array" is also confusing. At the [VLA](#) — and in task [FILLM](#) — sub-array means the subset of the 27 antennas actually used to observe your sources. (The [VLA](#) allows up to 5 simultaneous sub-arrays in this sense.) In the rest of [AIPS](#), sub-array refers to sets of antennas used together at the same time. If observations from separate times (e.g., separate array configurations) are concatenated into the same file, then [AIPS](#) will regard the separate sets of antennas as different "sub-arrays" whether or not the same physical antennas occur within more than one of these sub-arrays.

If your experiment contains data from several bands **FILLM** will place the data from each band in separate data sets. Also, if you observed with several sets of frequencies or bandwidths in a given observing run these will be assigned different FQ numbers by **FILLM**. You can determine which frequencies correspond to which FQ numbers from the ‘SCAN’ listing provided by **LISTR**. Line data are divided into the “channel 0” (central 3/4 of the of the observing band averaged) and the spectra. Data observed in the “LP” mode (or any other two-band mode) will be broken into separate data sets, one for each band.

4.1.1.1 Editing and applying nominal sensitivities to VLA data

FILLM scales the correlation coefficients by the instantaneous measured “nominal sensitivities,” producing data approximately in deci-Jy. The **VLA** nominal sensitivities are stored in the **TY** table as “system temperatures” (T_{sys}). For calibration purposes, it is best to have the nominal sensitivities applied, but it may be better to use a clipped and/or time-smoothed version of those sensitivities. If you want to do this, load the T_{sys} data into the **TY** table with the highest time resolution possible by setting **CPARM(9)=0** in **FILLM**. **FILLM** can also be told not to apply the nominal sensitivities and therefor produce correlation coefficients by setting **CPARM(2)=2048**, but this is not strictly necessary. In order to smooth and clip the **TY** table use the task **TYSMO**. If you have done editing such as **QUACK**, it may help to copy the data with **UVCOP**, applying your flag table not only to the visibilities but also to the **TY** table (**UVCOPPRM(6)=3**) before running **TYSMO** to remove questionable values at the start of scans. Alternatively, **SNEDT** will apply the data flags to the table allowing you to write a new, cleaned-up version of the table. Then a **TY** table may be applied (and/or removed) from a data set with **TYAPL**:

> TASK ‘TYAPL’ ; INP C_R	to review the inputs needed.
> INDI n ; GETN m C_R	to select the correct data set.
> FREQID 1 C_R	to select FQ number 1.
> INVERS 1 C_R	TY table to remove from data, will only work if data are not already correlation coefficients.
> IN2VERS 2 C_R	smoothed TY table to apply to data, will only work if data is in correlation coefficient form — either initially or after removal of INVERS .
> INP C_R	to re-check <i>all</i> the inputs parameters.
> GO C_R	to start the task.

EVLA users (see Appendix E) should have an **SY** table which contains system gain and temperature data. These data *should* be applied to the visibility data in order to correct for measured gain changes and to convert the data weights from a simple count of the integration time into more meaningful values. Tasks **TYSMO** and **SNEDT** may be used to clean up the **SY** data and then **TYAPL** may be used to apply the gain and system temperature data.

4.1.2 Reading data from FITS files with FITLD

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

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

> TASK ‘FITLD’ ; INP C_R	to review the inputs needed.
--	------------------------------

> INTAPE <i>n</i> C _R	to specify the tape drive for input from tape.
> DATAIN ' <i>filename</i> ' C _R	if the input is from a FITS disk file (see § 3.10.3).
> DOUVCOMP FALSE C _R	to write visibilities in uncompressed format.
> OUTNA '' C _R	take default (previous <i>AIPS</i>) name.
> OUTCL '' C _R	take default (previous <i>AIPS</i>) class.
> OUTSEQ 0 C _R	take default (previous <i>AIPS</i>) sequence #.
> OUTDI 3 C _R	to write the data to disk 3 (one with enough space).
> INP C _R	to review the inputs (several apply only to VLBA format files).
> GO C _R	to run the program when you're satisfied with inputs.

FITLD is the equivalent of **FILLM**, but for output from the VLBA, rather than the **VLA**, correlator. The data-selection adverbs **SOURCES**, **QUAL**, **CALCODE**, and **TIMERANG** and the table-control adverbs **CLINT** and **FQTOL** are used, for VLBA-format data only, in **FITLD** in ways similar to the data-selection and control adverbs of **FILLM**. See Chapter 9 for more specific information.

4.2 Record keeping and data management

4.2.1 Calibrating data with multiple FQ entries

In general an observing run with the **VLA**, especially a spectral-line run, will result in a *uv* data file containing multiple FQ entries. In early versions of the *AIPS* software, the different FQ entries would automatically have been placed in different physical files. Now, **FILLM** allows you to place all of them in the same file. This may be convenient, but it has a number of costs. If a file contains multiple, independent frequencies, then it occupies more disk space and costs time in every program to skip the currently unwanted data (either a small cost when the index file is used or a rather larger cost when the file must be read sequentially). Since multiple frequencies are still not handled correctly in all programs (*i.e.*, polarization calibration) and since it is not possible to calibrate all of the different FQ data in one pass, you might consider separating the multiple frequencies into separate files (as described in § 4.1.1). In either case, you must calibrate each frequency with a separate pass of the scheme outlined below. There are three adverbs to enable you to differentiate between the different FQ entries: **FREQID** enables the user to specify the FQ number directly (with -1 or 0 meaning to take the first found); **SELFREQ** and **SELBAND** enable the user to specify the observing frequency and bandwidth to be calibrated (the tasks then determine to which FQ number these adverbs correspond). If **SELFREQ** and **SELBAND** are specified they override the value of **FREQID**.

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

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

4.2.2 Recommended record keeping

It is useful to print a summary of the time stamps and source names of the scans in your data set. This reminds you of the structure of your observing program when you decide on interpolation and editing

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

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

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

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

- | | |
|---------------------------------|---|
| > NPRINT 0 C_R | to do all antennas |
| > INVERS n C_R | to do sub-array n |
| > GO PRTAN C_R | to print the list and a map of antenna locations. |

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

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

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

4.3 Beginning the calibration

After loading the data to disk, it has been traditional to begin with a substantial session of data checking and editing. With data from the **VLA**, this is always time consuming and often not necessary. Nonetheless, it is probably a good idea to check for two specific kinds of problems before beginning the actual calibration. These are corrupted data in the first record of most scans and totally dead antennas. Many other problems in the data are quickly and easily diagnosed by carefully inspecting the solution tables produced from the calibrators on un-edited data. Missing antennas and erratic amplitudes due to sampling problems and RF interference can be spotted from the **SN** tables and the closure-error messages produced by **CALIB**. If you *can't* spot errors from these, you may not need to edit the calibrator data. If the **SN** tables have well-behaved

phases for most antennas and rapidly rotating phases for one or two, then you may need to apply baseline corrections rather than editing. See § 4.4.4 for details of how to make antenna-position corrections.

The next section tells how to detect simple problems in the data and eliminate them to reduce the warnings from the calibration tasks. The following sections tell you how to enter fluxes for the primary calibrator sources and do a preliminary calibration for all calibrators. In so doing, you should generate one or more solution (SN) tables containing the complex gains at the times of the calibration observations. These tables may be examined for problems with the observations. If you find problems, then you need to edit the data or apply baseline corrections and should consult § 4.4. If you do not find problems, you may proceed directly to § 4.5. (Of course, you may decide to edit the data from your program sources at a later stage of the data reduction and have to return to § 4.4 then.)

4.3.1 Initial editing

The warning messages from the calibrations described in the next sections may be reduced by flagging those antennas which were not actually working, but which were not flagged by the on-line system. Another problem that has plagued the VLA (and other interferometers) persistently is that the first record in scans can be corrupted; usually its amplitudes are lower than they should be. These data can be flagged using **TVFLG** or **UVFLG**, but this can be time consuming. The task **EDITA** described in § 4.4.2 is now likely to be the best initial (and perhaps only) editing tool which you need. For a more traditional approach, we recommend that you do the following before beginning your regular data editing. Use the task **LISTR** on your terminal (to save time and paper) to see if you have the problem:

> TASK 'LISTR' C_R	to set the data listing task
> INDI n ; GETN m C_R	to select the data set, $n = 3$ and $m = 1$ above.
> OPTYPE 'LIST' C_R	to select column listing format
> ANTEN a1 , 0 C_R	to select one reliable antenna to display.
> BASEL 0 C_R	to select all baselines to this antenna.
> SOURCES '' ; CALCODE '*' C_R	to select all calibrator sources only.
> TIMER 0 C_R	to select all times.
> STOKES 'RR' C_R	to examine only one Stokes at a time.
> BIF 1 ; EIF BIF C_R	to specify the "AC" IFs only; it is quicker to look at only 1 IF at a time although more than one can be listed in sequence.
> FREQID 1 C_R	to select FQ number (each FQ number must be done separately).
> DOCRT 132 C_R	to see full width display on the terminal. Use your window manager to stretch the window to ≥ 132 characters width.
> DOCALIB -1 C_R	to turn off calibration.
> DPARM 0 C_R	to select amplitudes with no averaging.
> INP C_R	to re-check <i>all</i> the inputs parameters.
> GO C_R	to start the task.

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

> TASK 'QUACK' C_R	
> SOURCES '' C_R	to select all sources.
> TIMER 0 C_R	to select all times.
> ANTENNAS 0 C_R	to select all antennas.

> FLAGVER 1 C _R	to insert flagging information in FG table 1.
> OPCODE 'BEG' C _R	flag first APARM(2) min of each scan.
> REASON 'BAD START OF SCAN' C _R	reason for the flagging.
> APARM 0 , 1/6 , 0 C _R	flag first 10 seconds of each scan.
> GO C _R	

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

> BIF 2 ; EIF 2 C _R	to specify the “BD” IFs.
> GO C _R	to run the program again.

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

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

> TASK 'UVFLG' ; INP C _R	to select the editor and check its inputs.
> TIMER 0 C _R	to select all times.
> BIF 1 ; EIF 2 C _R	to specify the “AC” and “BD” IFs.
> BCHAN 0 ; ECHAN 0 C _R	to flag all channels.
> FREQID 1 C _R	to flag only the present FQ number.
> ANTEN 6 , 9, 22 C _R	to select the antennas.
> BASEL 0 C _R	to select all baselines to these antennas.
> STOKES '' C _R	to select all Stokes.
> REASON = 'ZOMBIE ANTENNA' C _R	to set a reason.
> OUTFGVER 1 C _R	to select the first (only) flag table.
> INP C _R	be careful with the inputs here!
> GO C _R	to run the task when ready.

4.3.2 Primary flux density calibrators

Careful measurements made with the D array of the **VLA** have shown that the Baars *et al.*² formulae for “standard” calibration sources are in error slightly, based on the assumption that the Baars’ expression for 3C295 is correct. Revised values of the coefficients have been derived by Rick Perley and Brian Butler. Task **SETJY** has these formulae built into it, giving you the option (**OPTYPE** 'CALC') of letting it calculate the fluxes for primary calibrator sources 3C48, 3C123, 3C138, 3C147, 3C196, 3C286, 3C295, and 1934-638. The default setting of **APARM**(2) = 0 will calculate the flux densities based on Perley-Butler 2017 values which cover the range 50 MHz to 50 GHz for the primary calibrators: 3C48, 3C138, 3C147, 3C286, and 3C295 plus 3C123 and 3C196, all with a number of synonyms. Other sources which will be computed, but which may not be good calibrator sources, are (one name each) J0444-2809, PictorA, 3C144 (Taurus A, Crab Nebula), 3C218 (Hydra A), 3C274 (Virgo A), 3C348 (Hercules A), 3C353, 3C380, 3C405 (Cygnus A), 3C444, and 3C461 (Cassiopeia A). The 3C name is the only one used in **SETJY** for those extra sources with a 3C name — the parenthetical remarks are here for clarity only. Most of the sources in the extra list have limits on the frequency range over which the function is valid; **SETJY** will tell you if the frequencies are out of range. Higher values of **APARM**(2) select older systems of coefficients if you need them to match previous data reductions. **SETJY** will recognize both the 3C and IAU designations (B1950 and J2000) for the standard sources. You may insert

²1977, *Astr. & Ap.*, 61, 99

your own favorite values for these sources instead (`OPTYPE = ''`) and you will have to insert values for any other gain calibrators you intend to use. Adverbs `SPECINDX` and `SPECURVE` allow you to enter spectral index information to help set the calibrator fluxes.

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

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

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

Or you can set the flux manually as shown below:

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

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

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

4.3.3 First pass of the gain calibration

4.3.3.1 Using calibrator models

It is now considered standard practice to use flux calibrator models and you are strongly encouraged to do so. As mentioned above, all the primary flux calibrators are resolved at *most* frequencies and configurations. Figure 4.1 shows the visibilities and image of the commonly used calibrator 3C48 at X-band, it is obvious this source is far from being point like. Since April 2004, source models have been shipped with `AIPS` as FITS files. Currently, models for 3C48, 3C286 and 3C137 are available for all bands except 90 cm and 3C147 at all bands except, X, C and 90 cm. Additional models are in the works, so you should always check to see what is available:

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

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

> `RUN VLAPROCS` C_R to compile the procedures.

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

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

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

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

Alternatively, you can load the model in manually using `CALRD` :

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

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

> TASK 'CALIB'; INP C _R	to select task and review inputs.
> INDI <i>n</i> ; GETN <i>m</i> C _R	to select the data set, <i>n</i> = 3 and <i>m</i> = 1 above.
> CALSOUR = 'Cala', ' ' C _R	flux calibrator for which you have a model.
> UVRANGE 0 C _R	no <i>uv</i> limits needed.
> ANTENNAS 0 C _R	antenna selection not needed.
> REFANT <i>n</i> C _R	reference antenna number — use a reliable antenna located near the center of the array.
> WEIGHTIT 1 C _R	to select $1/\sigma$ weights which may be more stable.
> IN2DI <i>o</i> ; GET2N <i>p</i> C _R	to select the model.
> NCOMP 0 C _R	to use all components.
> SOLMODE 'A&P' C _R	to do amplitude and phase solutions.
> APARM(6) 2 C _R	to print closure failures.
> MINAMPER 10 C _R	to display warning if baseline disagrees in amplitude by more than 10% from the model.
> MINPHSER 10 C _R	to display warning if baseline disagrees by more than 10° of phase from the model.
> CPARM(5) 1 C _R	to vector average amplitudes over spectral channels and then scalar average them over time before determining solution.
> FREQID 1 C _R	to use FQ number 1.
> INP CALIB C _R	to review inputs.
> GO C _R	to make the solution.

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

After running **CALIB** check the solutions for all antennas with **SNPLT** or **LISTR** (**OPTYPE**='**GAIN**'). If you have multiple primary or secondary calibrators you will have to run **CALIB** separately for each, using models where they are available and restricting the **UVRANGE** and **ANTENNAS** where they are not. You can either write into the same SN table by setting **SNVER** to a table number or to different SN tables by setting **SNVER** = 0. Then you can proceed as normal flagging and editing your data and proceed to final calibration as described in §4.5.

4.3.3.2 Flux calibration without calibrator models

We strongly encourage you to use the available models. If words alone do not convince you, we encourage you to look at Figure 4.1 which shows you the visibilities and image of 3C48 at X-band. It is rather far from a point source. At lower frequencies there are other sources in the field which have an effect on phases as well. This section used to list recommended **UVRANGE**s to use for the standard calibration sources. This is such bad practise that we have deleted those tables — use the models.

4.3. Beginning the calibration

4. Calibrating Interferometer Data

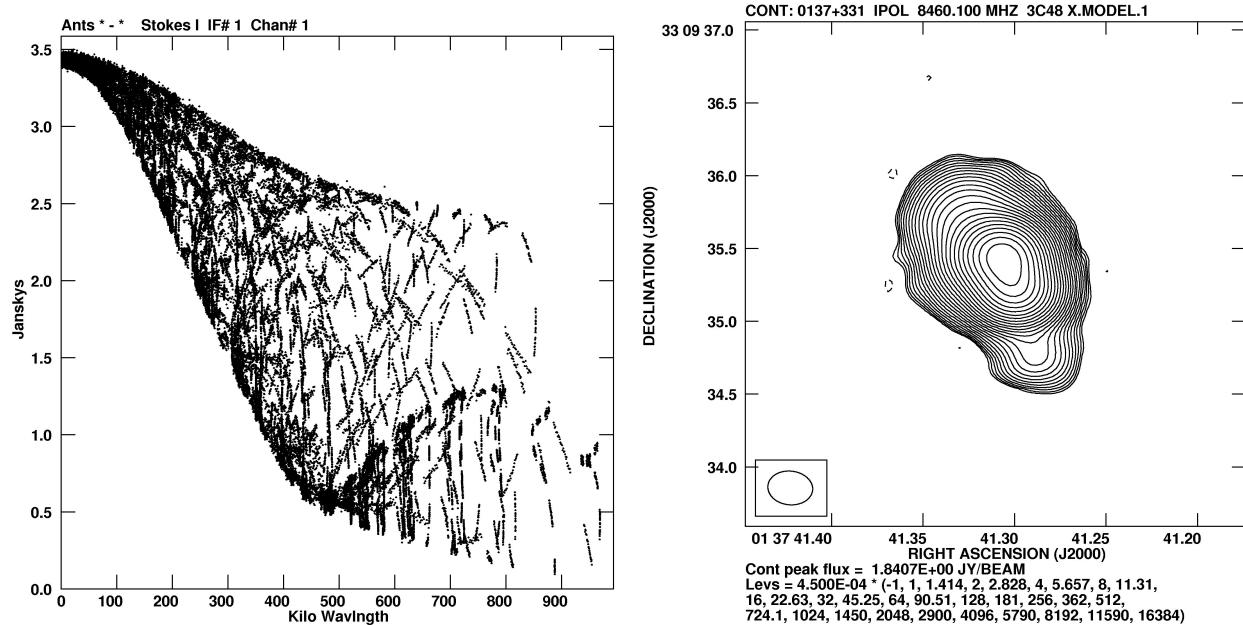


Figure 4.1: Displays of the visibilities (left) and image (right) for the fundamental calibration source 3C48. The plots were made using [UVPLT](#), [KNTR](#), and [LWPLA](#); see § 6.3.1 and § 6.3.2.1. Data from all [VLA](#) configurations including the VLBA antenna in Pie Town were used. A point source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

The values of [UVRANGE](#) for each secondary calibrator may be determined from the [VLA](#) Calibrator manual or by using [UVPLT](#) to plot the amplitudes as a function of baseline length. If your secondary calibrators are point sources over most baselines, then it may save you time to do the full calibration now. Not only will it save you, possibly, from re-running [CALIB](#) at a later time with a wider [UVRANGE](#), but it will provide information on the data quality from the longer baselines.

Once you have read in procedure [VLACALIB](#) (see § 4.3.3.1), you may use it to invoke [CALIB](#). You will have to do this once for each calibrator, unless you can use the same [UVRANGE](#) for more than one of them. Thus,

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

This procedure will first run [CALIB](#), then print any messages from [CALIB](#) about closure errors on the line

printer, and finally run `LISTR` to print the amplitudes and phases of the derived solutions. Plots of these values may be obtained using task `SNPLT`.

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

The `LISTR` outputs provided by `VLACALIB` should be examined carefully to check on the calibration; amplitudes should be consistent (both among antennas and among time stamps) and phases should vary smoothly. If you decide that the solutions are not acceptable (*e.g.*, there are no valid solutions) *and* you are creating a new SN table on each run of `CALIB`, then delete that SN table using `EXTDEST` before proceeding. The later stages of processing assume that all extant SN tables are valid. Note that re-running `CALIB` on the same SN table simply over-writes the old solutions with new ones. `CALIB` gives messages which indicate the number of valid and invalid solutions which should help you evaluate the results. If `VLACALIB` is run using the values of `MINAMPER` and `MINPHSER` shown above, it will print a list of baselines and times which show substantial “closure” errors. (If you use `CALIB` directly rather than `VLACALIB`, you may use these adverbs plus `CPARM(2-4)` to get additional reports and statistics on closure errors and `CPARM(7)` to limit reporting of individual closure errors.) It is important to remember that normal thermal noise and, at longer wavelengths, background confusion cause closure errors too. Thus, some closure error on weaker calibrators is to be expected and may be ignored. `CALIB` will ignore errors larger than `MINAMPER` and `MINPHSER` if the data weights (after application of the source model) indicate that they have significance less than `CPARM(7)` times the expected error. Interpreting closure errors is a real art, but a couple of generalizations are possible. If the same closure error shows up in both polarizations and both IFs, then you have probably got a resolved object. If one antenna dominates the closure list, especially if it is at only one IF and/or one polarization, then you have got a bad antenna. If the errors are uniformly small, distributed amongst all antennas, and not correlated between IFs or polarizations, then you have simply noise and/or background confusion. In this case, do *not* edit the data — the randomness of the “errors” nearly always averages out nicely and the solution is just fine. Large or systematic errors indicate either that the calibrator source is resolved or that there are problems with the data requiring editing. If a calibrator is being resolved, delete the bad SN table and re-run `VLACALIB` with an appropriate `UVRANGE`. One can now actually flag data based on closure errors using the `DOFLAG` option. This should be used carefully, if at all and then only if the model of the calibration source contains all of its flux. Modern versions of `CALIB` display some statistics of closure problems automatically and allow the use of “robust” solution methods.

4.4 Assessing the data quality and initial editing

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

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

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

There are three general methods of editing in *AIPS*. The “old-fashioned” route uses **LISTR** to print listings of the data on the printer or the user's terminal. The user scans these listings with his eyes and, upon finding a bad point, enters a specific flag command for the data set using **UVFLG**. While this may sound clumsy, it is in fact quite simple and by far the faster method when there are only a few problems. In a highly corrupted data set, it can use a lot of paper and may force you to run **LISTR** multiple times to pin down the exact problems. The “hands-off” route uses tasks which attempt to determine which data are bad using only modest guidance from the user. The most general of these are **RFLAG** and **FLAGR** mentioned below. The third and “modern” route uses interactive (“TV”-based) tasks to display the data in a variety of ways and to allow you to delete sections of bad data simply by pointing at them with the TV cursor. These tasks are **TVFLG** (§ 4.4.3) for all baselines and times (but only shows one IF, one Stokes, and one spectral channel at a time), **SPFLG** (§ 10.2.2) for all spectral channels, IFs, and times (but only shows one baseline and one Stokes at a time), **FTFLG** for all spectral channels, IFs, and times (but shows all baselines combined, one Stokes at a time), **EDITA** (§ 4.4.2) for editing based on **TY** (T_{ant}), **SN** or **CL** table values, **EDITR** (§ 5.5.2) for all times (but only shows a single antenna (1–11 baselines) and one channel average at a time) and **WIPER** for all types of data (but with the time, polarization, and sometimes antenna of the points not available while editing). **TVFLG** is the one used for continuum and channel-0 data from the **VLA**, while **FTFLG** is only used to check for channel-dependent interference. **SPFLG** is very useful for spectral-line editing in smaller arrays, such as the Australia Telescope and the VLBA, but is tedious for arrays like the **VLA**. Nonetheless, it may be necessary for the modern **EVLA**. (The redundancy in the spectral domain on calibrator sources helps the eyes to locate bad data.) **EDITR** is more useful for small arrays such as those common in VLBI experiments. **EDITA** has been found to be remarkably effective using **VLA** system temperature tables. All four tasks have the advantage of being very specific in displaying the bad data. Multiple executions should not be required. However, they may require you to look at each IF, Stokes, channel (or baseline) separately (unless you make certain broad assumptions); **EDITA** and **EDITR** do allow you to look at all polarizations and/or IFs at once if you want. They all require you to develop special skills since they offer so many options and operations with the TV cursor (mouse these days). A couple of general statements can be made

- For highly corrupted data (say with considerable RF interference, significant cross-talk between antennas, or erratic antennas) **TVFLG** is definitely preferred. It gives an overall view of the data which is far superior to that given by **LISTR**. **RFI** and similar problems are more troublesome at lower frequencies, so **TVFLG** is probably preferred for L, P, and “4” bands.

- Most **VLA** data at higher frequencies are of good quality and the flexibility of **TVFLG** is not needed. In such cases, **LISTR** with **OPCODE** = 'MATX' can find scans with erroneous points efficiently.
- The displays given by **TVFLG** and, to a lesser extent, **LISTR** in its MATX mode are less useful when there are only a few baselines. Thus, for arrays smaller than the **VLA**, users may wish to use **SPFLG** on spectral-line data sets and **EDITR** on continuum data sets.
- A reasonable strategy to use is to run **LISTR** first. If there are only a few questionable points, use **LISTR** and **UVFLG**, otherwise switch to an interactive task, such as **EDITA** followed by **TVFLG**.
- Task **FLAGR** is a new, somewhat experimental task to measure the rms in the data on either a baseline or an antenna basis and then delete seriously discrepant points and times when many antennas/correlators are questionable. It also clips amplitudes and weights which are outside specified normal ranges. Task **FINDR** reports the rmses and excessive values to assist in running **FLAGR**.
- Task **RFLAG** is a new task which flags data on the belief that **RFI** is highly variable in time and/or frequency. It can do plots showing the time and/or frequency statistics as a function of spectral channel and recommends flagging levels. These, or user-chosen values, may be applied to the data to produce large flag tables; see § E.5.
- Task **CLIP** makes entries in a flag table, applying calibration and then testing amplitudes for reasonableness on a source-by-source basis. It can be very useful for large data sets, but does not show you the bad data to evaluate yourself. **ACLIP** does the same operation on auto-correlation data.
- Task **DEFLG** makes entries in a flag table whenever the phases are too variable as measured by too low a ratio of vector-averaged to scalar-averaged amplitudes. This may be useful when applied to the calibrator source in phase-referencing observations and for other data at the highest and lowest frequencies which are affected by atmospheric and ionospheric phase variability.
- Task **SNFLG** makes entries in a flag table whenever the phase solutions in an **SN** or **CL** table change excessively between samples on a baseline basis. It can also flag data if the amplitude solutions differ from their mean excessively. In 31DEC12 it can instead flag data when the amplitude and/or phase solutions or solution weights are outside user-specified ranges.
- Task **WIPER** makes entries in a flag table for all data samples wiped from a **UVPLT**-like display of any *uv* data set parameter versus any other parameter. The source, Stokes, IF, time, etc. of the points are not known during the interactive editing phase, but some baseline information is displayed. It can plot and edit any choice of **STOKES** in one execution and can flag/unflag by baseline.
- Task **WETHR** makes entries in a flag table whenever various weather parameters exceed specified limits. **WETHR** also plots the weather (WX) table contents.
- Task **VPFLG** flags all correlators in a sample whenever one is flagged. Observations of sources with circular polarization (Stokes V) require this operation to correct the flagging done on-line (which flags only known bad correlators).
- Task **FGPLT** plots the times of selected flag-table entries to provide you information on what these powerful tasks have done.

Many of the above tasks produce large flag tables. The task **REFLG** attempts to compress such tables, sometimes quite remarkably. Task **FGDIF** may be used to confirm that the flag tables before and after flag the same things. **FGCNT** in 31DEC16 may also help confirm this.

4.4.1 Editing with LISTR and UVFLG

Data may be flagged using task **UVFLG** based on listings from **LISTR**. To print out the scalar-averaged raw amplitude data for the calibrators, and their *rms* values, once per scan in a matrix format, the following inputs are suggested:

> TASK 'LISTR' ; INP C_R	to review the inputs needed.
> INDI <i>n</i> ; GETN <i>m</i> C_R	to select the data set, <i>n</i> = 3 and <i>m</i> = 1 above.
> SOURCES '' ; CALCODE ** C_R	to select calibrators.
> TIMER 0 C_R	to select all times.
> ANTENNAS 0 C_R	to list data for all antennas.
> OPTYPE 'MATX' C_R	to select matrix listing format.
> DOCRT FALSE C_R	to route the output to printer, not terminal.
> DPARM 3 , 1 , 0 C_R	amplitude and <i>rms</i> , scalar scan averaging.
> BIF 1; EIF 0 C_R	to select all IFs, LISTR will list IFs separately.
> FREQID 1 C_R	to select FQ number 1 (note that FQ numbers must also be done separately).
> INP C_R	to review the inputs.
> GO C_R	to run the program when inputs set correctly.

For unresolved calibrators, the **VLA** on-line gain settings normally produce roughly the same values in all rows and columns within each matrix. At L, C, X, and U bands, these values should be approximately 0.1 of the expected source flux densities. At P band, the factor is about 0.01. The factors for other bands are unspecified. Any rows or columns with consistently high or low values in either the amplitude or the *rms* matrices should be noted, as they probably indicate flaky antennas. In particular, you should look for

- In the amp-scalar averages, look for *dead* antennas, which are easily visible as rows or columns with small numbers. Rows or columns that differ by factors of two or so from the others are generally fine. Such deviations mean only that the on-line gains were not set entirely correctly.
- In the *rms* listings, look for discrepant high values. Almost all problems are antenna based and will be seen as a row or column. Factors of 2 too high are normally okay, while factors of 5 high are almost certainly indicative of serious trouble.

The next step is to locate the bad data more precisely. Suppose that you have found a bad row for antenna 3 in right circular polarization in IF 2 between times (*d1*, *h1*, *m1*, *s1*) and (*d2*, *h2*, *m2*, *s2*). You might then rerun **LISTR** with the following new inputs:

> SOURCES '' C_R	to select all sources.
> TIMER <i>d1 h1 m1 s1 d2 h2 m2 s2</i> C_R	to select by time range.
> ANTENNAS 1 , 2 , 3 C_R	to list data for antenna 3 with two "control" antennas.
> BASEL 1 , 2 , 3 C_R	to list all baselines with these three antennas.
> OPTYPE 'LIST' C_R	to select column listing format.
> DOCRT 1 C_R	to route the output to terminal at its width.
> DPARM = 0 C_R	amplitude only, no averaging.
> STOKES 'RR' C_R	to select right circular.
> BIF 2 C_R	to specify the "BD" IFs.
> FLAGVER 1 C_R	to choose flag table 1.
> GO C_R	to run the program.

This produces a column listing on your terminal of the amplitude for baselines 1–2, 1–3 and 2–3 at every time stamp between the specified start and stop times. The '1–2' column provides a control for comparison with the two columns containing the suspicious antenna.

Note that “amp-scalar” averaging ignores phase entirely and is therefore not useful on weak sources, nor can it find jumps or other problems with the phases. To examine the data in a phase-sensitive way, repeat the above process, but set **DPARM(2) = 0** rather than 1. Bad phases will show up as reduced amplitudes and increased *rms*’s.

Once bad data have been identified, they can be expunged using **UVFLG**. For example, if antenna 3 RR was bad for the full interval shown above, it could be deleted with

> TASK 'UVFLG' ; INP C_R	to select the editor and check its inputs.
> TIMER d1 h1 m1 s1 d2 h2 m2 s2 C_R	to select by time range.
> BIF 2 ; EIF = BIF C_R	to specify the “BD” IFs.
> BCHAN 0 ; ECHAN 0 C_R	to flag all channels.
> FREQID 1 C_R	to flag only the present FQ number.
> ANTEN 3 , 0 C_R	to select antenna 3.
> BASEL 0 C_R	to select all baselines to antenna 3.
> STOKES 'RR' C_R	to select only the RR Stokes (LL was found to be okay in this example).
> REASON = 'BAD RMS WHOLE SCAN' C_R	to set a reason.
> OUTFGVER 1 C_R	to select the first (only) flag table.
> INP C_R	be careful with the inputs here!
> GO C_R	to run the task when ready.

Continue the process until you have looked at all parts of the data set that seemed anomalous in the first matrix listing, then rerun that listing to be sure that the flagging has cleaned up the data set sufficiently. If there are lots of bad data, you may find that you have missed a few on the first pass. If you change your mind about a flagging entry, you can use **UVFLG** with **OPCODE = 'UFLG'** to remove entries from the flag table. All adverbs of **UVFLG** are used when removing entries, so you may use **REASON** along with the channel, IF, source, et al. adverbs to select the entries to be removed. **OPCODES 'REAS'** and **'WILD'** may be used to undo an entry solely based on the **REASON**. If the table becomes hopelessly messed up, use **EXTDEST** to delete the flag table and start over or use a higher numbered flag table. The contents of the flag table may be examined at any time with the general task **PRTAB** and entries in it may also be removed with **TABED** and/or **TAFLG**. Two flag tables can be merged using **TAPPE**.

4.4.2 Editing with EDITA

The task **EDITA** uses the graphics planes on the *AIPS* TV display to plot data from tables and to offer options for editing (deleting, flagging) the associated *uv* data. Only the TY (system temperature), SN (solution), and CL (calibration) tables may be used. **EVLA** users will not have a TY table, but they may have an SY (SysPower) table which may be used instead. We recommend using **EDITA** with the TY or SY tables to do the initial editing of **VLA** and **EVLA** data sets, probably before running the programs described in § 4.3. For accuracy in evaluating and flagging your data, it is a good idea to have the TY table filled with the same interval as the data themselves; see § 4.1.1. Try:

> TASK 'EDITA ; INP C_R	to review the inputs needed.
> INDI n ; GETN m C_R	to select the data set, <i>n</i> = 3 and <i>m</i> = 1 above.
> INEXT 'TY' C_R	to use the system temperature table.
> INVERS 0 C_R	to use the highest numbered table, usually 1.
> TIMER 0 C_R	to select all times.
> FREQID 3 C_R	Select FQ entry 3.
> BIF 1 ; EIF 0 C_R	to specify all IFs; you can then toggle between them interactively and even display all at once.

> ANTENNAS 0 C _R	to display data for all antennas.
> ANTUSE 1 , 2 , 3 , 4 , 5 , 6 , 7 C _R	to display initially the first 7 antennæ, editing antenna 1. Others may be selected interactively.
> FLAGVER 1 C _R	to use flag (FG) table 1.
> OUTFGVER 0 C _R	to create a new flag table with the flags from FG table 1 plus the new flags.
> SOLINT 0 C _R	to avoid averaging any samples.
> DOHIST FALSE C _R	to omit recording the flagging in the history file.
> DOTWO TRUE C _R	to view a 2 nd observable for comparison
> CROWDED TRUE ; DO3COL TRUE C _R	to allow plots with all polarizations and/or IFs simultaneously, using color to differentiate the polarizations and IFs.
> INP C _R	to review the inputs.
> GO C _R	to run the program when inputs set correctly.

If you make multiple runs of **EDITA**, it is important to make sure that the flagging table entries are all in one version of the FG table. The easiest way to ensure this is to should set **FLAGVER** and **OUTFGVER** to 0 and keep it that way for all runs of **EDITA**. This may create an excessive number of flag tables, but unwanted ones may be deleted with **EXTDEST**. If you make a mistake two flag tables may be merged with the task **TAPPE**. A sample display from **EDITA** is shown on the next page.

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

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

Having flagged all obviously bad points, select **SWITCH ALL IF**, **SWITCH ALL TIME**, **SWITCH ALL ANT**, and **SWITCH ALL POL** so that the next flag command(s) apply to all of the data. (Decide whether the flags should apply only to the source(s) displayed or to all sources and set **SWITCH ALL SOURC** appropriately.) Set the

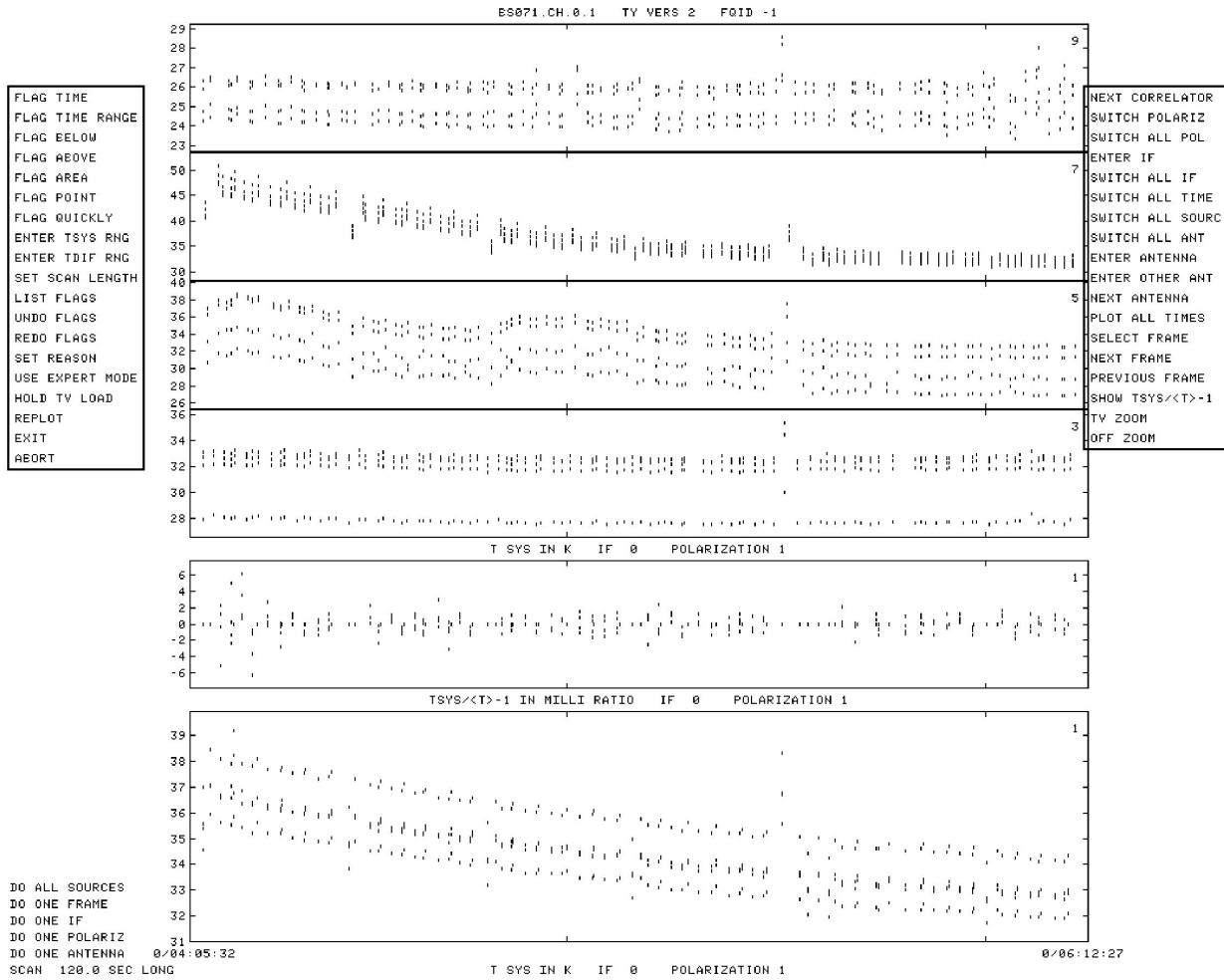


Figure 4.2: A display of a sample TV screen from *EDITA*, made using the *AIPS* task *TVCPS* to produce a negative black-and-white display. System temperatures are being used to edit VLBA data. The *EDITA* menu (in the boxes), the status lines (at the bottom), the editing area (bottom) of a portion of the data from the selected antenna (1), the subsidiary plots of data from selected secondary antennæ (3, 5, 7, 9), the edit tool (bar or box), and the edit location values are displayed in different graphics planes which normally appear in different colors. In this example, with *CROWDED=TRUE*, four IFs but only one polarization are displayed and may be edited simultaneously. Both polarizations can be displayed together along with either one or all four IFs. Color may be used to differentiate polarizations and IFs.

SCAN LENGTH long enough to include the shorter of the full scan and about 12 samples. Then display the difference between the current sample and the running mean by selecting **SHOW TSYS** – <T>. Use **FLAG ABOVE** and **FLAG BELOW** to flag all samples more than a few sigma away from the local mean. Finally, apply your flagging to your *uv* data set by selecting **EXIT**.

At this point, return to § 4.3.1 to run **QUACK** followed by the first pass of the gain calibration. Then run **TVFLG** below with **DOCAL TRUE** so that the data will be displayed on the same flux scale for all baselines.

4.4.3 Editing with TVFLG

If your data are seriously corrupted, contain numerous baselines, and you like video games, **TVFLG** is the visibility editor of choice. The following discussion assumes that you have read § 2.3.2 and are familiar with using the *AIPS* TV display. The following inputs are suggested:

- > **TASK 'TVFLG'; INP C_R** to review the inputs needed.
- > **INDI n ; GETN m C_R** to select the data set, $n = 3$ and $m = 1$ above.
- > **SOURCES '' C_R** to select all sources.
- > **TIMER 0 C_R** to select all times.
- > **STOKES 'RRLL' C_R** to select both right and left circular polarizations; you can then toggle between RR and LL interactively.
- > **FREQID 3 C_R** Select FQ entry 3.
- > **BIF 1 ; EIF 2 C_R** to specify both **VLA** IFs; you can then toggle between the two interactively.
- > **ANTENNAS 0 C_R** to display data for all antennas.
- > **BASELINE 0 C_R** to display data for all baselines.
- > **DOCALIB 1 C_R** to apply initial calibration to the data.
- > **FLAGVER 1 C_R** to use flag (FG) table 1.
- > **OUTFGVER 0 C_R** to create a new flag table with the flags from FG table 1 plus the new flags.
- > **DPARM = 0 C_R** to use default initial displays and normal baseline ordering.
- > **DPARM(6) = 30 C_R** to declare that the input data are 30-second averages, or to have the data averaged to 30 seconds. Note that one can interactively increase the time averaging, in integer units of **DPARM(6)**, after the master grid is created.
- > **DPARM(5) = 10 C_R** to expand the flagging time ranges by 10 seconds in each direction. The times in the master grid are average times and may not encompass the times of the samples entering the average without this expansion.
- > **DOCAT 1 C_R** to save the master grid file.
- > **INP C_R** to review the inputs.
- > **GO C_R** to run the program when inputs set correctly.

If you make multiple runs of **TVFLG**, it is important to make sure that the flagging table entries are all in one version of the FG table. The easiest way to ensure this is to set **FLAGVER** and **OUTFGVER** to 0 and keep it that way for all runs of **TVFLG**. If you make a mistake, two flag tables may be merged with the task **TAPPE**.

TVFLG begins by constructing a “master grid” file of all included data. This can be a long process if you include lots of data at once, although in 31DEC13 a new, faster, large-memory method of gridding is usually used. It is probably better to use the channel selection (including averaging channels with **NCHAV**), IF selection, source selection, and time range selection adverbs to build rather smaller master grid files and

then to run **TVFLG** multiple times. It will work with all data included, allowing you to select interactively which data to edit at any one moment and allowing you to resume the editing as often as you like. But certain operations (such as undoing flags) have to read and process the entire grid, and will be slow if that grid is large. The master grid file is always catalogued (on **IN2DISK** with class **TVFLGR**), but is saved at the end of your session only if you set **DOCAT** = 1 (actually > 0) before starting the task. To resume **TVFLG** with a pre-existing master grid file, set the adverb **IN2SEQ** (and **IN2DISK**) to point at it. When resuming in this way, **TVFLG** ignores all of its data selection adverbs since they might result in a different master grid than the one it is going to use. If you wish to change any of the data selection parameters, e.g., channels, IFs, sources, times, or time averaging, then you must use a new master grid.

Kept with the master grid file is a special file of **TVFLG** flagging commands. This file is updated as soon as you enter a new flagging command, making the master grid and your long editing time virtually proof from power failures and other abrupt program terminations. These flagging commands are not entered into your actual *uv* data set's flagging (FG) table until you exit from **TVFLG** and tell it to do so. During editing, **TVFLG** does not delete data from its master grid; it just marks the flagged data so that they will not be displayed. This allows you to undo editing as needed during your **TVFLG** session(s). When the flags are transferred to the main *uv* data set, however, the flagged data in the master grid are fully deleted since undoing the flags at that point has no further meaning. When you are done with a master grid file, be sure to delete it (with **ZAP**) since it is likely to occupy a significant amount of disk.

TVFLG keeps track of the source name associated with each row of data. When averaging to build the master grid and to build the displayed grids, **TVFLG** will not average data from different sources and will inform you that it has omitted data if it has had to do so for this reason. For multi-source files, the source name is displayed during the **CURVALUE**-like sections. However, the flagging table is prepared to flag *all* sources for the specified antennas, times, etc. or just the displayed source. If you are flagging two calibrator scans, you may wish to do all sources in between as well. Use the **SWITCH SOURCE FLAG** interactive option to make your selection before you create flagging commands. Similarly, you will need to decide whether flagging commands that you are about to prepare apply only to the displayed channel and/or IF, or to all possible channels and/or IFs. In particular, spectral-line observers often use **TVFLG** on the pseudo-continuum "channel-0" data set, but want the resulting flags to apply to all spectral channels when copied to the spectral-line data set. They should be careful to select all channels before generating any flagging commands. Each flagging command generated is applied to a list of Stokes parameters, which *does not have to include* the Stokes currently being displayed. When you begin **TVFLG** and whenever you switch displayed Stokes, you should use the **ENTER STOKES FLAG** option to select which Stokes are to be flagged by subsequent flagging commands.

If you get some of this wrong, you can use the **UNDO FLAGS** option in **TVFLG** if the flags have not yet been applied to the *uv* data set. Or you can use tasks **UVFLG**, **TABED** or **TAFLG** to correct errors written into the FG table of your multi-source *uv* data set. Flag tables are now used with both single- and multi-source data sets.

TVFLG displays the data, for a single IF, channel (average), and Stokes, as a grey-scale display with time increasing up the screen and baseline number increasing to the right. Thus baselines for the **VLA** run from left to right as 1-1, 1-2, 1-3, ..., 2-2, 2-3, ..., 27-27, 27-28, and 28-28. An input parameter (**DPARM(3)** = 1 allows you to create a master grid and display baselines both as, say 1-2 and 2-1. An interactive (switchable) option allows you to order the baselines from shortest to longest (ignoring projection effects) along the horizontal axis.

The interactive session is driven by a menu which is displayed on a graphics overlay of the TV display. An example of this full display is shown on the next page. Move the cursor to the desired operation (noting that the currently selected one is highlighted in a different color on many TVs) and press button A, B, or C to select the desired operation; pressing button D produces on-line help for the selected operation. The first (left-most column) of choices is:

OFFZOOM

turn off any zoom magnification

OFFTRANS	turn off any black & white enhancement
OFFCOLOR	turn off any pseudo-coloring
TVFIDDLE	interactive zoom, black & and white enhancement, and pseudo-color contours as in AIPS
TVTRANSF	black & white enhancement as in AIPS
TVPSEUDO	many pseudo-colorings as in AIPS
DO WEDGE ?	switches choice of displaying a step wedge
LOAD xxxx	switch TV load transfer function to xxxx
LIST FLAGS	list selected range of flag commands
UNDO FLAGS	remove flags by number from the FC table master grid
REDO FLAGS	re-apply all remaining flags to master grid
SET REASON	set reason to be attached to flagging commands
DO LABEL ?	turn axis labeling on and off

Note: when a flag is undone, all cells in the master grid which were first flagged by that command are restored to use. Flag commands done after the one that was undone may also, however, have applied to some of those cells. To check this and correct any improperly un-flagged pixels, use the REDO FLAGS option. This option even re-does CLIP operations! After an UNDO or REDO FLAGS operation, the TV is automatically re-loaded if needed. Note that the UNDO operation is one that reads and writes the full master grid.

The load to the TV for all non-phase displays may be done with all standard transfer functions: LINear, LOG, SQRT, LOG2 (more extreme log). The menu shows the next one in the list (xxxx above) through which you may cycle. The TV is reloaded immediately when a new transfer function is selected.

Column 2 offers type-in controls of the TV display and controls of which data are to be flagged. In general, the master grid will be too large to display on the TV screen in its entirety. The program begins by loading every n^{th} baseline and time smoothing by m time intervals in order to fit the full image on the screen. However, you may select a sub-window in order to see the data in more detail. You may also control the range of intensities displayed (like the adverb PIXRANGE in TVLOAD inside AIPS). The averaging time to smooth the data for the TV display may be chosen, as may the averaging time for the “scan average” used in some of the displays. Which correlators are to be flagged by the next flagging command may be typed in. All of the standard Stokes values, plus any 4-bit mask may be entered. The spectral channel and IF may be typed in. Flagging may be done only for the current channel and IF and source, or it may be done for all channels and/or IFs and/or sources. Note that these controls affect the next LOADs to the TV or the flagging commands prepared after the parameter is changed. When the menu of options is displayed at the top of the TV, the current selections are shown along the bottom. If some will change on the next load, they are shown with a trailing asterisk. Column 2 contains

ENTER BLC	Type in a bottom left corner pixel number on the terminal
ENTER TRC	Type in a top right corner pixel number on the terminal
ENTER AMP PIXRANGE	Type in the intensity range to be used for loading amplitude images to the TV
ENTER PHS PIXRANGE	Type in the phase range to be used for loading phase images to the TV
ENTER RMS PIXRANGE	Type in the intensity range to be used for loading images of the rms to the TV
ENTER R/M PIXRANGE	Type in the value range to be used for loading rms/mean images to the TV
ENTER SMOOTH TIME	Type in the time smoothing (averaging) length in units of the master grid cell size
ENTER SCAN TIME	Type in the time averaging length for the “scan average” in units of the master grid cell size

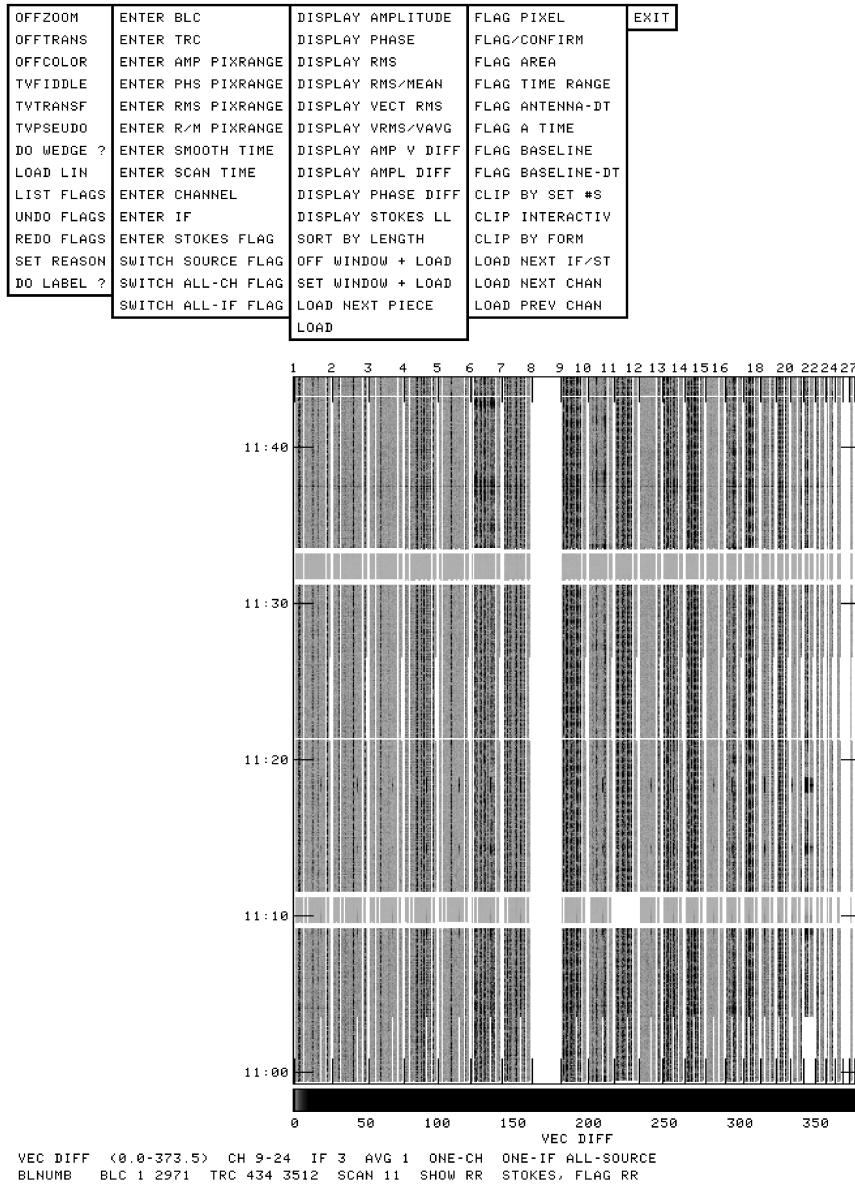


Figure 4.3: A display of a sample TV screen from **TVFLG**, made using the *AIPS* task **TVCPS** to produce a negative black-and-white display. The **TVFLG** menu (in the boxes) is displayed in a graphics plane which is normally colored light green. The status line(s) at the bottom and optional axis labels are displayed in a graphics plane that is normally cyan in color. The data are grey scales in a TV memory and may be enhanced in black-and-white or pseudo-colored. The particular display chosen is the amplitude of the vector difference between the sample and a running vector average of samples surrounding it. This particular parameter is sensitive to both phase and amplitude problems and may save you the extra time of looking at phase and amplitude separately. It requires that there be data to average, but does not blur the flagging by the averaging interval (as the RMS method does). The visibility data are from the JVLA. All baselines are shown once only in baseline number order. Antenna 8 and 27 are missing for all times, while antennas 11 and 22 are missing for some times in the piece displayed here. The displayed data are the RR Stokes samples and have been windowed to exclude some times. Flag commands generated at the moment illustrated will flag all source names, one spectral channel (actually channels 9-24 averaged here), one IF, and Stokes RR only (from a 2-Stokes data set). The step wedge and labeling options have been selected.

ENTER CHANNEL	Type in the desired spectral channel number using the terminal
ENTER IF	Type in, on the terminal, the desired IF number
ENTER STOKES FLAG	To type in the 4-character string which will control which correlators (polarizations) are flagged. Note: this will apply only to subsequent flagging commands. It should be changed whenever a different Stokes is displayed.
SWITCH SOURCE FLAG	To switch between having all sources flagged by the current flag commands and having only those sources included in this execution of TVFLG flagged. The former is desirable when a time range encompasses all of 2 calibrator scans.
SWITCH ALL-CH FLAG	To reverse the flag all channel status; applies to subsequent flag commands
SWITCH ALL-IF FLAG	To cycle the flag all IFs status; applies to subsequent flag commands

The all-channel flag remains true if the input data set has only one channel and the all-IF flag remains true if the input data set has no more than one IF. If there are more than 2 IFs, the SWITCH ALL-IF FLAG cycles between flagging one IF, flagging a range of IFs, and flagging all IFs. When going to the range of IFs, it will ask you to enter the desired range.

An extra word should be said about the “scan average” to which reference was made above. This is used solely for displaying the difference of the data at time T and the average of the data at times near T . This average is computed with a “rolling buffer.” Thus, for a scan average time of 30 seconds and data at 10-second intervals, the average for a set of 7 points is as follows:

time	average of times
00	00 10 20
10	00 10 20
20	10 20 30
30	20 30 40
40	30 40 50
50	40 50 60
60	40 50 60

The third column of options is used to control which data are displayed and to cause the TV display to be updated. The master grid must be converted from complex to amplitude, phase, the rms of the amplitude, or the rms divided by the mean of the amplitude for display. It may also be converted to the amplitude of the vector difference between the current observation and the “scan average” as defined above or the absolute value of the difference in amplitude with the scalar-average amplitude or the absolute value of the difference in phase with the vector scan average. Furthermore, the baselines may be reordered in the TV display by their length rather than their numerical position. This column has the options:

DISPLAY AMPLITUDE	To display amplitudes on the TV
DISPLAY PHASE	To display phases on the TV
DISPLAY RMS	To display amplitude rms on the TV
DISPLAY RMS/MEAN	To display amplitude rms/mean on the TV
DISPLAY VECT RMS	To display vector amplitude rms on the TV
DISPLAY VRMS/VAVG	To display vector amplitude rms/mean on the TV
DISPLAY AMP V DIFF	To display the amplitude of the difference between the data and a running (vector) “scan average”
DISPLAY AMPL DIFF	To display the abs(difference) of the amplitude of the data and a running scalar average of the amplitudes in the “scan”

DISPLAY PHASE DIFF	To display the abs(difference) of the phase of the data and the phase of a running (vector) “scan average”
DISPLAY STOKES <i>xx</i>	To switch to Stokes type <i>xx</i> (where <i>xx</i> can be RR, LL, RL, LR, etc. as chosen by the STOKES adverb).
SORT BY <i>xxxxxxxx</i>	To switch to a display with the <i>x</i> axis (baseline) sorted by ordered by LENGTH or by BASELINE number
OFF WINDOW + LOAD	Reset the window to the full image and reload the TV
SET WINDOW + LOAD	Interactive window setting (like TVWINDOW) followed by reloading the TV
LOAD LAST PIECE	Reload the TV with the previous piece of the total time range.
LOAD NEXT PIECE	Reload the TV with the next piece of the total time range.
LOAD	Reload TV with the current parameters

SET WINDOW + LOAD is “smarter” than **TVWINDOW** and will not let you set a window larger than the basic image. Therefore, if you wish to include all pixels on some axis, move the TV cursor outside the image in that direction. The selected window will be shown. When there are more times than will fit on the TV screen at the current smoothing (averaging) time, the task divides the data up into overlapping time-range “pieces.” When it has done so, the LOAD LAST PIECE and LOAD NEXT PIECE menu items will appear. This lets you view one piece after the next, rotating through all pieces, to edit each time interval at full resolution. Note that a **FLAG BASELINE** will flag that baseline through all pieces.

The fourth column is used to select the type of flagging to be done. During flagging, a TV graphics plane is used to display the current pixel much like **CURVALUE** in AIPS. Buttons A and B do the flagging (except A switches corners for the area and time-range modes). Button C also does the flagging, but the program then returns to the main menu rather than prompting for more flagging selections. Button D exits back to the menu without doing any additional flagging. Another graphics plane is used to show the current area/time/baseline being flagged. All flagging commands can create zero, one, two, or more entries in the flagging list; hit button D at any time. There are also two clipping modes, an interactive one and one in which the user enters the clip limits from the terminal. In both, the current image computed for the TV (with user-set windows and data type, but not any other windows or alternate pixels etc. required to fit the image on the TV) is examined for pixels which fall outside the allowed intensity range. Flagging commands are prepared and the master file blanked for all such pixels. In the interactive mode, buttons A and B switch between setting the lower and upper clip limits, button C causes the clipping to occur followed by a return to the main menu, and button D exits to the menu with no flagging. The options are

FLAG PIXEL	To flag single pixels
FLAG/CONFIRM	To flag single pixels, but request a yes or no on the terminal before proceeding
FLAG AREA	To flag a rectangular area in baseline-time
FLAG TIME RANGE	To flag all baselines for a range of times
FLAG ANTENNA-DT	To flag all baselines to a specific antenna for a range of times
FLAG A TIME	To flag all baselines for a specific time
FLAG BASELINE	To flag all times for a specific baseline
FLAG BASELINE-DT	To flag a time range for a specific baseline
CLIP BY SET #S	To enter from the terminal a clipping range for the current mode and then clip high and low samples
CLIP INTERACTIV	To enter with the cursor and LUTs a clipping range for the current mode and then clip data outside the range.
CLIP BY FORM	To clip selected channels/IFs using the “method” and clipping range of some previous clip operation
LOAD NEXT IF/ST	Load TV with the next IF or polarization.

LOAD NEXT CHAN	To load the next spectral channel to the TV with current parameters
LOAD PREV CHAN	To load the previous spectral channel to the TV with current parameters

The **CLIP BY FORM** operation allows you to apply a clipping method already used on one channel/IF to other channels and/or IFs. It asks for a command number (use **LIST FLAGS** to find it) and applies its display type (amp, phase, rms, rms/mean, differences), averaging and scan intervals and clip levels to a range of channels, IFs and Stokes (as entered from the terminal). To terminate the operation, doing nothing, enter a letter instead of one of the requested channel or IF numbers. To omit a Stokes, reply, if requested for a flag pattern, with a blank line. You may watch the operation being carried out on the TV as it proceeds.

The right-most column has only the option:

EXIT	Resume AIPS and, optionally, enter the flags in the data
-------------	--

Before the flags are entered in the data, **TVFLG** asks you whether or not you actually wish to do this. You must respond yes or no. Note that, if the master grid is to remain cataloged, there is no need to enter the flagging commands every time you decide to exit the program for a while. In fact, if you do not enter the commands, you can still undo them later, giving you a reason not to enter them in the main *uv* data set too hastily.

The two most useful data modes for editing are probably amplitude and amplitude of the vector difference. The former is useful for spotting bad data over longer time intervals, such as whole scans. The latter is excellent for detecting short excursions from the norm. For editing uncalibrated data, rms of two time intervals is useful, but the rms modes require data to be averaged (inside **TVFLG**) and therefore reduce the time resolution accuracy of the flagging. If you edit by phase, consider using the pseudo-coloration scheme that is circular in color (option **TVPSEUDO** followed by button B) since your phases are also circular.

Using **TVFLG** on a workstation requires you to plan the real estate of your screen. We suggest that you place your message server window and your input window side-by-side at the bottom of the screen. Then put the TV window above them, occupying the upper 70–90% of the screen area. (Use your window manager's tools to move and stretch the TV window to fill this area.) Instructions and informative, warning and error messages will appear in the message server window. Prompts for data entry (and your data entry) appear in the input window. Remember to move the workstation cursor into the input window to enter data (such as IF, channel, antenna numbers, and the like) and then to move the cursor back into the TV area to select options, mark regions to be flagged, adjust enhancements, and so on.

4.4.4 Baseline corrections

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

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

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

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

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

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

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

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

- > [TASK 'VLANT' C_R](#) to get the correct data set. Note that you don't have to keep doing this unless you switch between different input data files.
- > [INDISK m ; GETN n C_R](#) to choose FQ 1.
- > [SUBARRAY x C_R](#) to choose the antenna table to correct.
- > [GAINVER 1 C_R](#) to choose the correct version of the CL table to read. A new one will be produced.
- > [GO C_R](#) to run [VLANT](#).

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

- > [TASK 'CLCOR' C_R](#) to get the correct data set. Note that you don't have to keep doing this unless you switch between different input data files.
- > [INDISK m ; GETN n C_R](#) to do all sources, all Stokes, and all IFs.
- > [SUBARRAY x C_R](#) to choose the correct sub-array.
- > [OPCODE 'ANTP' C_R](#) to select the antenna position correction mode.
- > [GAINVER 1 C_R](#) to choose the correct version of the CL table to read.
- > [GAINUSE 0 C_R](#) to have [CLCOR](#) create a new table.
- > [ANTENNA k C_R](#) to select antenna.
- > [CLCORPRM Δb_x,Δb_y,Δb_z,0,0,0,1 C_R](#) to add the appropriate antenna corrections in meters; the 1 in [CLCORPRM](#)(7) indicates [VLA](#) phase conventions rather than VLB conventions.
- > [GO C_R](#) to run [CLCOR](#).

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

0 (FOR higheST VERSION). Note too that **CLCOR** and **VLANT** change the antenna file for the changed antenna location(s). Therefore, it is wise to save the AN table before running **CLCOR** or **VLANT** by running **TASAV**.

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

4.5 Antenna-based complex gain solutions

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

4.5.1 Bootstrapping secondary flux-density calibrators

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

To use **GETJY**:

> TASK 'GETJY' ; INP C _R	
> SOURCES 'cal1', 'cal2', 'cal3' ... C _R	to select secondary flux calibrators.
> CALSOU '3C286', '' C _R	to specify primary flux calibrator(s).
> CALCODE '' C _R	to use all calibrator codes.
> BIF 1 ; EIF 2 C _R	to do both IFs.
> FREQID 1 C _R	to use FQ number 1.
> ANTENNAS 0 C _R	to include solutions for all antennas.
> TIMERANG 0 C _R	to include all times.
> SNVER 0 C _R	to use all SN tables.
> INP C _R	to review inputs.
> GO C _R	to run the task when the inputs are okay.

GETJY will give a list of the derived flux densities and estimates of their uncertainties. These are now found by "robust" methods and additional information about numbers of aberrant solutions are given. If any of the uncertainties are large, then reexamine the SN tables as described above and re-run **CALIB** and/or **GETJY** as necessary. Multiple executions of **GETJY** will not cause problems as previous solutions for the unknown flux densities are simply overwritten. You may wish to run the task **SOUISP** to determine the spectral indices of your calibrators from their fluxes in the SU table. You can even replace the values in the SU table with the curve fit by **SOUISP** and, in 31DEC15 correct the gains in one or more SN tables with the newly determined fluxes. These spectral index parameters may be useful in running **BPASS** and **PCAL**. However, **BPASS** knows the flux coefficients for all standard calibration sources and can fit the spectral index of other calibration sources from the SU table.

4.5.2 Full calibration

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

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

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

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

- > **INDI** n ; **GETN** m C_R to select the data set, $n = 3$ and $m = 1$ above.
- > **CALSOEUR** = 'aaaa' , 'xxxx' C_R to name two calibration sources using the same **UVRANGE**.
- > **UVRANGE** uvmin uvmax C_R *uv* limits, if any, in kiloλ.
- > **ANTENNAS** list of antennas C_R antennas to use for the solutions, see discussion above.
- > **REFANT** n C_R reference antenna number.
- > **MINAMPER** 10 C_R display warning if baseline disagrees in amplitude by more than 10% from the model.
- > **MINPHSER** 10 C_R display warning if baseline disagrees by more than 10° of phase from the model.
- > **DOPRINT** -1 C_R to dispense with all the print out this time.
- > **FREQID** 1 C_R use FQ number 1.
- > **INP VLACALIB** C_R to review inputs.
- > **VLACALIB** C_R to make the solution and print results.

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

- > **CALSOEUR** = 'cal1' , 'cal2' , 'cal3' C_R to name secondary flux calibrator(s).
- > **ANTENNAS** 0 C_R solutions for all antennas.
- > **UVRANGE** 0 C_R no *uv* limits, or range if any, in kiloλ.
- > **INP VLACALIB** C_R to review inputs.
- > **VLACALIB** C_R to process the secondary calibrators.

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

4.5.3 Final (?) initial global calibration

At this point you should have gain and phase solutions for the times of all calibration scans, including the correct flux densities for the secondary calibrators. The next step is to interpolate the solutions derived from the calibrators into the CL table for all the sources. **CLCAL** may be run multiple times if subsets of the sources are to be calibrated by corresponding subsets of the calibrators, unless you limit it to one or more tables with **SNVER** and **INVERS**. **CLCAL** assumes that all SN tables contain only valid solutions and concatenates all of the SN tables with the highest numbered one. Therefore, any bad SN tables should be removed before using **CLCAL**. For polarization calibration, it is essential that you calibrate the primary flux calibrator (3C48 or 3C286) also so that you can solve for the left minus right phase offsets and apply **PCAL**.

CLCAL has caused considerable confusion and user error because it implements to somewhat contrary views of its process. The older view, represented by previous versions of this *CookBook*, had the user gradually

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

To use **CLCAL**:

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

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

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

Note that **CLCAL** uses both the **GAINUSE** and **GAINVER** adverbs. This is to specify the input and output CL table versions, which should be different. If you are building a single CL table piece by piece, then these must be set carefully and normally held fixed. In the more modern view, they are set to zero and the task takes the latest CL table as input and makes a new one. CL table version 1 is intended to be a "virgin" table, free of all injury from any calibration you do using the *AIPS* package. It may not always be devoid of information, as "on-line" corrections may be made and recorded here by some telescope systems, *e.g.*, the VLBA. The **VLA**, through tasks **FILLM** or **INDXR**, can now put opacity and antenna gain information in this file. **CLCAL** and most other *AIPS* tasks are forbidden to over-write version 1 of the CL table. This protects it from modification, and keeps it around so that you may *reset* your calibration to the raw state by using **EXTDEST** to destroy all CL table extensions with versions higher than 1. Be careful doing this, since you rarely want to delete CL version 1. Should you destroy CL table version 1 accidentally, you may generate a new CL table version 1 with the task **INDXR**. This new CL table may contain the calibration generated from

the weather and antenna gain files.

If you have any reason to suspect that the calibration has gone wrong — or if you are calibrating data for the first time — you should examine the contents of the output CL table. **LISTR** with **OPTYPE = 'GAIN'** will print out the amplitudes and phases in the specified CL or SN table. Note that these tables can be very large. Use the **SOURCES** and **TIMERANG** adverbs to limit the output, or look at it on your terminal (**DOCRT = 1**) so that you can stop the display whenever you have had enough. Task **SNPLT** will provide you with a graphical display which may be easier on the eye. **EVLA** users may wish to examine the SY table with **LISTR** or **SNPLT**; bad Psum and Pdif values may point to areas of bad data.

The task **EVASN** may help you determine the degree of phase and amplitude coherence in your calibration table. A lack of coherence suggests that the calibration is rather uncertain.

The most important step in the calibration is your verification that everything has gone according to plan. To check this, you should produce matrix listings for all your calibrator sources. For simplicity in interpretation, limit each listing to the **UVRANGE** to which you limited the calibrator during calibration. Thus:

> TASK 'LISTR' C_R	
> DOCRT -1 C_R	to direct output to the printer.
> SOURCES 'call', 'cal2', 'cal3', ... C_R	to list all selected calibrators by name.
> UVRANGE uvmin uvmax C_R	uv limits, if any, in kiloλ.
> OPTYP 'MATX' C_R	to get the matrix form of listing.
> DOCALIB TRUE C_R	to list with calibration applied.
> GAINUSE 0 C_R	TO point to the <i>latest</i> gain table.
> FREQID n C_R	list data for FQ <i>n</i> .
> DPARM = 5, 1, 0 C_R	to have amplitude and phase using scalar scan averaging.
> BIF 1; EIF 0 C_R	to select all IFs, LISTR will loop over IFs.
> INP C_R	to review the inputs.
> GO C_R	to run the program when inputs set correctly.

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

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

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

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

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

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

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

4.6 Polarization calibration

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

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

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

> TASK 'LISTR' C _R	
> SOURCES 'cal1', 'cal2', 'cal3', ... C _R	list all calibrators to be used.
> INEXT 'CL' C _R	to determine parallactic angle at times in CL table.
> INVER 1 C _R	CL version 1.
> FREQID n C _R	to use FQ number <i>n</i> .
> OPTYPE 'GAIN' C _R	to use gain table rather than visibility data.
> DPARM = 9, 0 C _R	to display parallactic angle.
> INP C _R	to review the inputs.
> GO C _R	to run the program when inputs set correctly.

Multiple calibrators may be used in determining the feed polarization, but the data from them must be accurately calibrated. In particular, the phase calibration of any calibrator used to determine antenna polarizations should be determined from that calibrator itself (*i.e.*, the source should be self-calibrated). Note that this will normally have occurred for all gain calibrators if the procedure described in the previous sections was followed.

The normal phase calibration technique treats parallel-hand visibilities in the two orthogonal polarizations independently. Thus, there will be a systematic phase difference between the two polarizations systems.

This difference may be due to differences in instrumental phase offset for the two systems or due to the propagation medium (*i.e.*, Faraday rotation) or both. Faraday rotation effects are particularly bothersome as they may be time variable and increase rapidly with wavelength. For data at L band or longer wavelengths, **AIPS** should be given an estimate of the ionospheric Faraday rotation measure using task **FARAD**. This task computes the ionospheric rotation measure using either total electron content from a nearby ionospheric monitoring station (Boulder Colorado for the **VLA**) or an empirical model that uses the monthly mean Zurich sunspot number (R1) as a measure of solar activity. If monitoring data are available, they should be used in preference to the model. **FARAD** enters the ionospheric Faraday rotation measure into the CL table. This is used by **PCAL** when determining antenna polarization parameters and is used by other calibration tasks to de-rotate the data when polarization corrections are applied. **FARAD** may be run any number of times with different parameters before **PCAL** is run; each run of **FARAD** over-writes the values written previously.

Data from Boulder for the total electron content for year *mm* is contained in the file TECB.*mm* in the directory with logical name AIPSIONS. Currently, data are available for 1980 onwards through part of 1992. Some gaps in the TEC data occur, particularly for years 1980 and 1988 and for times when solar activity has been high and no reliable estimate of the total electron content could be made. Unfortunately, we are no longer able to get the TEC data from Boulder. If your **VLA** data are more recent than early 1992, you can consider using the ionospheric model in **FARAD**, but you should be aware that the model is crude and should check that it improves matters before using it in your final calibration.

The phase offsets between the right-hand and left-hand polarizations at a given time may be determined from a source with a known angle of linear polarization USING data which have had the effects of imperfect feeds removed. The phase of the right-left correlations or the conjugate of the left-right correlations indicates the phase difference between the two polarizations.

The (initial) need for ionospheric corrections can be bypassed if either (1) you use unpolarized sources in **PCAL**, and/or (2) the ionosphere was well behaved during your observations. Typical rotation measures are only a few and therefore affect only L and longer-wavelength bands. The ionosphere is almost always well enough behaved to be ignored at shorter wavelengths and is usually able to be ignored even at L band. Changes of around 10° in the relative phases of R and L polarizations are not enough to disrupt a **PCAL** solution seriously. And, fortunately, calibrators at long wavelengths, such as P band, tend to be unpolarized. In general, if the ionosphere is well behaved, it can be ignored. If it is bad, no simple model is able to correct it and you may simply have to forget about polarization for that observing run. Note that an apparent position angle variation of 3C286 with time probably indicates that ionospheric rotation is significant. But, if 3C286 shows large rotations, it does not follow that its rotation can be applied to other directions in the sky. All it implies reliably is that a *model* is needed.

Polarization calibration of **EVLA** data is discussed extensively in Appendix E. These data are fundamentally multi-channel and a spectral-dependent polarization solution is required. **PCAL** and **RLDIF** now can determine and apply the calibration in a channel-dependent fashion. For data not requiring the spectral options, polarization calibration may be performed on amplitude- and phase-calibrated **VLA** data using the following five-step procedure:

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

- > **TASK** 'PCAL' C_R
- > **CALSOR** 'cal1', 'cal2', 'cal3', ... C_R list all calibrators to be used.
- > **TIMERANG** 0 C_R to use all times.
- > **ANTENNAS** 0 C_R to solve for all antennas.
- > **UVRANGE** *uvmin uvmax* C_R to set *uv* limits, if any, in kilo λ .
- > **BIF** 1 ; **EIF** 2 C_R to do both IFs.
- > **DOCALIB** 1 C_R to apply the calibration to the sources (very important!).
- > **GAINUSE** 0 C_R to use the latest CL table.

4.6. Polarization calibration

4. Calibrating Interferometer Data

> CLR2N C _R	to clear IN2NAME etc. since there is no Clean-image model.
> FREQID n C _R	to use FQ value n; only one polarization solution can be stored.
> DOMODEL 0 C _R	to solve for the polarization parameters of the calibration sources. PCAL can use a model only if that model has $Q = U = 0$ since it cannot solve for the right-left phase difference.
> SOLINT 2 C _R	to use a 2-minute solution interval; scan averages are usually sufficient.
> SOLTYPE 'APPR' C _R	to use linear approximation model.
> PRTLEV 1 C _R	to display the results and some diagnostic information.
> REFANT n C _R	only if REFANT reset since CALIB run.
> INP C _R	to review the inputs.
> GO C _R	to run the program when inputs set correctly.

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

The right minus left phase difference may not in fact be independent of time making the choice of reference antenna more important. In 31DEC12, the task SNREF examines the effect of the choice of reference antenna on the apparent stability of right minus left phases in SN or CL tables and can create tables with different choices.

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

> TASK 'RLDIF' C _R	to view only the polarization angle calibrator.
> SOURCE '3C286', '' C _R	to check all times.
> TIMERANG 0 C _R	antennas to use; the list used for CALIB.
> ANTENNAS list of antennas C _R	to limit uv, if appropriate.
> UVRANGE uvmin uvmax C _R	to view all IFs.
> BIF 1 ; EIF 0 C _R	to view the current FQ value (n).
> FREQID n C _R	to list with calibration applied.
> DOCALIB TRUE C _R	to correct for feed polarization and Faraday rotation.
> DOPOL TRUE C _R	to use the latest CL table.
> GAINUSE 0 C _R	to print the results on the line printer, DOPRINT > 0 prints on your terminal screen, and DOPRINT = 0 does no printing.
> DOPRINT -1 C _R	to do continuum polarization solutions.
> SPECTRAL -1 C _R	to examine the solutions without applying them to the tables.
> DOAPPLY -1 C _R	to review the inputs.
> INP C _R	to run the program when inputs set correctly.
> GO C _R	

The matrix of scan-averaged right minus left phase angles (actually RL and conjugate of LR polarizations) will be printed. Check that none of the phases differ from the mean by more than a few degrees. If any do, then use UVFLG to edit these data and go back to step 1. After the matrix of phases, the average over the matrix of the right minus left phases is displayed. This is the number to be used in step 4. RLDIF returns these, one for each IF, in the CLCORPRM adverb array. It even averages over multiple calibrator scans, getting a reliable estimate of the average by iteratively discarding outliers. To see the results, type

> **OUTPUTS** C_R to examine the output adverb values.

LISTR may also be used with OPTYPE 'MATX' ; STOKES 'POLC' to make the printer display, one IF at a

time. But you will have to do any averaging and placing of the results in `CLCORPRM` yourself.

This method will fail if the calibrator source (3C286 or 3C138, usually) is heavily resolved and the atmospheric phase stability is poor. (These two are frequently coupled!) Under these conditions, the self-calibration of the calibrator will have failed and will have to be done especially for the polarization calibration. In the steps below, you may safely relax the *uv* limits by about 20%, but should solve only for phases using `SOLMODE = 'P'`. The process consists of:

- 2.1 Apply `CALIB` to the inner (short-baseline) antennas on the calibrator source using the rules in the table found in § 4.3.3 but relaxed a bit. Set `DOCALIB = 1` ; `GAINUSE = 0` ; `SOLMODE = 'P'`.
- 2.2 Use `CLCAL` to apply these solutions to the calibrator source.
- 2.3 Run `LISTR` for cross-hand phases using *only* the antennas used with `CALIB`.
- 2.4 Use `EXTDEST` to delete CL table 3, a most important step.

After correcting the calibration, repeat steps 2.1 and 2.2 and the special calibration until satisfactory results are obtained.

Step 3: Use `TASAV` to copy all your table files to a dummy *uv* data set, saving in particular the CL table with the results of the amplitude and phase calibration. This step is not essential, but it reduces the magnitude of the disaster if the next step is done incorrectly. (Note - this may be a good idea at several stages of the calibration process!)

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

The task `TACOP` may be used to recover any tables that get trashed during later steps. `CLCOR` will make a new CL table now, so a `TACOP` step is not needed.

Step 4: The right minus left phase offset corrections should be made using task `RLDIF` although the old mechanism using `CLCOR` will also work. Use:

```
> TGET RLDIF                                to get the parameters used in the last and most successful run
                                                of RLDIF.
> DOPRINT 0 CR                            to turn off all printing.
> DOAPPLY 1 CR                            to apply the corrections to the SU, CL, and AN tables.
> INP CR                                to review the inputs.
> GO CR                                To change the tables applying the correction to the source
                                                polarizations, the antenna D terms, and the calibration
                                                phases.
```

The old method of correction is no longer recommended. It goes as follows. The phase offset correction is the expected value (*twice* the source polarization angle) minus the observed phases from step 2. The expected value is 66 degrees for 3C286, -18 for 3C138 (at L band, perhaps -24 at higher frequencies), and -140 for 3C48 (at 6-cm or shorter wavelengths). Thus, having used `RLDIF` and 3C286 in step 2 above

```
> FOR I = 1 : n ; CLCOR(I) = 66 - CLCOR(I) ; END CR      to convert the returned phases into
                                                               corrections for CLCOR, where n is the
                                                               number of IFs.
```

Then

```
> TASK 'CLCOR' CR
> SOURCE '' ; ANTENNAS 0 CR      to correct all sources and all antennas.
> TIMERANG 0 CR                to correct all times.
```

4.6. Polarization calibration

4. Calibrating Interferometer Data

> BIF 1 ; EIF 2 C_R	to correct both IFs.
> FREQID n C_R	to correct only the current FQ value.
> GAINVER 0 C_R	to modify the latest CL table produced by CLCAL .
> GAINUSE 0 C_R	to make a new CL table containing the phase corrections as well as all previous calibrations.
> OPCODE 'POLR' C_R	to do right minus left phase offset correction.
> STOKES 'L' C_R	correction applied to left circular polarization.
> INP C_R	to review the inputs.
> GO C_R	to run the program when inputs set correctly.

Task **RLCOR** may be used to apply this correction directly to a *uv* data set which is especially useful for single-source files to which **CLCOR** does not apply.

This will cause **CLCOR** to apply appropriate corrections to the CL, SU, and AN tables. If the CL table becomes hopelessly corrupted, delete it and return to Step 3. If the AN table is corrupted, then **PCAL** must be re-run. If more than one CL table needs to be corrected, use the **OPCODE='POLR'** option only once; other CL tables must be corrected using **OPCODE='PHAS'** and correcting 1 IF at a time. **CLCOR** (with **OPCODE = 'POLR'**) may be applied multiple times to the *same* CL table, in order to get the R-L phases "right." But you must *not* apply **CLCOR** in succession to different CL tables of the same database. If there is any doubt, rerun **PCAL**. The best way to judge if all is well in the final polarization solution is to look at the spread in the cross-hand phases for 3C286 or 3C138 (step 5 below). If the spread ("eyeball rms") is less than 3 degrees, then all is well. If more than ten, then there is definitely something wrong.

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

> TASK 'RLDIF' C_R	
> SOURCE 'cal1' , 'cal2' , ... C_R	to list the calibrators to be checked.
> TIMERANG 0 C_R	to display all times.
> ANTENNAS <i>list of antennas</i> C_R	to list the antennas to use.
> UVRANGE <i>uvmin uvmax</i> C_R	to set <i>uv</i> limits, if appropriate.
> BIF 1 ; EIF 0 C_R	to list all IFs.
> FREQID n C_R	to use the current FQ value (n).
> DOCALIB 1 C_R	to list with calibration applied.
> DOAPPLY -1 C_R	to leave the table values unchanged.
> DOPOL TRUE C_R	to correct for feed polarization and Faraday rotation.
> GAINUSE 0 C_R	to use CL table written by CLCOR .
> DOCRT 1	to display on your terminal.
> INP C_R	to review the inputs.
> GO C_R	to run the program when inputs set correctly.

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

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

An important consideration needs to be mentioned at this point. The polarization corrections applied to the RL and LR data multiply the RR and LL values. If those values are not well known, then the "calibrated" cross-hand data will be anything but calibrated. Normally, you should self-calibrate the parallel hand data of the target source and apply the phase and amplitude calibrations so derived to the cross-hand data while

applying the polarization D terms (with **DOPOL** 1). Note that, when polarization is measurable, there is almost always enough signal to self-calibrate the parallel-hand data.

4.7 Spectral-line calibration

The calibration of spectral-line data is very similar to that of continuum data with the exception that the antenna gains have to be determined and corrected as a function of frequency as well as time. The model used by *AIPS* is to determine the antenna gains as a function of time using a pseudo-continuum (“channel-0”) form of the data. Then the complex spectral response function (“bandpass”) is determined from observations of one or more strong continuum sources at or near the same frequency as the line observation. In general, the channel-0 data are calibrated using the recipes in the previous sections of this chapter. The sub-sections below are designed to bring out the few areas in which spectral-line calibration differs from continuum.

4.7.1 Reading the data

If your data are on a **VLA** archive tape then they should be read into *AIPS* using **FILLM**, as described in § 4.1.1. **FILLM** will fill a typical line observation into two files, a large one containing the line data only, and a smaller file containing the “channel-0” data. (Note that **FILLM** computes channel-0 from the line data rather than using the channel-0 provided by the on-line system.) The standard calibration and editing steps are performed on channel 0 and the results copied over to the line data set. You must be careful with the tolerance you allow **FILLM** to use in determining the FQ numbers. If you desire all of your data to have the same FQ number, so that you can calibrate it all in one pass, then set **CPARM(7)** in **FILLM** to an appropriately large value. If you wish to retain spectral-line autocorrelation data, you must set **DOACOR** to true.

By default for the **VLA**, the channel-0 data are generated by the vector average of the central 3/4 of the observing band. If this algorithm is not appropriate for your data, you may generate your own channel-0 data set by averaging only selected channels. You may now select different spectral channels in different IFs. To do this, use the task **AVSPC**:

> TASK 'AVSPC' C_R	
> INDI n ; GETN m C_R	to specify line data set.
> OUTDI i ; OUTCL 'CH 0' C_R	to specify output “channel-0” data set disk and class.
> ICHANSEL 10, 30, 1, 0, 31, 55, 2, 1 C_R	for example, to average every channel between 10 and 30 in all IFs and also every other channel between 31 and 55, but only in IF 1.
> GO C_R	to create a new channel-0 data set.

You might find this necessary when observing neutral hydrogen at galactic velocities. Most calibrator sources have some absorption features at these frequencies.

4.7.2 Spectral-line aspects of SETJY

The **LISTR** output with **OPTYPE = 'SCAN'** will show information from the source table including spectral-line parameters. **VLA** data from **FILLM** are normally supplied with adequate information regarding the source velocity, line rest frequency, and velocity reference (radio versus optical, LSR versus barycentric). However, data from the **EVLA** and other telescopes may be missing these parameters. **SETJY** must then be used to fill in the needed values. In 31DEC12, a new **OPTYPE='VCAL'** option computes the velocity of the

reference channel from first principles. It is recommended over inaccurate guesses of adverb values in other **SETJY OPTYPES**. It may be run over all sources after the rest frequencies and velocity reference information has been filled in.

4.7.3 Editing the data

You should follow the steps outlined in § 4.4 to edit the calibrator data using the channel-0 data set. Even though channel-0 data is continuum, be careful to have **TVFLG** and **UVFLG** generate the flagging commands for all channels, not just channel 1. Then, copy the resulting FG table to the line file. Use **TACOP**:

> TASK 'TACOP' C _R	
> INDI n ; GETN m C _R	to specify channel-0 data set.
> OUTDI i ; GETO j C _R	to specify the line data set.
> INEXT 'FG' C _R	to copy the FG table.
> INVER 1 C _R	to copy table 1.
> NCOUNT 1 C _R	to copy only one table.
> OUTVER 1 C _R	to copy it to output table 1
> INP C _R	to review the inputs.
> GO C _R	to run the program when inputs set correctly.

Specifying the “ALL-CH” setting in **TVFLG** and specifying **BCHAN** 1 ; **ECHAN** 0 C_R in **UVFLG** cause all channels to be flagged when the FG table is copied to the line data set.

Spectral-line observers should also use **SPFLG** (§ 10.2.2) to examine and, perhaps, to edit their data. This task is very similar to **TVFLG** described in § 4.4.3, but **SPFLG** displays spectral channels for all IFs on the horizontal axis, one baseline at a time. If you have a large number of baselines, as with the **VLA**, then you should examine a few of the baselines to check for interference, absorption (or emission) in your calibrator sources, and other frequency-dependent effects. Use the **ANTENNAS** and **BASELINE** adverbs to limit the displays to a few short spacings and one or two longer ones as well. If there are serious frequency-dependent effects in your calibrators, use **SPFLG** and **UVFLG** to delete them. (You might wish to delete the FG table with **EXTDEST** to begin all over again.) Then use **AVSPC** to build a new channel-0 data set and repeat the continuum editing. Note that you should not copy the FG table from the spectral-line data set to the new continuum one. The reason for this is the confusion over the term “channel.” If you have flagged channel 1, but not all channels, in the spectral-line data set — a very common occurrence — then a copied FG table would flag all of the continuum data since it has only one “channel.” When you have flagged the channel-0 data set, you can merge the new flags back into the spectral-line FG table with task **TABED**.

> TASK 'TABED' C _R	
> INDI n ; GETN m C _R	to specify channel-0 data set.
> OUTDI i ; GETO j C _R	to specify the line data set.
> INEXT 'FG' C _R	to copy the FG table.
> INVER 1 C _R	to copy table 1.
> OUTVER 1 C _R	to copy it to output table 1.
> BCOUNT 1 ; ECOUNT 0 C _R	to copy from the beginning to the end.
> OPTYPE 'COPY' C _R	to do a simple copy appending the input table to the output table.
> TIMER 0 C _R	to copy all times.
> INP C _R	to review the inputs.
> GO C _R	to run the program when inputs set correctly.

If the channel-0 data set is meaningful for your program sources, you might consider doing a first-pass editing of them along with your calibrators before copying the FG table back to the line data set. If your

program sources contain significant continuum emission, then this is a reasonable operation to perform. If they do not, then the standard channel-0 data set is not useful for editing program sources. You can use **SPFLG** to edit all channels, or if the signal is strong in a few channels, you could run **TVFLG** on those channels from the spectral-line data set or average those with the **BCHAN**, **ECHAN**, and **NCHAV** adverbs.

4.7.4 Bandpass calibration

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

> TASK 'BPASS' C_R	
> INDI <i>i</i> ; GETN <i>j</i> C_R	to specify the line data set.
> CALSOUR ' <i>cal1</i> ', ' <i>cal2</i> ', ... C_R	to specify bandpass calibrators.
> FREQID 1 C_R	to select which FQ value to use.
> ANTENNAS 0 C_R	to solve for all antennas.
> REFANT <i>n</i> C_R	to set the reference antenna number.
> DOCALIB FALSE C_R	to avoid applying calibration.
> BPASSPRM 0 C_R	to turn off all “parameters.”
> BPASSPRM(5) 0 C_R	to divide by channel 0 on a record-by-record basis before determining antenna-based bandpasses. Other normalization options are available and may be preferred.
> IN3DI <i>a</i> ; GET3N <i>b</i> C_R	to specify the channel 0 data file, or
> CLR3NAME C_R	to have channel 0 found from the input data themselves.
> ICHANSEL 20 50 1	to use the average, in each IF, of all channels from 20 through 50, for example, to determine channel 0, when the third input file name is empty.
> FLAGVER 1 C_R	to apply flag table 1.
> SOLINT 0 C_R	to use scan averages.
> BPVER 1 C_R	to select the output BP table number.
> INP C_R	to review the inputs.
> GO C_R	to run the program when inputs set correctly.

Be careful with the adverb **SMOOTH**. If you smooth, or do not smooth, the data while finding a bandpass solution, then you must apply the same **SMOOTH** adverb values whenever you apply that bandpass solution to the data. The only exception is that you may smooth the data after applying the bandpass solution with **SMOOTH(1)** values 5 through 8 when you did no smoothing in **BPASS**.

The stored bandpass correction table should be corrected for the spectral index of the calibration source. Use adverbs **SPECINDEX** and **SPECURVE** to describe the spectral index to **BPASS** and new task **SOUSP** to find the spectral index values from the fluxes in the source table. **BPASS** knows the flux coefficients for all standard calibration sources and, in 31DEC13, can fit the spectral index of other calibration sources from the SU table. It can handle spectral indices for multiple calibration sources at one time, although the adverbs apply of necessity only to the first such source.

The divide by channel 0 option is very convenient in that it allows one to ignore both source structure (when the bandwidth is narrow enough) and continuum calibration. However, the average of some channels on

a record-by-record basis can be rather noisy and the “division” operation is actually a subtraction of the average phase and a division by the average amplitude. The latter suffers from a “Ricean” bias — the average amplitude will always be larger than the correct amplitude, averaging one rms larger. Therefore, if the continuum calibration is stable (or already known and able to be applied) and the source structure is negligible, then it would be better to defer the normalization (on a baseline by baseline basis) until the data are averaged over **SOLINT** or, better still, to defer the normalization (on an antenna basis) until the unnormalized solutions are determined. **BPASSPRM(5)** and **BPASSPRM(10)** control the normalization options. Do note also that, with no normalization, **BPASS** is capable of replacing any use of **CALIB** including calibration of the data weights.

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

- | | |
|--|--|
| > MINAMPER <i>a</i> C _R | to count and, if BPASSPRM(2) > 1, to report amplitude closure failures > <i>a</i> per cent. Note that closure errors are accumulated as logarithms so that 0.5 and 2.0 are both errors of 100%. |
| > MINPHSER <i>p</i> C _R | to count and, if BPASSPRM(2) > 1, to report phase closure failures > <i>p</i> degrees. |
| > BPASSPRM(2) 1 C _R | to report statistics of amplitude and phase closure failures without reporting individual failures. |
| > BPASSPRM(6) <i>a</i> C _R | to report all channels in which the average amplitude closure error > <i>a</i> per cent. |
| > BPASSPRM(7) <i>p</i> C _R | to report all channels in which the average phase closure error > <i>p</i> degrees. |
| > SOLTYPE 'R' C _R | to select robust solutions which discard data with serious closure problems. Try other types if there are solution failures. |

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

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

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

- | | |
|---|--------------------------------------|
| > TASK 'POSSM' C _R | to specify the line data set. |
| > INDI <i>i</i> ; GETN <i>j</i> C _R | to specify the bandpass calibrators. |
| > SOURCES ' <i>cal1</i> ', ' <i>cal2</i> ', ... C _R | to include all antennas. |
| > TIMER 0 C _R | to average over all times. |
| > BCHAN 1; ECHAN 0 C _R | to display all channels. |
| > BPVER 1 C _R | to select the BP table. |
| > FREQID 1 C _R | to set the FQ value to use. |

> APARM = -1, 0 C_R	to do a scalar average and have the plot self-scaled and labeled in channels.
> APARM(8) 2 C_R	to plot BP table data.
> NPLOTS 0 C_R	to make one plot only, averaging all included data.
> INP C_R	to review the inputs — check closely.
> GO C_R	to run the program when inputs set correctly.
> GO LWPLA C_R	to send the plot to the (PostScript) printer/plotter.

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

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

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

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

In 31DEC16, task **BPEDT** offers a TV-inteactive graphical way to check your BP table. It displays the amplitude and phase of one antenna to be edited and up to 10 more antennas for comparison. The task is like **EDITA** except that the horizontal axis is spectral channel rather than time. You may walk through the entire bandpass table selecting the editable antenna and time in sequence. If some of the solutions show residual effects from **RFI**, then you may generate flags in a flag table to delete certain spectral channels for certain antennas and times. If you do this, of course, you should re-run **BPASS** applying the new flags.

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

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

4.7.5 Amplitude and phase calibration

The channel-0 data set should be calibrated as described above for continuum data (§ 4.4 and § 4.5). When you are satisfied with your results, you should copy the relevant CL table over to the line data set with **TACOP**:

> TASK 'TACOP' C _R	
> INDI <i>n</i> ; GETN <i>m</i> C _R	to specify the channel-0 data set.
> OUTDI <i>i</i> ; GETO <i>j</i> C _R	to specify the line data set.
> INEXT 'CL' C _R	to copy a CL table.
> INVER 0 C _R	to copy highest numbered table from CLCAL step.
> NCOUNT 1 C _R	to copy only one table.
> OUTVER 0 C _R	to create new output table.
> INP C _R	to review the inputs.
> GO C _R	to run the program when inputs set correctly.

If you copy SN, TY, or SY tables, you may apply a flagging table to the table values.

At this point it is often useful to examine your fully calibrated data using **POSSM**:

> TASK 'POSSM' C _R	
> INDI <i>i</i> ; GETN <i>j</i> C _R	specify line data.
> SOURCES 'source1', '' C _R	to specify the source of interest.
> ANTENNAS 0 C _R	to plot all antennas.
> BCHAN 10; ECHAN 55 C _R	to plot spectrum for this channel range only.
> DOCALIB 1 C _R	to apply the antenna gain to both visibilities and weights (if appropriate). calibration.
> GAINUSE 0 C _R	to use most recent CL table.
> DOBAND 3 C _R	to apply the bandpass calibration time smoothed.
> BPVER 1 C _R	to use BP table 1.
> FREQID 1 C _R	to use only one FQ value.
> APARM 0 C _R	to do vector averaging of amplitudes and self-scale the plots.
> SMOOTH 5,0 C _R	to apply Hanning smoothing in the spectral domain after bandpass calibration is applied. Use 13,0 to smooth after the data are averaged, which is faster and less prone to oddities due to channel-dependent flagging. Use 1,0 only if the data were Hanning smoothed when BPASS was run.
> INP C _R	to review the inputs.
> GO C _R	to run the program when inputs set correctly.
> GO LWPLA C _R	to send the plot to the (PostScript) printer/plotter.

If you have multiple FQ entries in your data set, you should repeat the calibration for each additional FQ entry. Bookkeeping is simplified if you eliminate all extant SN tables before calibrating the data associated with each frequency identifier. However, it is not essential to do this.

4.8 Solar data calibration

The calibration of solar *uv* data differs from normal continuum and spectral-line calibration in one critical respect: the system temperature correction to the visibility data is applied by the observer in *AIPS*. See Lecture 21 in *Synthesis Imaging in Radio Astronomy* for a discussion of the system temperature correction as it applies to *VLA* solar visibility data. The system temperature correction is embodied in a quantity referred to as the “nominal sensitivity,” an antenna-based numerical factor normally applied in real time to the scaled correlation coefficients before they are written to the *VLA* archive. With the exception of X and L band, only a handful of *VLA* antennas are equipped with so-called “solar CALs.” The nominal sensitivity is only computed for those antennas so-equipped, namely (for the old *VLA*) antennas 5, 11, 12, and 18 (at K, U, and C bands) and antennas 7, 12, 21, and 27 (at P band). The system-temperature correction for those antennas without solar CALs must, therefore, be bootstrapped from those antennas which do have solar CALS. This is accomplished through two tasks for the old *VLA* and two other tasks for the new vLA.. For the old *VLA*, *FILM* fills the uncalibrated visibility data to disk and places the nominal sensitivities in a TY extension table. Then, *SOLCL* applies the nominal sensitivities to calibration parameters in the CL table. For the new *VLA*, *BDF2AIPS* reads the data in, writing a SysPower (SY) and CalDevice (CD) table. Then *SYSOL* applies the gain and weight corrections. See § E.1 and § E.6.

4.8.1 Reading solar data from the VLA archive

New *VLA* data are loaded from the archive following the instructions in § E.1 with no special considerations for Solar observing except that the versions of *OBIT* and *AIPS* must be after mid-June 2017.

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

- > *VLAMODE* 'S' *C_R* to indicate solar mode observing.
- > *CPARM*(2) 16 *C_R* to indicate that moving sources are allowed without renaming.

If your experiment involved observing active solar phenomena, (e.g., flares), you may wish to update the system-temperature correction every integration time. For example, if you observed a flare with an integration time $\tau = 1.67$ seconds, choose

- > *CPARM*(8) 1.67 / 60 *C_R* for 1.67 sec CL and TY table intervals.

Loading an entire solar *uv*-data set to disk with the minimum integration time results in very large disk files which make all subsequent programs take a long time to run. A useful strategy is to load the data with relatively low time resolution (20–30 seconds for observations of active solar phenomena) and to proceed with the usual continuum data calibration, deferring the system temperature correction. When a satisfactory calibration is obtained, the relevant SN table may be saved using *TASAV*. (Note that you must save the SN table, before running *CLCAL* rather than the final CL table.) Then run *CLCAL* and inspect the data for interesting periods of activity — try *UVPLT* with *BPARM* = 11, 1 for plots of amplitude versus time or *TVFLG*, displaying amplitudes as a function of baseline length and time. Use *FILM* to load the relevant time ranges of solar *uv* data to disk with no averaging. The saved SN table is then copied to each high-time resolution data set. Assess, and possibly edit, the nominal sensitivities (§ 4.8.2) and then apply the system-temperature corrections (§ 4.8.3). Finally, apply the saved/copied SN table to the CL table of each using *CLCAL*.

4.8.2 Using SNPLT and LISTR to assess the nominal sensitivities

For new VLA data sets, use **SNPLT**, **LISTR**, and **EDITA** or **SNEDT** on the SY table (§ E.6). The following applies to old VLA data primarily.

When solar *uv* data are written to disk, **FILLM** writes the nominal sensitivities of those antennas equipped with solar CALs into the TY table. Before bootstrapping the system temperature correction for antennas without solar CALs from those which do, it is always wise to examine the nominal sensitivity for each of the solar CAL antennas for each of the IFs. The tools available for this purpose include: **SNPLT**, which plots the nominal sensitivities in graphical form, **LISTR** or **PRTAB**, which allow one to inspect the values directly, and **EDITA**, which provides an interactive display of the TY data and allows you to edit the data. To make plots:

> TASK 'SNPLT' ; INP C_R	to review the inputs needed.
> IND m ; GETN n C_R	to specify the input <i>uv</i> file.
> INEXT 'TY' C _R	to plot data from TY extension table.
> INVERS 0 C_R	to use the highest version number.
> SOURCES 'SUN', '' C_R	to plot solar source only.
> TIMERANG 0 C_R	to select all times.
> ANTENNAS 5 11 12 18 C_R	to select only CAL-equipped antennas; this sample list for K, U, or C band.
> PIXRANGE 0 C_R	to self-scale each plot.
> NPLOTS 4 C_R	to do 4 plots on a page.
> FACTOR 2; SYMBOL 5 C_R	to use triangles to mark the data and enlarge them by a factor of 2. The symbols may even be connected by lines.
> XINC 1 C_R	to plot every XINC th point.
> OPTYPE 'TSYS' C_R	to plot nominal sensitivities.
> INP C_R	to review the inputs.
> GO C_R	to run the program when you're satisfied with inputs.

SNPLT produces a PL extension file which may be plotted using **LWPLA**, **TKPL**, or **TVPL** — or you could set **DOTV TRUE** in **SNPLT** and get the display directly (and temporarily) on the TV. Then to inspect the values over some limited time range in detail, run **LISTR** (assuming the adverbs set above and):

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

Task **SNIFS** is similar to **SNPLT** except that it plots IF on the *x* axis to compare solutions across them. It has numerous binning options to control the otherwise excessive plotting.

The use of **EDITA** with TY tables is described extensively in § 4.4.2 and need not be described further here.

4.8.3 Using SOLCL or SYSOL to apply the system-temperature correction

For the new VLA, task **SYSOL** is provided with both non-Solar and Solar Tcals in the CD table and with measured switched power values in the SY table. The latter also tells *AIPS* tasks which Tcals apply to

which times. Therefore, the inputs to [SYSOL](#) are simpler than those for the old [VLA](#).

For the old [VLA](#), once you have identified the appropriate subset of reference solar CAL antennas for each source and IF, you are ready to bootstrap the system-temperature correction of the remaining antennas. It is recommended that you run [SOLCL](#) before applying any other calibration to the CL table. In this way, you can easily verify that the appropriate corrections have been made to each antenna. Then you apply the system-temperature correction to version 2 and correct mistakes by deleting and recreating version 2. To run [SOLCL](#):

> TASK 'SOLCL' ; INP C_R	to review the inputs needed.
> SOURCES ** C_R	to correct all sources.
> STOKES '' C_R	to correct both polarizations.
> TIMERANG 0 C_R	to correct all times.
> ANTENNAS 5 11 12 18 C_R	to use the listed antennas as references.
> SUBARRAY 1 C_R	to modify sub-array 1.
> GAINVER 2 C_R	to write corrected entries to CL table version 2.
> INP C_R	to review the inputs.
> GO C_R	to run the program when you're satisfied with inputs.

After applying the system temperature correction, you may proceed with the usual *AIPS* data calibration procedures outlined in previous sections, including the special solar tactics described in § 4.8.1.

4.9 Completing the initial calibration

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

4.9.1 Using FITTP and FITAB to write multi-source data to tape

The recommended way out of *AIPS* for multi-source *uv* data is to use [FITTP](#) to write a FITS-format tape. This will preserve the data and all associated calibration and editing tables in a machine-independent form. [FITAB](#) also writes a FITS-format tape or disk file using tables rather than random groups. This has the advantages of allowing a compressed format and of allowing *uv* files to be broken into “pieces” for increased reliability and control of space. [FITAB](#) output from earlier than 15-Oct-2007 can be read by versions of *AIPS* between 15APR99 and 15-Oct-2007, after 15-Oct-2007 it can only be read by versions of [UVLOD](#) or [FITLD](#) later than 15-Oct-2007. Also note that [FITAB](#) output cannot be read by other *uv*-data software packages, except *obit*. [FITTP](#) output can be read by some other packages. Consult § 3.9 about magnetic tapes in *AIPS*. That section tells you to mount your tape on the hardware device and then to do a software mount in *AIPS*. For example,

> INTAP n C_R	to specify which tape drive to use.
> DENSITY 6250 C_R	to set the density to 6250-bpi, if needed.
> MOUNT C_R	to mount the tape in software.

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

To write the data to tape:

> TASK 'FITTP' C_R	to specify the multi-source data set.
> IND m ; GETN n C_R	to write at the end of tape — if there are other data files on the tape you wish to preserve.
> DOEOT TRUE C_R	to write to tape just mounted.
> OUTTA INTAP C_R	to write associated tables.
> DOTABLE TRUE C_R	to use IEEE floating format for data.
> FORMAT 3 C_R	to use blocked FITS for tape efficiency.
> BLOCKING 10 C_R	to review the inputs.
> INP C_R	to run the program when inputs set correctly.
> GO C_R	

Most people use 8mm Exabyte or 4mm DAT tapes today. These have very large capacities. However, if you must still use half-inch reel tapes, you will find that many data sets (particularly spectral line) may be too large to fit on one 6250 bpi tape even with **BLOCKING** = 10. Since it is not possible to write multi-volume FITS tapes, it is recommended that you back up the single-source data sets formed after applying the calibration tables in **SPLIT** (see § 4.9.2). **FITAB** allows you to break up the data set into pieces which can fit on your tape. Multiple executions will be needed for multiple tapes. Alternatively, since all of the calibration information is contained in the extension tables, you may copy these to a dummy *uv* file with task **TASAV** and write this new file to tape with **FITTP**.

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

4.9.2 Creating single-source data files with SPLIT

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

> TASK 'SPLIT' C_R	to select sources, '' means all.
> SOURCE 'sou1', 'sou2', ... C_R	to keep all times.
> TIMERANG 0 C_R	to keep both IFs
> BIF 1 ; EIF 2 C_R	to set the one FQ value to use.
> FREQID 1 C_R	to apply calibration to the data and the weights.
> DOCALIB 1 C_R	to use the highest numbered CL table.
> GAINUSE 0 C_R	to correct for feed polarization.
> DOPOL TRUE C_R	to correct bandpass with time smoothing.
> DOBAND 3 C_R	to select BP table to apply.
> BPVER 1 C_R	to write the input Stokes type.
> STOKES '' C_R	to write visibilities in uncompressed format.
> DOUVCOMP FALSE C_R	to avoid channel averaging and autocorrelation data.
> APARM 0 C_R	to review the inputs.
> INP C_R	to run the program when inputs set correctly.
> GO C_R	

The files produced by this process should be completely calibrated and edited and ready to be imaged or further processed as described in later chapters. Note that one may wish to defer the **DOPOL** 1 part of this until after self-calibration of the parallel-hand visibilities.

It is not necessary to run **SPLIT** to make images with **IMAGR** and it is probably a good idea to make a couple of quick images to make sure that the calibration is okay. However, for serious imaging, it is probably best to run **SPLIT** and then use the single-source output files. See § 5.2 for details of the imaging process.

4.9.3 Making images from multi-source data with IMAGR

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

> TASK 'IMAGR' ; DEFAULT C _R	to select task and initialize all its parameters. This selects the usual convolution and weighting functions among other things.
> IND <i>m</i> ; GETN <i>n</i> C _R	to specify the multi-source data set.
> SOURCE 'sou1' , , C _R	to choose one source to image.
> STOKES 'I' ; TIMERANG 0 C _R	to image total intensity from all times.
> FREQID 1 C _R	to select FQ value to image.
> BIF 1 ; EIF 0 C _R	to image all IFs — multi-channel mode images only one IF.
> BCHAN <i>n</i> ; ECHAN <i>m</i> C _R	to combine a range of channels.
> NCHAV <i>N</i> C _R	to include <i>N</i> spectral channels in each image where $N \leq (m - n + 1)$; for each spectral channel, IFs <i>bif</i> through <i>eif</i> are also included. Note that each channel and IF included in the “average” image is handled individually at its correct frequency.
> DOCALIB 1 C _R	to apply calibration. Use DOCAL 100 C _R if the weights should <i>not</i> be calibrated.
> GAINUSE 0 C _R	to use highest numbered CL table.
> FLAGVER 1 C _R	to edit data.
> DOPOL TRUE C _R	to correct for feed polarization.
> DOBAND 3 C _R	to correct bandpass with time smoothing.
> BPVER 1 C _R	to select BP table to apply.
> OUTNAME 'sou1' C _R	to set the output file name to the source name.
> OUTDISK 0 C _R	to use any output disk with enough space.
> IMSIZE 512 512 C _R	to set the size in cells of image.
> CELLSIZE 0.25 , 0.25 C _R	to set the size of each image cell in arc-seconds.
> RASHIFT 0 ; DECSHIFT 0 C _R	to (not) shift image center.
> NFIELD 1 ; NGAUSS 0 C _R	to make only one image at high resolution.
> UVWTFN '' C _R	to use uniform weighting.
> ZEROSP 0 C _R	to introduce no zero-spacing flux.
> NITER 0 C _R	to do no Cleaning.
> INP C _R	to review the inputs.
> GO C _R	to run IMAGR when the inputs are set correctly.

5 Making Images from Interferometer Data

This chapter is devoted to the use of *AIPS* to make and improve images from interferometer visibility data. It begins with a brief description of the routes by which such data arrive in *AIPS*. The basics of weighting, gridding, and Fourier transforming the data to make the so-called “dirty” image are described, followed by a discussion of deconvolution, particularly Clean. The output of Clean is a model of the sky which, in cases of good signal-to-noise, can be fed back to improve the calibration of the interferometer data, a process called “self-calibration.” How this is done in *AIPS* is described. This entire process often isolates bad data samples, not previously removed from the data set. An interactive, baseline-based data editor called **EDITR** is described at the end of the chapter. You may find it more useful than **TVFLG** (§ 4.4.3) for removing data at this stage in the processing. Task **IMAGR** now does “3-dimensional” imaging, **SCMAP** contains an editing option at each self-calibration cycle, and **EDITR** has replaced **IBLED** as the baseline-based editor of choice. A *UV*-plane based interactive editor called **UFLAG** became available in 31DEC16.

Lists of *AIPS* software appropriate to this chapter can be obtained at your terminal by typing **ABOUT UV** C_R , **ABOUT CALIBRATION** C_R , **ABOUT EDITING** C_R , and **ABOUT IMAGING** C_R . Relatively recent versions of these lists are also given in Chapter 13 below. Basic data calibration is discussed in Chapter 4, editing is discussed in § 4.4 and § 8.1, and imaging and self-calibration are also discussed in § 8.4 for spectral-line data and in § 9.7 for VLBI data.

5.1 Preparing *uv* data for imaging

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

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

5.1.1 Indexing the data — PRTTP

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

- > **DATAIN = 'MYAREA:filename** C_R to point at the data file named *filename*.
- > **TPHEAD** C_R to see a summary of the file header to make sure it is the file you want.

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

5.1.2 Loading the data — FITLD and UVLOD

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

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

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

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

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

> **INDI OUTDISK ; UCAT C_R**

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

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

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

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

This sequence takes a little longer to execute, but the hard-copy list (sent to the appropriate printer) may be useful if your catalog is a long one. Note that the catalog has assigned an ordinal number to the data set in the first (CAT) column of the listing. This number and the disk number (3) should be noted for future reference as they are useful when selecting this data set for further processing. See § 3.3 and § 3.3.1.

5.1.3 Sorting the data — UVSRT

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

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

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

Once [UVSRT](#) has finished, check that a *uv* database with the “class” [.UVSRT](#) has appeared in your disk catalog by:

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

The catalog listing might now look like:

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

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

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

5.2 Basic image making — IMAGR

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

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

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

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

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

5.2.1 Making a simple image

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

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

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

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

- | | |
|---|-------------------------------------|
| > DOCRT = -1 ; EXPLAIN IMAGR C _R | to print the long explain file, and |
| > HELP <i>xxx</i> C _R | |

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

> **INP** C_R

then:

> **MAPP**R C_R to run **IMAGR**.

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

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

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

> **INDI 0 ; MCAT C_R**

This would produce a listing such as:

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

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

5.2.2 Imaging multiple fields and image coordinates

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

You specify the multiple-field information with:

- > **NFIELD n C_R** to make images of *n* fields.
- > **IMSIZE i, j C_R** to set the *minimum* image size in *x* and *y* to *i* and *j*, where *i* and *j* must be integer powers of two from 64 to 8192.
- > **FLDSIZ i₁, j₁, i₂, j₂, i₃, j₃, ... C_R** to set the area of interest in *x* and *y* for each field in turn. Each *i_n* and *j_n* is rounded up to the greater of the next power of 2 and the corresponding **IMSIZE**, **FLDSIZE** controls the actual size of each image and sets an initial guess for the area over which Clean searches for components. (That area is then modified by the various box options discussed later.)
- > **RASHIFT x₁, x₂, x₃, ... C_R** to specify the *x* shift of each field center from the tangent point; *x_n* > 0 shifts the field center to the East (left).
- > **DECSHIFT y₁, y₂, y₃, ... C_R** to specify the *y* shift of each field center from the tangent point; *y_n* > 0 shifts the field center to the North (up).
- > **DO3DIMAG TRUE C_R** to specify that the (*u*, *v*, *w*) coordinates are re-projected to the center of each field. **DO3DIMAG FALSE** also re-projects (*u*, *v*, *w*) to correct for the sky curvature while keeping all fields on the same tangent plane. The two choices now produce very similar results.

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

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

fields. When **ONEBEAM** is false, a beam of class **IBMnnn** is used for fields of class **IIMnnn** or **ICLnnn**, with similar substitutions for other Stokes parameters.

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

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

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

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

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

5.2.3 Data weighting

The minimum noise in an image is produced by weighting each sample by the inverse square of its uncertainty (thermal noise). *AIPS* assumes that the input weights are of this form, namely $W \propto 1/\sigma^2$. **FILLM** offers the option, for **VLA** data, of weighting data in this fashion using recorded system temperatures (actually “nominal sensitivities”). **EVLA** data may be weighted using SysPower data with **TYAPL** or simply with the variances found in the data with **REWAV**. In 31DEC14, channel-dependent weights may also be determined by **BWPWY** which finds a normalized weighting “bandpass.” The resulting weights are reasonably accurate for this use *after calibration*. In some cases, weights are simply based on integration times and the assumption that each antenna in the array had the same system temperature. If the weights are not of the correct form, run **REWAV** or, if necessary, **FIXWT** on the *uv* data set to calculate weights based on the variances in the data themselves. Then to get the minimum-noise image, specify

> **UVWTFN** 'NA' C_R to get “natural” weighting.

to have all samples simply weighted by their input weights. Unfortunately, most interferometers do not sample the *uv* plane at all uniformly. Typically, they produce large numbers of samples at short spacings with clumps of samples and of holes at longer spacings. Thus, the beam pattern produced by natural weighting tends to have a central beam resembling a core-halo source with the broad halo (or plateau)

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

produced by all the short spacing data and also to have rather large sidelobes due to the clumps and holes. In some VLBI arrays, data from some baselines have weights much greater than from other baselines due to differences in antenna size and receiver temperature. Only the high-weight baselines contribute significantly to a natural-weighted image in this case.

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

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

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

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

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

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

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

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

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

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

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

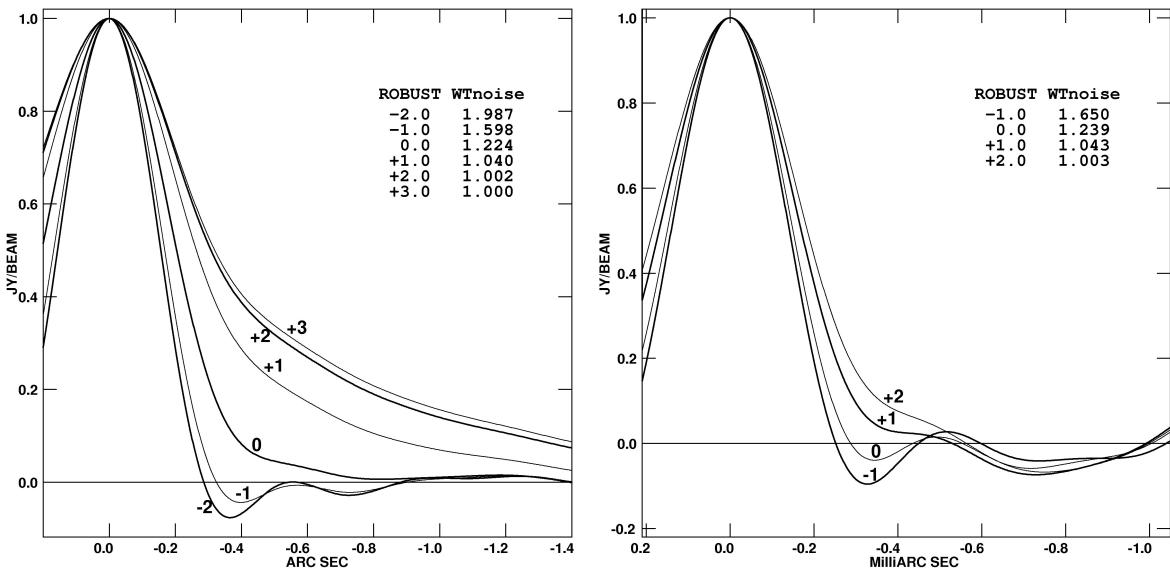


Figure 5.1: Slices taken through the centers of synthesized beams for various values of the **ROBUST** parameter. Plot at left for a **VLA** A- and B-array data set, while the plot at right is for a VLBA data set. Do not assume that these plots apply to your data sets, however. Tables give noise increase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

be improved with reduction of erroneous stripes, noise, and sidelobe levels. You should explore a range of **UVTAPER** and **ROBUST** (at least) in a systematic way in order to make an informed choice of parameters.

The option to average the input data in time over a baseline-length dependent interval adds further complications. It has the desirable effect of reducing the size of your work file, thereby speeding the imaging process. But it changes the distribution of data samples ahead of uniform weighting since it averages the short-spacing samples a great deal more than the long-spacing samples. This makes natural weighting less undesirable and causes uniform weighting to be less sensitive to extended source structures. Set **IM2PARM**(11) and (12) to request this option. In 31DEC12, **IM2PARM**(13) adds frequency averaging to the time averaging option. It may be desirable to keep narrow spectral channels in the data set for editing purposes, but the bandwidth synthesis may be speeded up significant while causing no degradation by a judicious choice of channel averaging.

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

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

5.2.4 Cell and image size, shifting

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

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

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

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

5.2.5 Zero-spacing issues

You help Clean to guess what may have happened in the unsampled “hole” at the center of the *uv* plane by including a zero-spacing (usually single-dish) flux density when you make the image. This gives Clean a datum to “aim at” in the center of the *uv* plane. Extended structure can often be reconstructed by deep Cleaning when the zero-spacing flux density is between 100% and \sim 125% of the average visibility amplitude at the shortest spacings (run UVPLT to estimate this average for your data set). If your data do not meet this criterion, there may be no reliable way for you to image the extended structure of your source without adding further information to your observations (e.g., by adding *uv* data from a more compact array, by Fourier transforming a suitably tapered and deconvolved single dish image of the VLA primary beam, or by using such an image as the default image for a maximum entropy deconvolution as in § 5.3.7). See § 10.5 for further discussion. IMAGR treats the zero spacing differently from previous tasks. The adverb ZEROSEP gives five values, the I, Q, U, V fluxes, and a weight. This weight should be in the same units as for

your other data, since the `ZEROSP` sample is simply appended to your data set and re-weighted and gridded just like any other data sample. To have the zero spacing be used, both `ZEROSP(1)` and `ZEROSP(5)` must be greater than zero, even when you are imaging some other polarization. Previous “wisdom” held that the weight should be “the number of cells that are empty in the center of the *uv* plane,” but this does not appear to be correct with `IMAGR`.

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

5.3 Deconvolving images

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

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

5.3.1 Basic Cleaning with `IMAGR`

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

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

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

The procedure **MAPP** may be used for single-field Cleaning.

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

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

1024x1024 images, smaller values are better for smaller images of compact objects, and rather larger values may be good for extended objects or large numbers of fields. If the first Clean component of a major cycle is significantly larger than the last component of the previous cycle (and the messages let you tell this), then too few cells are being used.

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

- > **BMAJ** *bmaj* C_R to set the FWHM of the major axis of the restoring beam to *bmaj* arc-sec. **BMAJ** = 0 specifies that the beam is to be fitted.
- > **BMIN** *bmin* C_R to set the FWHM of the minor-axis of the restoring beam to *bmin* arc-sec; used if **BMAJ** > 0.
- > **BPA** *bpa* C_R to set the position angle of beam axis to *bpa* degrees measured counter-clockwise from North (*i.e.*, East from North); used if **BMAJ** > 0.

Use **BMAJ** < 0 if you want the *residual* image, rather than the Clean one, to be stored in the output file.

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

IMAGR makes a *uv* “workfile” which is used in its Clean step to hold the residual fringe visibilities. Its name is controlled with the **IN2NAME**, **IN2CLASS**, **IN2SEQ**, and **IN2DISK** parameters. If the first three are left blank and 0, the workfile will be deleted when **IMAGR** terminates. Even if the workfile already exists, **IMAGR** assumes that its contents must be initialized from the main *uv* file unless the **ALLOKAY** adverb is set ≥ 2 . This file is useful if you suspect that there are bad samples in your data. Use **LISTR** (§ 4.4.1) **UVFND** (§ 6.2.1), **PRTUV** (§ 6.2.1), **UVPLT** (§ 6.3.1) or even **TVFLG** (§ 4.4.3) to examine the file. If you find data which you think are corrupt, remove them from the *input uv* data set with **UVFLG**. These workfiles may eventually use an annoying amount of disk if **IN2SEQ** is left 0. Be sure to delete old ones with **ZAP** in this case.

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

- > **OUTDISK** *d* ; **GETONAME** *ctn* C_R to set the output name parameters to the name parameters of catalog entry *ctn* on disk *d*.

The other parameters that must be set to restart a Clean are **OUTVER**, the output Clean Components version number, and **BCOMP**, the number of Clean components to take from the previous Cleans. A restart saves you much of the time it took **IMAGR** to do the previous Clean, although it will make new beam images and a new file of residual visibilities unless you specify that it should not use **ALLOKAY**. An image can be re-convolved by setting **NITER** = the sum of the **BCOMP**s and specifying the desired (new) Clean beam. Images can be switched between residual and Clean (restored) form in the same way, setting **BMAJ** = -1 to

get a residual image. (For single fields, tasks **RSTOR** and **CCRES** may be used for this purpose.) **IMAGR** writes over the Clean image file(s) as it proceeds to Clean deeper. You can preserve intermediate Clean images, however, either by copying them to another disk file with **SUBIM** or by writing them to tape with **FITAB** or **FITTP**.

5.3.2 Multiple fields in IMAGR

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

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

- > **NFIELD** n C_R to make images of n fields.
- > **IMSIZE** i, j C_R to set the minimum image size in x and y to i and j , where i and j must be integer powers of two up to 8192.
- > **RASSHIFT** x_1, x_2, x_3, \dots C_R to specify the x shift of each field center from the tangent point; $x_n > 0$ shifts the map center to the East (left).
- > **DECSHIFT** y_1, y_2, y_3, \dots C_R to specify the y shift of each field center from the tangent point; $y_n > 0$ shifts the map center to the North (up).
- > **FLDSIZ** $i_1, j_1, i_2, j_2, i_3, j_3, \dots$ C_R to set the area of interest in x and y for each field in turn. Each i_n and j_n is rounded up to the greater of the next power of 2 and the corresponding **IMSIZE**, **FLDSIZE** controls the actual size of each image and sets an initial guess for the area over which Clean searches for components. (That area is then modified by the various box options discussed in §5.3.3.)

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

The manner in which the multi-field Clean is conducted requires some discussion. When **ONEBEAM** is true, there is a single dirty beam for all fields. For either value of **DO3DIMAG**, the dirty beams for each facet are, at least subtly, different. Using a single dirty beam allows the task to run faster at some compromise to

accuracy. Since the Clean component models are subtracted from the *uv* data at each major cycle, Clean will correct much of this compromised accuracy. In `OVERLAP < 2` mode with one beam, all pixels within Clean windows above the current threshold from all fields are selected for the Clark Clean at the same time. The component flux at which the major cycle terminates is adjusted by the number of iterations before and during that major cycle. All components found from all fields in the major cycle are subtracted at once from the residual data and a new set of residual images is constructed. When `ONEBEAM` is false, there is a different dirty beam for each field. Thresholds are set by reviewing the data in all fields as above. However, a major cycle is then conducted for each field individually in order of decreasing peak residuals (within the Clean boxes). The first field alone determines the flux at which the major cycle terminates for all fields. Components are subtracted from the residual data one field at a time.

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

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

There are a number of aspects of multi-field Clean that can trip up the unwary. The first is that the sidelobes of an object found in one field are *not* subtracted from the other fields in the minor Clean cycle. In fact, they are not even subtracted from pixels more than the beam patch size away in the same field. This can cause sidelobes of the strongest sources to be taken to be real sources during the current major cycle. (The $\text{OVERLAP} \geq 2$ sequence reduces this effect significantly.) At the end of the major cycle, all components from all fields are subtracted from the *uv* data. At this point, all sidelobes of the components are gone from all fields, but the erroneously chosen “objects” with their sidelobes will appear (in negative usually). This is normally not a problem. During the next cycle, Clean will put components of the opposite sign on the erroneous spots and they will eventually be corrected. Nonetheless, it is a good idea to restrict the Cleaning to the obvious sources to begin with, saving Clean the trouble of having to correct itself, and to open up the search areas later in the Clean. The TV options make this easy to do in `IMAGR`; see §5.3.3.

The situation is more complex if the multiple fields overlap. If a sidelobe in the overlap area is taken as a source in one major cycle, it will appear as a negative source in both fields at the start of the next major cycle (*only when OVERLAP < 2*). Clean will then find negative components in both fields and correct its original error twice, producing a positive “source” at the next major cycle. Such errors never get fully corrected. A simple rule of thumb is never to allow the search areas of one field to overlap with the search areas of another field — or use $\text{OVERLAP} \geq 2$. Even then, there is one other “gotcha.” In the restore step, Clean only restores components to the fields in which they were found (again, unless $\text{OVERLAP} > 0$). Thus, a real source visible in two fields will be found in only one after Clean; your two images of the same celestial coordinate will be in substantial disagreement. Therefore, you must be careful about which parts of which images you believe to represent the sky. Instead, use $\text{OVERLAP} \geq 1$ to have Clean components from all fields restored to all fields as needed. If $\text{OVERLAP} \geq 2$, the Clean and imaging are done in a fashion which greatly reduces the instabilities arising from Cleaning the same source (or sidelobe) from more than one field.

5.3.3 Clean boxes and the TV in IMAGR

Clean works better if it is told which pixels in an image are allowed to have components. The initial information on this is provided by the **FLDSIZE** adverb which gives the pixel dimensions of a rectangular window centered in each field in which Clean looks for components. This window can be nearly the full size of the image because the components are subtracted from the ungridded *uv* data. Cleaning windows or “boxes” can be specified with the adverbs:

- > **NBOXES** *n* C_R to set the number of boxes in which to search for Clean components. Must be ≤ 50 ; if 0, one Clean box given via **FLDSIZE** is used and **CLBOX** is ignored.
- > **CLBOX** *lx1,by1,rx1,ty1,lx2,by2,rx2,ty2,...* C_R to specify the pixel coordinates of the Clean windows as leftmost *x*, bottommost *y*, rightmost *x*, topmost *y* for boxes 1 through **NBOXES**. Circular boxes may also be specified as -1 , radius, center *x*, center *y* interspersed in any order with the rectangular boxes. Default is given by **FLDSIZE**(1).
- > **BOXFILE** '*area : infilename*' C_R to specify the name of a text file listing the Cleaning windows. Blank means no file.
- > **OBOXFILE** '*area : outfilename*' C_R to specify the name of an output text file to list the Cleaning windows after any modifications made while running **IMAGR**. Blank means a temporary file in \$HOME. **OBOXFILE** can be the same as **BOXFILE**. Can also be simply an environment variable followed by a colon to be assigned a unique name in that area corresponding to this execution of **IMAGR**.

The **BOXFILE** text file is an optional means by which Clean windows may be entered at the start of a run of **IMAGR** for all fields, not just the first. It is also the only way to enter more than 50 boxes for the first field; the limit is min (2048, 131072/**NFIELD**) (!) boxes per field with this option. The format of the file is one box per line beginning with the field number followed by the four numbers describing the box as in **CLBOX** above. Any line in which the first non-blank character is not a number is taken as a comment, a field definition (see § 5.2.2), a **BCOMP** value, a channel weight, or an “UNClean” box definition. This last begins with a U or u, followed by a field number and a circular or rectangular box which defines a region in which Clean components are not allowed to be found. This is used to force time-dependent components into their own field in the **TDEPEND** set of procedures. There may well be other reasons, such as centering a point component on a pixel in its own field, for forbidding the Cleaning of a source in a particular field. **NBOXES** and **CLBOX** are overridden if any boxes for the first field are given in the file.

You can use the TV cursor in advance of running **IMAGR** to set the Cleaning boxes. First, load the TV display with either the dirty image or a previous version of the Clean image of the first field; see § 6.4.1. Then type:

- > **TVBOX** C_R to begin an interactive, graphical setting of up to 50 boxes, or
- > **REBOX** C_R to do a similar setting of the boxes, beginning with the **NBOXES** boxes already in **CLBOX**.

Position the TV cursor at the bottom left corner of the first Cleaning box and press a trackball or mouse button. Then position the cursor at the top right corner of the box and press Button B. Repeat for all desired boxes. This will fill the **CLBOX** array and set **NBOXES** for the first field. Note that the terminal will display some additional instructions. These will tell you how to switch to a circular box and how to reset any of the previously set corners or radii/centers should you need to do so. **HELP REBOX** C_R will provide rather more details. The verb **DELBOX** allows you to delete boxes from **CLBOX** interactively.

You can also use the TV cursor in a very similar way to build and modify the **BOXFILE** text file. (You can also use your favorite text editor of course; see § 3.10.1 for general information about specifying and using external text files.) The verb **FILEBOX** reads the text file (if any) given by **BOXFILE** selecting those boxes (if

any) already specified for the specified field number which fit fully on the current image on the TV. Which field number you want is given with the **NFIELD** adverb, or, if that is zero, deduced from the Class name of the image on the TV. (Be careful to load the TV with the desired image before running **FILEBOX!**) You then carry out a graphical setting or resetting of boxes in exactly the same manner as with **REBOX**. The new and changed boxes are then added to the end of the text file. Different portions of the current field and other fields may be done and redone as often as needed. Verb **DFILEBOX** may be used like **DELBOX** to delete boxes interactively from a **BOXFILE**. The **BOXFILE** may be edited by task **BOXES** to put Clean boxes around sources from a source list such as the NVSS or WENSS. The task **FIXBX** may be used to convert the Clean boxes from the fields and cell sizes of one box file to those of another.

Task **FILIT** does something like **FILEBOX** but it does the TV load for you, using a roam mode if the image is larger than the TV display area. From a TV menu, you may then add boxes, change boxes, and delete boxes and even use an auto-boxing feature to add boxes automatically based on the peak values and noise in the image. **FILIT** will work on a set of facet images and should become the task of choice for examining multi-facet images and their Clean boxes. The non-interactive task **SABOX** will find boxes to use for a set of facet images, prepared, for example, by a shallow un-boxed Clean.

IMAGR will also create Clean boxes for you automatically. The adverbs **IM2PARM(1)** through **IM2PARM(6)** control the process, selecting the maximum number of boxes to be found at any one time, the cutoff levels as factors times the residual image robust rms, and more. **IM2PARM(7)** controls the starting list of Clean boxes for the next spectral channel — do they start with those input to **IMAGR** or also include those found in the current channel. See **HELP IM2PARM C_R** for information and *AIPS Memo No. 115 “Auto-boxing for Clean in AIPS”* by Greisen for a detailed explanation of this new option.

The real power of **IMAGR** becomes apparent if you set **DOTV = n**, where **NFIELD ≥ n > 0** is the field number first displayed on the TV. Before each major cycle, the current residual image is displayed on the TV and a menu of options is offered to you. (Note that the residual image before the first major cycle is the un-Cleaned dirty image.) The image displayed is interpolated up or decremented down (by taking every n^{th} pixel in each direction) to make it fit on the display and the current Clean boxes are shown. If you do not select a menu option, **IMAGR** proceeds after 30 seconds.

The interactive options appear in two columns. To select an option, move the TV cursor to the option (remember the left mouse button — see § 2.3.2) and press buttons A, B, or C. Button D will get you some on-line help about the menu option. The basic options, in the order in which they are displayed, are:

ABORT TASK

to stop the task abruptly, destroying the output images and exiting as quickly as possible.

TURN OFF DOTV

to resume the Cleaning now and stop using the TV to display the residual images. To turn the TV display back on, if needed, use the **TELL IMAGR** verb with suitable adverbs, including **DOTV TRUE**.

STOP CLEANING

to stop the Clean at this point, restore the components to the residual images, write them on disk, and exit.

OFFZOOM

to turn off any zoom magnification

OFFTRANS

to turn off any black & white enhancement

OFFCOLOR

to turn off any pseudo-coloring

TVFIDDLE

to interactively zoom and enhance the display in black & and white or pseudo-color contours as in AIPS

TVTRAN

to enhance in black & white as in AIPS

TVPSEUDO

to select many pseudo-colorings as in AIPS

TVFLAME

to enhance with flame-like pseudo-colorings as in AIPS

TVZOOM

to set the zoom interactively as in AIPS

CURVALUE	to display the pixel value and x, y pixel coordinates at the TV cursor position as in AIPS
SET WINDOW	to select a sub-image of the whole to be reloaded with better resolution — all boxes must be included.
RESET WINDOW	to select the full image and reload the display
TVBOX	to set the Clean boxes for this field beginning at the beginning as in AIPS. If existing boxes will be deleted, you are asked if you want that. If no, REBOX is done instead.
REBOX	to reset the current Clean boxes and create more as in AIPS
DELBOX	to delete some of the current Clean boxes as in AIPS
UNBOX	to make or re-make “UNClean” boxes for this field
CONTINUE CLEAN	to resume Cleaning now rather than wait for the time out period.

A blank **OBOXFILE** adverb causes a temporary one to be created in \$HOME. A **TELL** operation can change this adverb to a more permanent file. If **TVBOX**, **REBOX**, or **DELBOX** used, the new Clean boxes will be written to the text file, replacing any previously in that file. (All non-box cards in that file are preserved unchanged; a new **OBOXFILE** will be filled with the non-box cards from **BOXFILE**.)

If **NFIELD** > 1, a sufficient number of additional options appear of the form

SELECT FIELD <i>n</i>	to display field <i>n</i> , allowing its Clean boxes to be altered or
SELECT NEW FIELD	to prompt on the terminal for a new field number to be displayed, allowing its Clean boxes to be altered (when > 64 fields).

Thus you can look interactively at the initial dirty images, place boxes around the brightest sources, and start the Clean. As it proceeds and weaker source become visible, you can expand the boxes and add more to include other sources of emission. Do be careful, however. Boxes that are too tight around a source can affect its apparent structure. The author once made Cas A into a square when stuck with a too-tight box. If **OVERLAP** = 2, the **SELECT FIELD** options are displayed when all residual images are current, *i.e.*, at the beginning, but are replaced by the options

REMAKE IMAGES	to re-compute all fields using the current residual <i>uv</i> data and then to display all fields on the TV.
THIS IS FLD <i>n</i>	to indicate that only field <i>n</i> is displayed and available to have its boxes altered.
FORCE A FIELD	to prompt on the terminal for a field number, exit TV, re-compute and display that field with current residual data (if needed) and then Clean that field (only available if NFIELD > 64 or DOWAIT true).
STOP FLD <i>n</i>	to mark field <i>n</i> as Cleaned sufficiently.
ALLOW FLD <i>n</i>	to allow further cleaning of previously stopped field <i>n</i> .
CHECK BOXES	to see if Clean boxes in one field overlap with Clean boxes in other fields and drop some to avoid this situation.

We encourage use of **DOTV TRUE** **QP** when you are Cleaning an image, especially for the first time or when using the options described in the next section. Watching the TV display as the Clean proceeds will help you to gauge how to set up control parameters for future Cleans and how long to iterate. It may also warn you about instabilities in the deconvolution if you compare the appearance of extended structures early and late in the Cleaning process. The instabilities referred to in § 5.2.4 were first seen while Cleaning with the TV option.

5.3.4 Experimental variations on Clean in IMAGR

Experimental variations of the familiar Cleaning methods have been introduced in **IMAGR**. One deals with the so-called “Clean bias” which causes the fluxes of the real sources to be underestimated. The other two deal with the inadequacies of Clean in modeling extended sources.

5.3.4.1 Clean-component filtering

It has been found that Clean will eventually assign some components to noise spikes in regions which do not have real sources, producing the “Clean bias.” Real source flux is underestimated, presumably because “sidelobes” of the noise “sources” get subtracted from areas of real sources. The magnitude of the effect is rather variable and is not understood. There are two older tasks discussed in § 5.3.6 which deal with the problem. However, it would be better to remove “weak, isolated” (presumably spurious) Clean components as Clean proceeds rather than only after the fact. One cannot do this at every Clean major cycle, since all components are likely to be weak and isolated initially. But it is a good idea to do it a few times while *uv*-plane based Cleans still have the ability to respond to the filtering. Note that this option may be used to remove negative “bowls” surrounding sources imaged with too little short-spacing data. However, it does not remove the bowl that sits on the source itself and so the final flux will be too low. You should consider the multi-scale Clean instead.

To have **IMAGR** filter Clean components set **IMAGRPRM(8)** $\neq 0$. Then **IMAGR** will select only those Clean components having $> \text{abs}(\text{IMAGRPRM}(8))$ Jy within a radius of $\text{abs}(\text{IMAGRPRM}(9))$ cells of the component. Note that this rejects all areas of negative flux, unless **IMAGRPRM(8)** < 0 , in which case the absolute value of the flux near the component is used. You can change these parameters with **TELL**, but only if **IMAGRPRM(8)** was non-zero to begin with. A copy of the input data has to be made for this option and it is only made if **IMAGRPRM(8)** is non-zero. If this option is selected, the output CC files will have been merged. Note that **IMAGRPRM(8)** should always be ≤ 0 for images of Q, U, and V Stokes parameters since negative brightnesses are valid. Filtering is done on restarts, when requested from the TV, on certain Cleaning failures, and near the exit. Clean is continued after this last filtering if **IMAGRPRM(9)** < 0 , but usually terminates fairly quickly. When this option is available, an addition TV option will appear:

FILTER COMPS

to exit TV, filter all Clean components, and then re-compute
all fields using new residual data.

This is the only way to filter before the end of the current Clean.

5.3.4.2 SDI modification of Clean in IMAGR

A modified Clean algorithm that attempts (often successfully) to suppress the striping and bumpiness in Cleans of extended sources has been developed by Steer, Dewdney and Ito (1984, *Astron. & Astrophys.* **137**, 159). In the *AIPS* modification of the algorithm, Clean proceeds normally until the residual image becomes rather smooth. It then takes many components at once from all high-residual cells rather than trying to decide exactly which *one* cell is the highest. This algorithm attempts to cut the top off the plateau of emission found in the residual image(s) in a relatively uniform way. Unfortunately, it is very expensive to determine the correct weights to use for each pixel in this algorithm. The technique used by **IMAGR** does apply larger weights to isolated pixels and pixels at the edges of the “plateau” but these weights are still not quite large enough to avoid a tendency to make a slight rim around the plateau. The next cycle of SDI Clean does trim this down and the method converges well for the extended sources for which the algorithm was designed. **IMAGR** allows you to switch back and forth automatically between BGC and SDI depending on the contrast between the brightest and median residual pixel in the Clean windows. (SDI is used when the contrast is low.)

To allow **IMAGR** to use the SDI algorithm, specify **IMAGRPRM**(4) > 0. SDI Clean will be used when the fraction of pixels in the Clean windows exceeding half of the peak residual exceeds **IMAGRPRM**(4). **IMAGRPRM**(19) is used to limit the depth of an SDI Clean cycle. When this option is used, additional TV options appear:

FORCE SDI CLEAN	to force the next Clean cycle to use SDI method.
FORCE BGC CLEAN	to force the next Clean cycle to use Clark method.

These allow you to force the choice of SDI or BGC methods for the next Clean major cycle. After that, it reverts to selecting the method based on **IMAGRPRM**(4) and the histogram of residual values.

5.3.4.3 Multi-scale modification of Clean in **IMAGR**

Clean has problems with extended sources because the point-component model is so far from the reality for them. Greisen experimented in the 1970s with modeling sources as Gaussians rather than points, but found that there are always point sources in the image which cannot be modeled sensibly with an extended component. (“Bull’s eyes” get painted around every point object.) Wakker and Schwarz (1998, *Astron. & Astrophys.* **200**, 312) proposed a scheme in which a smoothed image and a difference image were Cleaned. This still used point-source models although the Clean beam used for restoring the smoothed-image components was extended. Cornwell and Holdaway (July 1999, Socorro imaging conference) described a scheme in which an image is Cleaned simultaneously at several scales.

A *uv*-based variation of this last algorithm is available in **IMAGR**. The multi-field capability is used to image for each of **NFIELD** fields, images at **NGAUSS** scales specified in the array **WGAUSS** in arcsec. The full-resolution image is convolved with a Gaussian of width **WGAUSS**(*i*) while a dirty beam appropriate to a component of that width is constructed. One of the **WGAUSS** must be zero if a point-source model is desired; a warning is issued if none of the resolutions is zero. **OVERLAP** = 2 mode is used. See **EXPLAIN IMAGR** for details of this new option. Users have found that **WGAUSS** values increasing by factors of $\sqrt{10}$ are frequently optimal. The most important additional control parameter is **IMAGRPRM**(11) which down-weights higher scales to allow Clean to work on the higher-resolution images with roughly equal probability. **FGAUSS** is used to select the minimum brightness to be cleaned at each scale; higher values at higher scales are usually desired. Reasonable values of these three parameters will require running **IMAGR** to determine the point-source resolution and then to determine the peak brightnesses in each scale and the apparent noise levels after some Cleaning. The other available “knobs” for this algorithm may safely be left at zero.³

5.3.4.4 Spectral-index corrections

When using wide-band, multi-channel data to image continuum sources, serious effects are seen if the source intensities change as a function of frequency. If these changes can be modeled as spectral index images with or without “curvature,” then **IMAGR** will allow the spectral-index effects to be compensated during Cleaning. First image each spectral channel (or group of closely-spaced channels) separately. Combine them into a cube with **FQUBE**, transpose the cube with **TRANS**, and solve for spectral index images with **SPIXR**. To use these images, set **IMAGRPRM**(17) to a radius (> 0) in pixels of a smoothing area and put the image name parameters in the 3rd and 4th input image names. Note that this algorithm is expensive, but that it can be sped up with judicious use of the **FQTOL** parameter.

³See “Aperture Synthesis Observations of the Nearby Spiral NGC 6503: Modeling the Thin and Thick Disks” 2009, AJ, 137, 4718, by Greisen, E. W., Spekkens, K. and van Moorsel, G. A. for a detailed description of this algorithm and an example of its use.

5.3.5 Data correction options in IMAGR

There are a number of effects which degrade the usual image deconvolution, but which are, optionally, handled differently by **IMAGR**. These corrections are primarily for observations made with widely spaced frequencies over fields comparable to the single-dish field of view. If you have such data and hope to achieve high dynamic range images, then these corrections are for you. Otherwise skip to the next section.

5.3.5.1 Frequency-dependent primary-beam corrections

The primary-beam pattern of the individual telescopes in the interferometer scales with frequency. Therefore, each channel of multi-frequency observations of objects well away from the pointing center effectively observes a different sky. When a combined source model is produced, there will be residuals in the visibility data that cannot be Cleaned as the data does not correspond to a possible sky brightness distribution. If **IMAGRPRM(1)** is larger than 0, then a correction is made in the subtraction of Clean components from the *uv* data to remove the effects of the frequency dependence of the primary beam. The primary beam is assumed to be that of a uniformly illuminated disk of diameter **IMAGRPRM(1)** meters. This correction is made out to the 5% power point of the beam with a flat correction further out. Note: this correction is only for the relative primary beam to correct to a common frequency and *does not* correct for the primary beam pattern at this frequency. Note that this algorithm is expensive, but that it can be sped up with judicious use of the **FQTOL** parameter. It adds essentially no cost when doing the spectral-index corrections, however.

5.3.5.2 Frequency-dependent correction for average spectral index

If the sources observed do not have a flat spectrum, then the source spectrum will have channel-dependent effects on the Cleaning of a similar nature to the primary beam effects described above. This problem does not depend on position in the field except, of course, that the spectral index usually varies across the field. Normally, however, it varies around -0.8 rather than about 0. To the degree that the structure in the field can be characterized by a single spectral index, the amplitudes of the data can be scaled to the average frequency. This is done, before imaging, by scaling the amplitudes of the *uv* data to the average frequency using a spectral index of **IMAGRPRM(2)**. For optically thin synchrotron sources, this spectral index is typically between -0.6 and -1.0. This correction cannot remove the effects of variable spectral index but allows a single correction which should usually be better than no correction at all. Note that a much more expensive correction and more accurate may now be made (see above).

5.3.5.3 Error in the assumed central frequency

If the frequency used to compute the *u*, *v* and, *w* terms is in error, there will be a mis-scaling of the image by the ratio of the correct frequency to that used. Since central frequencies are frequently computed on the basis of unrealistic models of the bandpass shape, the “average” frequency given in data headers is frequently in error. If **IMAGRPRM(3)** is larger than 0, it is assumed to be a frequency scaling factor for the *u*, *v*, and *w* that is to be applied before imaging. Again, this can only correct for some average error. Since individual antennas will have different bandpass shapes, no single factor can correct all of the error.

5.3.5.4 Array mis-orientation effects

Images made with a coplanar array not oriented towards the instrumental zenith will have a distortion of the geometry which increases in severity away from the phase tracking center. For non-coplanar arrays, the image is distorted rather than just the geometry. [VLA](#) snapshots are misaligned coplanar arrays, whereas [VLA](#) synthesis images cannot be considered to have been made with a coplanar array. Images made with mis-aligned coplanar arrays can be corrected using task [OHGEO](#) to remove the effects of this misalignment. Since this correction requires the knowledge of the observing geometry, in particular, the average parallactic and zenith angles, [IMAGR](#) computes these values and leaves them as header keywords for [OHGEO](#) to use.

5.3.5.5 Non-coplanar effects

[IMAGR](#) has a [IMAGRPRM](#)(4) option to attempt to correct for non-coplanar effects in imaging. If this worked, it would be very very slow. At this writing, it is not believed to work at all and is disabled in the code. See the explain information for further details. The [DO3DIMAG](#) option removes a good part of the non-coplanar effects by rotating the projected baselines to make each field tangent at its center.

5.3.5.6 Units mismatch of residuals and Clean components

In principle, the units of the residuals are different from those of the restored components. Both are called Jy per beam area, but the beam areas differ; that of a dirty image is — in principle — zero. If the area of the central lobe of the dirty beam is similar to the restoring beam area, then this effect is negligible. Similarly, if the Clean has proceeded well into the noise then this difference is of little consequence. However, if there is significant flux left in the residual image, then this difference may be important. If [IMAGRPRM](#)(5) > 0, [IMAGR](#) will attempt to scale the residuals to the same units as the restored components. The principal difficulty is determining the effective area of the dirty beam. Operationally, this is done inside a box centered on the peak in the beam with half-width [IMAGRPRM](#)(6) in *x* and [IMAGRPRM](#)(7) in *y*. It may be better, in this regard, to use the default Clean beam and this option at this stage and change resolution and units later with [CCRES](#); see § 7.6.4.

5.3.6 Manipulating Clean components

The list of Clean components associated with a Clean image can be printed with:

- > **TASK** 'PRTCC' ; **INP** to select the task and review its inputs.
- > **INDI** *n* ; **GETN** *ctn* **C_R** to select the Clean image, where *n* and *ctn* select its disk and catalog numbers.
- > **BCOUNT** *n₁* ; **ECOUNT** *n₂* **C_R** to list Clean components from *n₁* to *n₂*.
- > **XINC** *n₃* **C_R** to list only every *n₃*th component.
- > **DOCRT** FALSE **C_R** to route the list to the line printer, or use [TRUE](#) to route the display to your workstation window.
- > **GO** **C_R** to execute the task.

Some users of the CC file for self-calibration suggest that only the components down to the first negative, or down to some factor times the flux at the first negative, should be used. The justification for this advice is the assumption that negative components occur near the noise level. This is not always the case. They also occur to correct for previous over-subtraction or for an object which does not lie on a cell. In any case, [PRTCC](#) will display the first negative component if it is found during the printing (*i.e.*, before or during the range

printed). The task **CCFND** is designed solely to find the component number of the first negative and the number of the component having **FACTOR** times the component flux of that first negative. The total fluxes at these two positions in the file are also displayed.

You can plot the list of Clean components associated with a Clean image in various ways with **TAPLT**. For example, to plot the sum of the components as a function of component number enter:

- > **APARM 0 ; BPARM 0 ; CPARM 0** C_R to clear input parameters.
- > **APARM(6) 1; APARM(10) 1** C_R to have the component flux summed and plotted on the y axis.
- > **GO TAPLT** C_R to create the plot file.
- > **GO LWPLA** C_R to display the plot file on the laser printer.

TAPLT offers many options for plotting functions of table columns against each other. Enter **EXPLAIN TAPLT** C_R for details.

You can compare the source model contained in the CC file with the visibility data in a variety of ways. **UVSUB** allows you to subtract the components of some or all fields from the data, producing a residual visibility data set. **OOSUB** is similar, but allows for the frequency-dependent corrections performed in **IMAGR** and for limiting the components subtracted to those inside or outside of the primary beam. Of course, **IMAGR**'s workfile already contains these residuals with the CC files of all fields subtracted. Various display options can be used on these *uv* files; see § 5.3.1. **VPLT**, described in § 9.4.3, will plot a CC model against visibility data, one baseline at a time, n baselines per page.

The algorithm used by all *AIPS* Cleans assigns to a component only a fraction (**GAIN**) of the current intensity at the location of that component. As a result, the list of components contains many which lie on the same pixels. **CCMRG** combines all components that lie on the same pixel. This can reduce the size of the list greatly and, hence, the time required for model computations in tasks such as **CALIB** (§ 5.4) and **UVSUB**. Do this with

- > **TASK 'CCMRG' ; INP** to select the task and review its inputs.
- > **INDI n ; GETN ctn C_R** to select the Clean image, where n and ctn select its disk and catalog numbers.
- > **INVERS m ; OUTVER m C_R** to select the input version of the Clean components and to replace it with the compressed version.
- > **GO C_R** to execute the task.

Under a variety of conditions, the Clean component files produced by **IMAGR** will already be merged.

There should seldom be a need to edit Clean component files in detail. However, task **TAFLG** allows editing based on comparison of a function of one or two table columns with another function of another one or two columns. One interesting use for **TAFLG** would be to delete all components below some cutoff before running **CCMRG**. Enter **EXPLAIN TAFLG** C_R for details.

It has been found that Clean will eventually assign some components to noise spikes in regions which do not have real sources and that this produces the so-called “Clean bias” which causes the fluxes of the real sources to be underestimated. This is presumably because “sidelobes” of the noise “sources” get subtracted from areas of real sources, but the magnitude of the effect is rather variable and is not understood. There are two tasks which can help. **CCEDT** copies a CC file keeping only those components which occur in specified windows. Then it merges the file (like **CCMRG**) and discards all merged components of flux below a specified cutoff. Under some circumstances, such filtering of Clean components before self-calibration can be a more effective way of obtaining convergence of hybrid mapping (mostly for VLBI) than restricting Clean windows in **IMAGR**.

The second task, **CCSEL**, explicitly addresses the Clean bias problem. It sums the flux of all components within a specified distance of each component and then discards those components for which this sum is less than a specified threshold. The idea is to eliminate “weak isolated” components which are likely

to be those on noise points. You should run [CCMRG](#) before using [CCSEL](#) since the compute time increases quadratically with the number of components. [IMAGR](#)'s internal algorithm for filtering is much more efficient.

5.3.7 Image-plane deconvolution methods

The previous sections have described the task [IMAGR](#) which implements Clean by subtracting model components in groups from the ungridded *uv* data and re-imaging. This can be rather expensive. If you have a significant number of visibilities contributing to a fairly small image, it may be faster to use an image-plane deconvolution method. The venerable [APCLN](#) implements the Clark Clean in the image plane. Clean components are found during “minor” iteration cycles by Cleaning the brightest parts of the residual image with a “beam patch” of limited size, just as in [IMAGR](#). More precise Cleaning is achieved at the ends of “major” iteration cycles when the Fourier transform of the Clean components is computed, multiplied by the transform of the beam, transformed back to the image plane, and then subtracted from the dirty image. This method does a good job Cleaning the inner quarter of the image area, but artifacts of the Cleaning and aliasing of sidelobes do interesting things to the remaining 75% of the image. Make the dirty image using [IMAGR](#) and be sure to make it large enough to include all of the source in the inner quarter of the area. [APCLN](#) uses many of the now-familiar adverbs of [IMAGR](#), including [GAIN](#), [FLUX](#), [NBOXES](#), [CLBOX](#), [FACTOR](#), [MINPATCH](#), [MAXPIXEL](#), [BMAJ](#), and more. [APCLN](#) recognizes only rectangular boxes and its [DOTV](#) option only displays the residual image with a pause for you to hit button D to end the Cleaning early.

The subject of image deconvolution has been widely studied and many methods have been proposed for tackling it. Clean is renowned for yielding images that contain many artificial beam-sized lumps or stripes in smooth low-brightness regions. Point sources are a poor model for such regions. You should compare heavily Cleaned images with dirty, or lightly Cleaned, images to test that any features you will interpret physically have not been introduced by these Clean “instabilities.” The *ATPS* Clean tasks have an optional parameter [PHAT](#) that will add a small-amplitude δ -function to the peak of the dirty beam in an attempt to suppress these instabilities as described by Cornwell (*Astron. & Astrophys.* **121**, 281 (1983).)

A modified Clean algorithm that attempts (often successfully) to suppress these instabilities has been developed by Steer, Dewdney and Ito (*Astron. & Astrophys.* **137**, 159 (1984)). In this algorithm, Clean proceeds normally until the residual image becomes rather smooth. It then takes many components at once from all high-residual cells rather than trying to decide exactly which *one* cell is the highest. The algorithm is embodied in the well-tested *ATPS* task [SDCLN](#), which is actually an enhanced version of [APCLN](#). The source must be contained in the inner quarter of the image area as in that task. Type [EXPLAIN SDCLN](#) C_R for information. [SDCLN](#) gives excellent results on extended sources, but is exceptionally CPU-intensive.

The most widely used, best understood, and probably most successful alternative to Clean is the Maximum Entropy Method (“MEM”). This is implemented in *ATPS* by the task [VTESS](#). This requires a dirty image and beam, such as those produced by [IMAGR](#) with [NITER](#) set to 0, each twice the (linear) size of the region of interest (as for [APCLN](#) and [SDCLN](#)). The deconvolution produces an all-positive image whose range of pixel values is as compressed as the data allow. The final [VTESS](#) image is therefore stabilized against Clean-like instabilities while providing some “super-scale” wherever the signal-to-noise ratio is high. [VTESS](#) can also deconvolve multiple images simultaneously; see below.

There are three main reasons to prefer MEM deconvolution over all of the Clean deconvolution methods:

1. MEM can be much faster for images which have strong signals in many pixels. “Many” seems to be $\geq 512^2$ or so.
2. MEM produces smoother reconstructions of extended emission than does Clean.
3. MEM allows introduction of *a priori* information about the source in the form of a “default” image.

Because [VTESS](#) can produce excellent deconvolutions of extended sources in much less computation time than Clean, but requires careful control, we recommend studying the output of [EXPLAIN VTESS](#) C_R before using the task. The [NOISE](#) parameter is particularly important; some have claimed that [VTESS](#) requires this to be within 5% of the correct value in order to deconvolve fully without biasing the total flux. (Use [IMEAN](#) (§ 7.3) to estimate the true rms.) Chapters 8 and 15 of the NRAO Summer School on *Synthesis Imaging in Radio Astronomy* also provide useful general background.

MEM can be used for quantitative work on regions of good (> 10) signal-to-noise ratio, if the dirty image is convolved with a Clean beam prior to deconvolution. Use the [AIPS](#) task [CONVL](#) for this purpose. The images may also be post-convolved, and added to the residuals, within [VTESS](#). In many cases, the images of extended sources produced by [SDCLN](#) and [VTESS](#) are functionally identical. [VTESS](#) usually converges in *much less* CPU time, however, at the expense of leaving significantly larger residual sidelobes close to bright compact (point-like) features. To get around this deficiency of [VTESS](#), first use Clean to remove the peaks of bright point-like features, then run [VTESS](#) on the residual image produced by this restricted Clean. (The [AIPS](#) Clean tasks will output a residual image if you set [BMAJ < 0](#).) The Clean components may be restored to the final image with tasks [CCRES](#) or [RSTOR](#).

[VTESS](#) can also combine information from different types of data. For example, single-dish data can be used to constrain the imaging of interferometer data, or many pointings covering one large object can be processed together. [VTESS](#) takes up to 4087 pairs of images and beams, together with some specification of the primary beam for each, either a circular Gaussian model or the [VLA](#) primary beam, and performs a joint maximum entropy deconvolution to get an image of one field. The images must all be in the same coordinate system, and a noise level must be known for each. The time taken is approximately the time [VTESS](#) would take for one input map and beam, multiplied by the number of map/beam pairs.

[VTESS](#) cannot be used on images which are not intrinsically positive, such as images of the Stokes Q, U, and V parameters. [UTESS](#) is a version of [VTESS](#) designed to deconvolve polarization images, for which a positivity constraint cannot be applied. For further information type [EXPLAIN UTESS](#) C_R

Two further alternatives to Clean have been implemented in [AIPS](#) as *experimental* tasks. These are algorithms due to Gerchberg and Saxton ([APGS](#)) and van Cittert ([APVC](#)). Type [EXPLAIN APGS](#) C_R , [EXPLAIN APVC](#) C_R for further information on these tasks.

5.4 Self-calibration

The task [CALIB](#) was described in some detail in Chapter 4 as the tool to determine the instrumental gains on calibrator sources which were then interpolated in time and applied to your program sources. If you have sufficient signal to noise in the latter, you may now use [CALIB](#) to improve the dynamic range of your images. The assumption is made that your images have been degraded by antenna-based (complex) gain errors which vary too rapidly with time or direction to have been fully calibrated with the calibrator sources. [CALIB](#) compares the input *uv* data set with the predictions of a source model — a point-source initial guess or your current best set of Clean components — in order to compute a set of antenna-based amplitude and phase corrections as a function of time which would bring the data into better agreement with your current model. For an n -element array, there are $(n - 1)/2$ times more observations than unknown antenna gains at any time, so the process is well-determined when n is reasonably large. Since this process uses the data to calibrate themselves, it is called self-calibration.

Do not use [CALIB](#) unless your data have enough signal-to-noise to warrant improvement. Ask yourself whether your externally-calibrated Clean images contain un-Cleanable artifacts well above the noise, and whether your source meets the criteria for self-calibration given by Tim Cornwell and Ed Fomalont in Lecture 9 of *Synthesis Imaging in Radio Astronomy*. Note that if your images are limited by receiver noise, self-calibration may produce erroneous results, including the fabrication of weak sources!

5.4.1 Self-calibration sequence and SCMAP or SCIMG

If you decide to use self-calibration, a good sequence of steps is:

1. Use **UVPLT** to make a plot file showing the shape of the visibility function as a function of baseline length in the externally-calibrated data set. (See § 6.3, especially § 6.3.1, for information about plotting in *AIPS*.) *N.B.*, for large data sets, use **XINC** to reduce the number of points plotted to no more than a few thousand; otherwise it will take too long to make and plot the plot file. Use **LWPLA** to get hard copy of the plot file.
2. If you can use a point-source model for the first iteration, *i.e.*, if a range of baselines sufficient to calibrate all antennas is dominated by a single component (flat visibility function well above the noise), go to step 6 directly. This is frequently done with VLBI data, but is less common with arrays for which the initial calibrations are better such as the **VLA**.
3. If you must use a more complicated model, obtain a Clean-component representation of it by making and Cleaning an image of the externally-calibrated data using **IMAGR**. Leave the *uv* data in “TB” sort order for **CALIB**; **IMAGR** will sort them if it has to. Note that you may want to use a somewhat higher loop **GAIN** in a Clean to be used as an input model for an early iteration of self-calibration than you would for final deconvolution of a very extended structure. Task **FACES** can prepare an initial model for wide-field observations particularly at long wavelength.
4. Consider running **CCMRG** to reduce the number of components in the model. This improves the speed of the calibration and makes the first negative component be a real negative rather than a minor correction to previous positive components. Remember that merging the components does alter the model which is used to compute the gains unless you were going to include all components anyway. Note that **IMAGR** often merges the components automatically.
5. Use **PRTCC** or **TAPLT** (as in the example in § 5.3.6) to help you decide how many components from this Clean to include in the **CALIB** model. **CCFND** is also helpful. When you have decided this, determine the appropriate *uv*-limits for the gain solution by referring to the hard copy of the visibility function you made at step 1.
6. Plan your **CALIB** inputs using the information given in the following two sections. The first few iterations are usually used to correct only phases; amplitude is normally corrected only in the last one or two iterations.
7. Use **CALIB** to calculate the gain corrections. If **DOAPPLY** ≥ 0 , it will apply them to produce a new, (hopefully) improved data set, and will also catalog the gain corrections as an SN extension to the *input uv* data file. You may set **DOFLAG** to produce and use new data flags based on closure failures.
8. Use **SNPLT** on the input data file with **DOTV = TRUE** to review the gain corrections before proceeding further. Use **DOBLANK = 1** to plot failed solutions as well as good ones. To take hard copy for future reference, run **SNPLT** with **DOTV = FALSE** and then run **LWPLA** on the plot files (usually more than one) produced. To plot the extrema of the gains use **OPTYPE = 'SUM'** in **SNPLT**.
9. Ask whether the gain corrections were believable — were they smaller than at the previous iteration of **CALIB**, if any? If not, is there a good reason why not? Did you change input parameters such as the model, the type of solution, or the solution interval, in a way that may have forced larger corrections than before? Proceed only if you are reasonably sure you understand what is happening at this point — otherwise consult a local expert at your site.
10. If the corrections were believable, run **IMAGR** to produce a new Clean image. Lower **GAINS** and higher **NITER** to produce deeper and more careful Cleans are appropriate as the self-calibration progresses.

11. Go back to step 4 and repeat the whole process if your new Clean image is a significant improvement over the previous one (with comparable Cleaning parameters on both occasions). You may want to go back to step 1 and repeat the process from there if you have been using amplitude self-calibration and wish to check that your amplitude calibration has not drifted significantly. If the new Clean image differs little from the previous one, do not continue on with further iterations of steps 4 through 10 unless you feel you can make an informed change to the [CALIB](#) input parameters at step 6. Task [UVDIF](#) (§ 6.2.1) may help you to decide whether there have been significant changes to your data due to the previous iteration of [CALIB](#).

If you have used the spectral-index (§ 5.3.4.4) and/or primary beam (§ 5.3.5.1) options in [IMAGR](#), you may continue to use those options in self-calibration. The procedure [OOCAL](#) will divide your multi-facet, channel-dependent model into the visibility data with task [OOSUB](#). It then runs [CALIB](#) on that output and copies the resulting SN table back to the original input file.

The tasks [SCMAP](#) and [SCIMG](#) attempt to implement this sequence inside a single task. [SCIMG](#) contains almost all of [IMAGR](#) and all of [CALIB](#). [SCMAP](#) is similar, but limited to a single field for simplicity. They attempt to make the decision about the number of merged components and the range of *uv* spacings to use in each self-calibration based on σ times the rms in the residual image of the current Clean, where you provide the σ . The process is somewhat less flexible, but also less painful, than running [CALIB](#) and [IMAGR](#) multiple times. They do not let you change imaging parameters while they are running, but they do provide automatic and interactive methods to change and save Clean boxes and to set a variety of Cleaning and self-calibration parameters including loop gain, solution interval, solution smoothing interval, and whether negative components are included or terminate the list. They let you switch from phase-only to amplitude and phase self-calibration or they will do it automatically when the phase only stops converging. Both tasks offer the full editing options of task [EDITR](#) (see § 5.5.2) displaying the input and current residual *uv* data with a wide variety of data selection and editing options. The [CALIB](#), [IMAGR](#), and [EDITR](#) process is similar, but conceptually simpler, so it is the one described here.

5.4.2 Self-calibration with CALIB

[CALIB](#) is the heart of the *AIPS* calibration package. The inputs to [CALIB](#) are extensive and spread over several screen pages. This is because the routines in [CALIB](#) are used in many situations — general calibration, real-time interferometry and VLBI. The task solves for antenna-based complex gains *i.e.*, “self-calibration,” whether the source being calibrated is a “calibrator” source (usually taken to be a point) or a “program” source (usually taken to be complex). The solutions that [CALIB](#) generates are stored in SN “solution” tables which are attached to the *input* data file. The SN tables can be plotted with [SNPLT](#) and listed with [LISTR](#). They can be edited themselves with [SNEDT](#) or be used to edit the *uv* data with [EDITA](#). If the corrections are usually small, the `OPTYPE='A&P'` option in [SNFLG](#) may perform the flagging you would do with [EDITA](#), but with very little effort on your part.

The following input parameters are used by [CALIB](#) for self-calibration of a single-source *uv* data set:

- > `TASK 'CALIB' ; INP CR` to specify the task and review the inputs.
- > `INDI n1 ; GETN ctn1 CR` to select the ‘TB’ sorted *uv* database.
- > `IN2D n2 ; GET2N ctn2 CR` to select the Clean model image(s) to use.
- > `NMAPS q CR` to specify the number of images with CC files to use for the model. If $q > 1$, the image class names are assumed to have the first three characters of [IN2CLASS](#) with the field number one given in the last three characters as is done by [IMAGR](#).
- > `NCOMP = n1, n2, ... CR` to cut off the model at the n_i^{th} Clean component in the i^{th} image.

- > **INVERS** m C_R to specify the CC file version number to use from *every* model image; 0 means the highest.
- > **SMODEL** S, x, y, m C_R to specify a point-source (or Gaussian or uniform spherical) model rather than a Clean component model. **CALIB** uses a source model (type m) of S Jy located at x, y arc-sec with respect to the pointing center. For a point model $m = 0$; see the help for details of the other types.
- > **SUBARRAY** s C_R to select the appropriate sub-array — **SUBARRAY** = 0 implies all sub-arrays.
- > **UVRANGE** $= x_1, x_2$ C_R to give full weight (in doing the gain solutions) only to data from projected baselines between x_1 and x_2 in kilo wavelengths.
- > **WTUV** w C_R to set the weight for projected baselines outside the range **UVRANGE**(1) → **UVRANGE**(2). **WTUV** = 0 is interpreted as zero weight and should *not* be used.
- > **REFANT** n_r C_R to select the reference antenna; for best results, choose one known to be good over most of the time range.
- > **SOLMODE** 'A&P' C_R to solve for amplitude and phase corrections simultaneously.
- > **SOLMODE** 'P' C_R to solve for phase weighted by amplitude, the default for single-source files.
- > **SOLMODE** 'PIA' C_R to solve for phase ignoring amplitude.
- > **SOLTYP** '' C_R to use a normal (non-linear) least squares solution.
- > **SOLTYP** 'L1' C_R to use an "L1" solution method in which a weighted sum of the moduli of the residuals is minimized. The computed gain solutions are less influenced by wild data points, but there is some loss of statistical efficiency and a modest increase in compute time. See F. R. Schwab, **VLA** Scientific Memo #136 for further details.
- > **SOLTYP** 'GCON'; **SOLMOD** 'GCON' C_R to solve for amplitude and phase using least squares with a gain constraint — this requires **GAINERR** and **SOLCON** as well; see the help file.
- > **ANTWT** w_1, w_2, w_3, \dots C_R to apply additional weights to each antenna (in order) in generating the solutions; 0 implies 1.
- > **APARM**(1) $= x_5$ C_R to reject solutions from fewer than x_5 antennas; default is 6.
- > **APARM**(2) $= x_6$ C_R to tell **CALIB** whether the data have already been divided by a model ($x_6 > 0$) or not.
- > **APARM**(3) $= x_7$ C_R to solve for RR and LL separately ($x_7 \leq 0$) or to average RR and LL correlators before solving ($x_7 > 0$).
- > **APARM**(5) $= x_8$ C_R to make separate solutions for each IF ($x_8 \leq 0$) or to average all IFs to make a single solution ($x_8 > 0$). It is better to do separate solutions unless you are desperate for signal to noise.
- > **APARM**(6) $= x_9$ C_R to set the level of diagnostic information as 0 (very little), 1 (some including time and closure error statistics), 2 (more including individual closure failures), 3 (even more including S/N ratio), or more (too much or much too much).
- > **APARM**(7) $= x_{10}$ C_R to discard solutions having S/N ratios $< x_{10}$; default is 5.
- > **SOLINT** $= x_{11}$ C_R to set the length of the solution interval (in minutes); default is 10 seconds for single-source files.

> NORMALIZE = 1 C _R	to scale the gain corrections by the mean modulus of all gains to keep the flux density scale from drifting; ≤ 0 lets the gains float free. NORMALIZE = 3 to normalize over antennas and polarizations, but not subarrays and IFs, is especially useful for wide bandwidth data.
> MINAMPER a_1 C _R	to set the level of amplitude closure error regarded as “excessive” to a_1 per cent. If APARM(6) ≥ 1 , summaries of the number of excessive errors by antenna are printed and, if APARM(6) > 1 , up to 1000 of the individual failures are printed. 0 means do not check or report amplitude closure errors of any sort. Note that amplitude closure errors are accumulated using logarithms so that gains of 0.5 and 2.0 are both errors of 100%. errors are reported only if they are “significant” following CPARM(7).
> MINPHSER p_1 C _R	to set the level of phase closure errors regarded as “excessive.” APARM(6) controls the display as for MINAMPER.
> CPARM(3) = a_2 C _R	to display a line when the average absolute value of amplitude closure errors is $> a_2$ % if $a_2 > 0$ and APARM(6) ≥ 1 .
> CPARM(4) = p_2 C _R	to display a line when the average absolute value of phase closure errors $> p_2$ degrees if $p_2 > 0$ and APARM(6) ≥ 1 .
> CPARM(5) = 1 C _R	to form scalar averages of amplitudes before doing solutions. This is useful only if the phases are bad, but the amplitudes have high signal to noise.
> CPARM(7) = N_σ C _R	to display excessive amplitude and phase errors individually only if they exceed MINAMPER or MINPHSER) and if their significance exceeds N_σ times the uncertainty implied by the data weights.
> DOAPPLY -1 C _R	to solve for a new SN table without writing a new corrected <i>uv</i> data set.

Other parameters are defaulted sensibly — type EXPLAIN CALIB C_R for further information. In general, the AIPS philosophy is such that if you don’t know what value to set for an adverb, leave it at the default — this will usually give you what you want, or at least something reasonable! There are two approaches to the self-cal loop. In one, a new *uv* file is written at each iteration and the next SN table should then be an incremental improvement over the previous. In the other, the initial single-source file is the input file in each iteration and the new SN tabole contains all of the corrections so far found. Both methods have things to recommend them; DOAPPLY controls the choice.

5.4.3 Considerations in setting CALIB inputs

In many cases, only a few input parameters to CALIB need be set, other than those selecting the *uv* data and the input model. The key parameters are NCOMP, UVRANGE, SOLINT and, if you are interested in polarization, REFANT.

It pays to be conservative when using NCOMP to select the number of Clean components which will comprise the input source model. Setting NCOMP too high will fossilize errors from the earlier calibrations in the model for the next one; after this, you are stuck with them as long as you continue feeding CALIB a model with as much Cleaned flux density. When calibrating Stokes I images, consider setting NCOMP in CALIB so that few negative Clean components are included. The first few iterations of CALIB should be phase-only calibration, since the tropospheric and ionospheric phase errors will almost always dominate amplitude

errors due to the atmosphere or to system drifts. In these first iterations, it is prudent to be even more conservative, setting **NCOMP** so that the total Cleaned flux included in the model is between 50% and 80% of that at which the first negative Clean component appeared. **CCFND** will help you with this (§ 5.3.6). If your field is dominated by a few very strong, small-diameter regions, it is a good idea to make the first iterations of **CALIB** work on Clean components from these regions alone, restricting the range of baselines suitably by setting **UVRANGE**(1). Setting Clean windows in **IMAGR** or using **CCEDT** (§ 5.3.6) suitably will help you do this. Even later in the self-calibration cycle, it is probably still a good idea to eliminate weak, isolated Clean components. Try **CCSEL** for this.

It is always important to restrict the high-weight domain of the **CALIB** solution to the part of the *uv* plane that is described well by the model. In the early stages of self-calibration, the trustworthy part of your Clean model will almost always contain less flux density than was measured in the visibility function at the shorter baselines. Another way of putting this is that the large-scale structure of the source will be poorly represented by the model. You should therefore set **UVRANGE**(1) so that the total flux density in the input model (the sum of the Clean components up to the Clean iteration selected by **NCOMP**) exceeds the peak visibility amplitude in your data at a baseline of **UVRANGE**(1) kilo wavelengths (read this off a plot file output from **UVPLT**). It is also important to give some slight weight to the rest of the *uv* plane so that some solution may be found for most all antennas including those having no baselines in the high-weight region.

SOLINT sets the length of the time interval, in minutes, over which the model and the data are averaged when computing the gain corrections. This must be *short* enough that the gain corrections can track the fluctuations produced by the atmosphere over the longer baselines with sufficient accuracy. It must be *long* enough that the variances of the computed gain corrections (which depend on the signal-to-noise ratios in the data over the *uv* range in which the model is being compared with the data) are acceptably small. These constraints vary from source to source, frequency to frequency, and (because of the “weather”) from day to day. They may not in fact be reconcilable for weak sources, especially in the wider **VLA** configurations and/or at the higher frequencies. In many combinations of these circumstances, you may not be able to self-calibrate your data. See Lecture 9 in *Synthesis Imaging in Radio Astronomy* for details of how to make this assessment. In VLBI imaging, it may be helpful to use a point-source model and quite small **SOLINT** for the first iteration of self-calibration to remove the gross and rapid changes due to atmospheric fluctuations. With that problem removed, it may then be possible to use longer **SOLINT**s and more complicated models.

REFANT selects the number of the reference antenna for the gain solutions. For total intensity continuum calibration, the choice of this **CALIB** input is unimportant. It is always best, however, to choose a reference antenna that was stable and present in all data throughout the run, if only because this prevents propagation of noise or glitches in the reference antenna through the gain solutions (and plots of them) for the other antennas. For polarization work, it is important to select an antenna for which both polarizations were always present; otherwise any polarization calibration which preceded **CALIB** may be seriously compromised.

Note that **CALIB** should almost always be run with **SOLMODE** set to phase-only calibration for the first iteration or two. Consider turning on amplitude calibration by setting **SOLMODE** ‘A&P’ only when either the phase adjustments being made are generally small (*i.e.*, the worst cases being a few tens of degrees) or the new re-Cleaned image is clearly dominated by amplitude errors — which will give symmetric Y-shaped patterns around strong point sources for **VLA** observations. In general, you will want to set **CPARM**(2) = 1 when using **SOLMODE** ‘A&P’, to prevent drifting of the flux-density scale during amplitude self-calibration.

CALIB has a number of new options to deal with difficult data. The adverb **WEIGHTIT** controls how the data are weighted when being processed by the gain-fitting routines. The default is $w = 1/\sigma^2$ which may cause too much contrast between the highest weighted points and the lowest. This problem is much worse when self-calibrating extended sources than when doing the primary calibration on point sources. If you encounter many failed solutions, try **WEIGHTIT** = 1 which uses $w = 1/\sigma$. If you have trouble with bad data and failed solutions, consider trying the “robust” forms of **SOLTYP** selected with ‘R’, ‘L1R’, and ‘GCOR’. A robust solution is one in which a solution is found, outlier data are temporarily flagged, a new solution

found, and the process repeated while gradually tightening the omission criteria. This should make for more reliable solutions when there are bad correlators or antennas and, as a side benefit, allows more permanent flagging of the data under control of adverb **DOFLAG**.

CALIB will also edit out bad data according to the following criteria:

1. there are too few antennas (**APARM(1)**) to form a solution,
2. the solution does not converge, or
3. the signal-to-noise ratio for a given antenna (**APARM(7)**) is too low.

The signal-to-noise ratio is calculated from the post-fit scatter of the residuals from the gain model. Note that the scatter will contain contributions from thermal noise *and* unmodeled source structure. This is a good reason to restrict the *uv* range of the data. For further guidance and information on other **CALIB** inputs, type **EXPLAIN CALIB C_R** and/or read Lectures 9 and 16 in *Synthesis Imaging in Radio Astronomy*.

We note again the recommendation from Chapter 4. It is best to self-calibrate the parallel-hand data of the target source before applying the polarization corrections (D terms) to the cross-hand data. The correction to the cross-hand data is the product of the D terms found by **PCAL** and the parallel-hand data. The latter are not fully correct until after the self-calibration step. There is almost always adequate signal for self-calibration when there is measurable polarization.

5.4.4 Experimental extension of multi-field self-calibration

High-quality, multi-field images often suffer from position-dependent calibration effects. At lower frequencies, the strongest sources may well lie outside the main portion of the primary beam and so be effected by different instrumental gains including pointing errors, differences in atmosphere or ionosphere, etc. An experimental **RUN** file **PEELR** compiles a procedure by that name. One “interfering” field at a time, it performs a self-cal to improve that facet. After a list of fields is processed, it restores the original multi-field model to the corrected residual *uv* data. See **HELP PEELR** for details. It seems almost magical, but it really does improve the final images. A slight increase in the image rms is the price one pays for removing larger, more systematic problems.

5.5 More editing of *uv* data

5.5.1 General remarks on, and tools for, editing

There are many programs which aid in the processing, display, and editing of *uv* data. Summaries of this software may be listed on your terminal with:

- | | |
|--------------------------------------|--|
| > ABOUT UV C_R | to list all <i>uv</i> -related software. |
| > ABOUT EDITING C_R | to list all editing software. |
| > ABOUT PLOT C_R | to list all plotting software. |

and are also in Chapter 13 of this *CookBook*. Type

> **DOCRT -1 ; EXPLAIN taskname C_R** to print information about task *taskname*.

to get more information about any of the tasks mentioned below. The discussion below assumes that you have deduced that there are suspect samples in your data set and that you want to remove them. Read § 4.4 before investing large amounts of time in editing even at this stage.

There are facilities in **CALIB**, **RFLAG**, **FLAGR**, **CLIP**, **CORER**, **UVMLN**, **FLGIT**, **DEFLG**, and **SNFLG** to flag *uv* data in *AIPS* based on deviations from specified norms. There is also the task **UVFLG** to flag and unflag by antenna-IF or by correlator. The task **UVPLT** plots various combinations of *uv* data; see § 6.3.1. The task **WIPER** makes a similar plot on the TV and allows you to wipe away offending data. The task **UVFND** is also recommended for printing out suspicious portions of the database; see § 6.2.1. Note that **CLIP** examines the data correlator by correlator, but **UVFND** normally converts the data to Stokes components (using the same criteria as **UVMAP**) before checking that the amplitudes and/or phases are in range. To examine the correlators individually, use **STOKES** 'CORR' in **UVFND**, or to flag the data based on their values after conversion to true Stokes use **STOKES** = 'IQUV' in **CLIP**. Task **FINDR** is a companion to **FLAGR** intended to assist you in determining what is normal within your data.

CLIP is also useful for flagging discrepant data (e.g., due to interference or malfunctions) on the basis of their deviations from the visibility predicted by a set of Clean components. Tasks **OOSUB** or **UVSUB** will subtract the Fourier transform of a set of Clean components from visibility data. You may then use **UVPLT** to display the residual *uv* data set and **CLIP** to flag abnormally high points. You may wish to be cautious, and run **UVFND** to display such points before running an automatic **CLIP** task — be especially careful not to **CLIP** away evidence for real extended structure near the center of your *uv* plane! Before re-imaging, you must copy, with **TACOP**, the flag table produced by **CLIP** to the data set used for imaging. Note that **IMAGR**'s workfile is also a *uv* data set from which the current Clean component model has been subtracted. It may also be used with **UVPLT** to help you to diagnose problems. If you run **CLIP** on it, you will need to append the resulting FG table to one on the imaging data set; use **TAPPE**.

FFT is another useful tool for finding suspicious data. Transform your image back into the (*u,v*) plane by running **FFT** and then display the results on the TV. Use image read-back verbs like **CURVALUE** and **IMPOS** (§ 6.4.5) to find the *u* and *v* values for abnormally high cells. Then use **UVFND** with **OPCODE 'UVBX'** to print the data surrounding these cells and **UVFLG** to delete any bad data. This method is particularly effective when applied to residual images from Clean. (You can instruct **IMAGR** to put out a residual image by setting **BMAJ <0**. **CCRES** can also make a residual image from a single-field normal image.)

In 31DEC16, the process described above has been improved upon considerably. Task **UFLAG** makes a *uv*-plane grid (with pill-box convolution) of your visibility data selected by adverbs much like those used in **IMAGR**. It shows you the vector and scalar averaged grids in amplitude and the vector averaged grid in phase and offers a variety of editing options. You may choose to mark an entire cell as "bad" or look at the visibilities contributing to cells you select and choose to mark some of them (rather than all) as bad. There are even automatic routines to do this last operation over cells meeting criteria that you set. The entire process is described in an *AIPS* Memo.⁴

There is a task called **FLAGR** which goes through a data set determining what are normal rmses and weights and then flagging those that deviate excessively including clipping all those that have amplitudes or weights outside specified normal ranges. **FLAGR** is intended for use eventually in pipeline data-reduction procedures, but at present should be considered experimental, but potentially very valuable. Before calibration, try it on your calibration sources using default values for most adverbs plus **SOLINT** set to 2.5 times the basic integration time and **VECTOR** set to 0. Task **FINDR** is a companion intended to determine what is normal in the data and then to print those values and return selected adverb values to AIPS for use by procedures.

There is a task for automatic flagging called **RFLAG**. Since **RFI** tends to be highly variable over time and/or frequency, **RFLAG** accumulates statistics about this variation which it plots along with recommended clip levels. Running it a second time, with **DOPLOT** ≤ 0 , causes the clip levels to be applied to make a large flag table. This appears to be a very successful program with **EVLA** wide-band data (§ E.5). Task **REFLG** may then be used to compress and extend the flag table.

TVFLG, **SPFLG**, **FTFLG**, **WIPER**, **IBLED** and **EDITR** are TV-based, interactive editors. **TVFLG** is most suitable

⁴Greisen, E. W. 2016, "Editing on a *uv* grid in *AIPS*" AIPS Memo 121, <http://www.aips.nrao.edu/aipsdoc.html>

for data sets with large numbers of baselines, *e.g.*, the [VLA](#), but it can be used usefully for VLBI data experiments with 10 or more antennas. [TVFLG](#) allows you a global overview of your data and can display the data for all baselines simultaneously as a function of time. This task is documented extensively in § 4.4.3 of the *CookBook*. [SPFLG](#) is a very useful task for data with a significant number of spectral channels. It is effective in examining data for frequency-dependent errors and interference and can be an effective data editor for interferometers with a small number of baselines; see § 10.2.2 and § 8.1. [FTFLG](#) is like [SPFLG](#), but combines all baselines into one plot. This is a much faster way to look for global [RFI](#), but it must flag all baselines with each flag generated. [IBLED](#) has a different philosophy; it plots one baseline at a time in a graphical rather than gray-scale (image) fashion. It is able to average data over time, spectral channels, and/or IFs to make a more manageable amount of data and to measure the “decorrelation index” which is a measure of how variable the phase is over the averaging intervals. The capability of averaging IFs and displaying decorrelation may be of special interest for VLBI data sets. Otherwise, [IBLED](#) has been replaced in 15APR98 by [EDITR](#), which also uses the graphics planes rather than gray-scale images but which can plot multiple baselines to a chosen antenna and can display two data sets at the same time. This is obviously more useful for smaller arrays — *e.g.*, VLBI, MERLIN, and the Australia Telescope. This task is described below.

The editing task [WIPER](#) should be used with caution. It makes a plot like [UVPLT](#) of almost any parameter of a *uv* dataset against any other parameter. The plot is displayed on the TV and you may “wipe away” any points you do not like one point at a time or many at a time with a “fat brush.” The task will be very useful for fields with a well-behaved visibility function seen with good signal-to-noise. It may also be useful with data sets from which a fairly good [IMAGR](#) model has been subtracted with [UVSUB](#). It allows you to plot and edit any reasonable [STOKES](#) polarizations in one execution and displays which baselines enter into each plotted point.

In 31DEC16, the new task [FCCNT](#) may be used to count the number of correlators flagged by a particular flag table. Many of the tasks above can generate very large flag tables and can, if the choice of parameters was not right, flag way too much data. This task provides a way to check on this.

5.5.2 Baseline-based *uv*-data editing — EDITR

[EDITR](#) is a very effective editing tool from the beginning of data analysis on data sets with modest numbers of antennæ. Since it can display two data sets at the same time for comparison purposes, [EDITR](#) may also be used to good purpose with larger data sets during the self-calibration and imaging stage. The visibility amplitude or phase or the amplitude of the visibility with a running vector average subtracted may be displayed. The data for the selected baseline are shown in an edit window at the bottom of the display. Optionally, a second observable (*e.g.*, phase) from the selected baseline is shown in the same color in a window directly above the edit window. This option is controlled by the [DOTWO](#) adverb. Data for 0 to 10 other baselines to the selected antenna may be displayed in a different color in windows above these. A second *uv* data set may also be displayed along with the first. These data are not used for editing but may help you to select the data to be deleted. A “normal” choice for the second data set would be the residuals after Cleaning or [UVSUB](#). A menu-like control interface is available to select the data antenna and time range to be edited and to select various forms of editing. Instructions, explanations, informative messages, and the results of various functions appear in the standard *AIPS* message window. When prompted for information, such as an antenna number, type it into your normal *AIPS* input window (which is where the prompt message should have appeared).

[EDITR](#) is for editing continuum data from one or more IFs. Multiple spectral channels may be averaged on input with the vector average used for display and editing; multiple IFs are kept separate. The data may also be averaged over time as they are read into memory. This is useful for improved signal-to-noise, but will cause the data flags to be less selective in time. The program will allocate sufficient dynamic memory to

hold the selected data. If you have a very large data set and a modest computer, it would probably improve efficiency to limit the time range and run the task more than once to cover all time ranges.

To run it, enter:

- > **TASK 'EDITR ; INP C_R** to select the task and review the inputs.
- > **INDI n1 ; GETN ctn1 C_R** to select the 'TB' sorted *uv* single- or multi-source data set.
- > **DOCAL FALSE C_R** to apply no calibration. The SN or CL table from previous calibrations can be applied.
- > **FLAGVER fg1 C_R** to apply flag table *fg1* to the data on input.
- > **OUTFGVER 0 C_R** to write a new flag table containing version *fg1* and all new flag commands generated.
- > **SOLINT = Δt C_R** to have the data averaged over a time interval Δt minutes. If you do not want averaging, set this parameter to a small value; the default is $1/6000 = 0.01$ second. Editing times are recorded with an offset of **SOLINT**/2 which may cause confusion when no averaging was actually done.
- > **DETIME T C_R** to set the initial scan length estimate to *T* minutes (which can be changed later interactively) and to set the interval regarded as a break in the regular time sequence of the data. Setting this parameter suitably helps the program do a better display, but its exact value is not critical.
- > **CLR2NAME C_R** to display only one data set.
- > **DOTWO TRUE C_R** to display a second observable from the main baseline.
- > **CROWDED TRUE C_R** to allow all IFs and all polarizations to be displayed and edited at one time.
- > **DO3COL TRUE C_R** to use color to separate IFs and polarizations when more than one is displayed and edited at one time.
- > **INP C_R** to review the other parameters, which we assume here to be set to their null values.
- > **GO C_R** to run the task.

You can average the data over spectral channels (the default will average all channels present). IFs are edited separately; the default will include all IFs after which you can choose the one to edit interactively.

Since the display used by **EDITR** is very similar to the one used by **EDITA** displayed in § 4.4.2, we do not include a figure here; see Figure 4.2. The upper left corner of the display is reserved for displays of the selected data sample during editing while the bottom left corner is used for status information including flagging options. Menus, discussed below, appear down the left and right sides of the screen. The data are displayed in a stack of plots in the center of the screen. At the bottom are the data from the selected baseline in the primary observable; then the data from the primary baseline in a second observable (if **DOTWO** is true), and finally the data in the primary observable from 0–10 other baselines to the primary antenna. Data which have been flagged are shown in a different color. The data are plotted on a linear axis vertically, while the horizontal axis is monotonic but irregular in time. Tick marks are plotted at integer hours and the time interval of the edit area is indicated by times at the left and right ends of the axis. The time range displayed in all plots may be selected interactively and editing may therefore be done in crowded full time-range plots or in well separated short time-range plots. Surrounding the plot are various annotations describing the data plotted and the status of the various flags which control which data will be deleted on the next flagging command. If a second data set was specified, then data from that file are displayed in a different color in the same plot areas used for the primary data set.

The interactive session is driven by a menu which is displayed on the same screen as the data. Move the cursor to the desired operation (noting that the currently selected one is highlighted in a different color on

many TVs) and press button A, B, or C to select the operation. Press button D for a short explanation of the selected operation. The right-hand column contains options to select which data are displayed and to select which data are flagged on the next flag command. The menus are changed to adapt to the input data in order to avoid, for example, offering options to select IF in a one-IF data set. The left-hand column contains 7 interactive modes for editing the data plus options to set the display ranges and scan averaging length, to turn on error bars in plotting samples, to review, alter, and re-apply the existing flag commands, to defer or force a TV display, to switch to entering commands from the keyboard instead of the menu, and to exit with or without applying the current flag commands.

The right-hand menu can contain

NEXT CORRELATOR	To switch to viewing the next correlator, switching to the other polarization and, if needed, incrementing the IF.
SWITCH POLARIZ	To switch to viewing and editing the other polarization, cycles through both if CROWDED was true.
SWITCH ALL POL	To switch functions from applying to one polarization to applying to both polarizations or vice versa.
ENTER IF	To select which IF is viewed and edited. When CROWDED is true, zero means all.
SWITCH ALL IF	To switch functions from applying to one IF, to applying to a range of IFs, to applying to all IFs.
SWITCH ALL TIME	To switch FLAG ABOVE and FLAG BELOW between all times and the time range of the frame.
ROTATE ALL ANT	To rotate functions from applying to (a) one baseline, (b) all baselines to the main antenna, and (c) all baselines.
SWITCH ALL SOURC	To switch between flagging only the current source and flagging all sources.
ENTER ANTENNA	To select the main antenna, baselines to which are displayed on the screen.
ENTER OTHER ANT	To select up to 11 other antennas to define the baselines to be displayed; enter 11 numbers, 0's are then ignored (to plot 5 enter the 5 plus 6 0's). The first one is used for the edit area.
NEXT BASELINE	To advance the list of other antennas, selecting the next one for the edit area.
NEXT ANTENNA	To select a new main antenna, one higher than the current main antenna. The "others" will also be adjusted if appropriate.
PLOT ALL TIMES	To display all data for the selected baselines.
SELECT FRAME	To select a window into the current data interactively.
NEXT FRAME	To select the next time range window of the same size as the current frame.
PREVIOUS FRAME	To select the previous time range window of the same size as the current frame.
SHOW AMPLITUDE	To display and edit amplitudes.
SHOW PHASE	To display and edit phases.
SHOW DIFF AMPL	To display and edit the amplitudes of the vector difference between the sample and its running mean.
SHOW ALSO AMPL	To display amplitudes of the edit baseline for reference with the phase or difference amplitude edit window.

SHOW ALSO PHASE	To display phases of the edit baseline for reference with the amplitude or difference amplitude edit window.
SHOW ALSO DAMP	To display difference amplitudes of the edit baseline for reference with the phase or amplitude edit window.
TV ZOOM	To alter the display zoom used while in the flag functions.
OFF ZOOM	To turn off any zooming.
2ND UV OFF	To disable the display of the 2 nd uv data set.
2ND UV ON	To enable the display of the 2 nd uv data set.

The data displayed are of a single polarization, single IF, and 1–11 baselines to a single antenna. If **CROWDED** is true, then you may also choose to display and edit both polarizations and/or all IFs at the same time and input adverb **DO3COL** determines if polarizations and IFs are shown in different colors. The **NEXT CORRELATOR** cycles through all polarizations and IFs, show one at a time. The **SWITCH POLARIZATION** option switches the displayed polarization, the **ENTER IF** option prompts you if necessary for a new IF number, the **ENTER ANTENNA** option prompts you for a new primary antenna number, and the **ENTER OTHER ANT** prompts you for up to 11 other antenna numbers to select the main editing baseline and up to 10 secondary baselines to the primary antenna. (Note that you have to type in 11 numbers, but zeros are then ignored.) A flag command can apply to one or both polarizations and to one, some, or all IFs. It can apply to one baseline, to all baselines to the primary antenna, or to all baselines. The **FLAG ABOVE** and **FLAG BELOW** commands can apply only to the time range displayed in the data “frame” or they can apply to the full time range in the data set. The **SWITCH ALL POL**, **SWITCH ALL IF**, **ROTATE ALL ANT** and **SWITCH ALL TIME** options control these choices and the current state of these switches is displayed at the lower left of the TV screen. The task is able to zoom the display during interactive editing operations if you should need magnification to see what you are doing. The **TV ZOOM** and **OFF ZOOM** options let you control this. In larger data sets, however, a more useful display is obtained by interactively selecting a narrower time range with the **SELECT FRAME** option. To step forward and back through the frames, use the **NEXT FRAME** and **PREVIOUS FRAME** options, respectively. To display *uv* data amplitude, select **SHOW AMPLITUDE** and to display *uv* data phase, select **SHOW PHASE**. You may also display the difference between the current data sample and a running vector average of the data centered on the current sample and extending no more than plus or minus the “scan length” divided by two. To display the amplitude of the vector difference, select **SHOW DIFF AMPL**. Such displays are particularly sensitive to short-term problems while ignoring longer-term changes due to source structure. Since it takes time to compute things for, and display, the second data set, you may wish to turn it off part of the time. The **2ND UV OFF** and **2ND UV ON** options control this choice.

The left-hand menu can contain

FLAG TIME	To delete one time at a time.
FLAG TIME RANGE	To delete one or more time ranges.
FLAG BELOW	To delete all displayed times with data below a cutoff value.
FLAG ABOVE	To delete all displayed times with data above a cutoff value.
FLAG AREA	To delete one or more areas in the data-value <i>vs</i> time plane.
FLAG POINT	To delete one sample at a time using both horizontal and vertical cursor position.
FLAG QUICKLY	To delete samples using only mouse clicks
ENTER AMPL RNG	To select the display range for amplitude plots. Use 0 – 1 for zero to maximum, 00 for minimum to maximum.
ENTER PHASE RNG	To select the display range for phase plots.
ENTER DAMP RNG	To select the display range for plots of the amplitude of the visibility minus a running vector average visibility.
PLOT ERROR BARS	To plot error bars based on data weights.
SET SCAN LENGTH	To set the averaging time used to determine the running average in seconds.

LIST FLAGS	To list all flags now in the Flag Command table.
UNDO FLAGS	To undo one of the flag operations in the FC table
REDO FLAGS	To reapply all remaining flags after one or more have been undone
SET REASON	To set the “reason” string to be put in the <i>uv</i> -data flag table.
USE EXPERT MODE	To control the task from the keyboard instead of the menu.
HOLD TV LOAD	To stop updating the TV display with every change of parameter; change several, then select
DO TV LOAD	To update the TV display now and with each change of display parameter.
REPLOT	To do the current plot over again, recomputing the differences from the running mean if appropriate.
EXIT	To exit EDITR , moving the FC table to a <i>uv</i> -data FG table.
ABORT	To exit EDITR , deleting the FC table.

The first seven items select interactive flagging modes to delete all selected data at a single time, over a range of times, over all values below or above a specified value, or within a range of times and values (respectively). When one of these options is invoked, the screen zooms (if set to do so), a line or box appears in the editing window, and a display of the sample (source, time, value) under the cursor appears at the upper left. Follow the instructions in the message window to select and edit data. Note that this is a very good way to look at your data values even if you do not want to delete anything. The **FLAG QUICKLY** method is very efficient, but it requires caution in its use. Whenever the left mouse button is depressed, the sample closest to the cursor position is flagged. The next three options set the range of amplitudes, phases, and difference amplitudes displayed. These default to the full range in the data (separately and differently for each baseline) and can be set back to default by entering 0 0. The **SET SCAN LENGTH** option prompts you for a “scan” length in seconds used as the averaging interval for computing the running mean used in the difference displays. A longer scan length takes longer to compute, but is likely to be less noisy and more meaningful as an editing tool. If you are not using the difference display, set the scan length to a short interval. The running mean is not carried between sources and, as a result, is not normally carried across actual scan boundaries.

When you execute a flagging option, one or more lines are written to a flag command (FC) table attached to the input data set. If **EDITR** dies abnormally, this FC table can even be used in a later session. To list all of the flagging commands now in the table select the **LIST FLAGS** option. If you decide that you no longer want one of these flags, select **UNDO FLAGS** and enter the number (from **LIST FLAGS**) of the undesirable flag command. More than one flag command may apply to the same datum. After undoing flags, it is probably a good idea to select **REDO FLAGS** first to undo all remaining flags and then to reapply them to the data to make sure that everything is consistent. When the flag commands in the FC table are entered into a normal flag table, a 24-character “reason” is attached which is both descriptive and can even be used in **UVFLG** when removing entries in the FG table. The **SET REASON** command prompts you for the reason to be attached to subsequent flag commands. The default reason is the task name, time and date. Normally, **EDITR** updates the display whenever anything is changed. If you are about to change more than one display parameter (*i.e.*, polarization, IF, antenna, other antennæ, frame) before doing more editing, select **HOLD TV LOAD** to defer the display update until you select the **DO TV LOAD** option. If the display appears not to be current, select the **REPLOT** option. Finally, you may exit the program with the **EXIT** or **ABORT** options. The former applies your editing to a flag (FG) table attached to the input data set, while the latter discards any editing commands you may have generated.

Note that value-dependent flagging (**FLAG BELOW**, **FLAG ABOVE**, and **FLAG AREA**) use the values currently plotted to make a list of value-independent flag commands, namely a single time for the specified antennæ, IFs, polarizations, etc. When a value-dependent flag operation is undone with **UNDO FLAGS** or redone with **REDO FLAGS**, it is these value-independent flags which are undone or redone. You may have to undo more

commands and then repeat flag commands to get the results you could have gotten by doing the now desired value-dependent command in the first place. You need also to be careful with the **ROTATE ALL ANT** setting with these value-dependent commands. If one baseline is set, then the commands only apply to the current baseline. If one antenna is set, the commands apply to all baselines to the current main antenna, while if all antennas is set, the commands apply to all baselines. The first two set a clip level, below or above which data are deleted, based on the value of the observable in each baseline independently. The **FLAG AREA** command, however, only looks at the values of the observable in the main edit baseline and flags those samples from all applicable baselines.

Be careful when choosing **EXIT** versus **ABORT**. The former applies the flag commands to a flag table attached to the input uv data, the latter causes the flag commands to disappear without a trace. After **EXIT**, of course, one may use, edit, or ignore the output flag (FG) table. For single-source files, it may be necessary to run **SPLIT** to apply the FG table to the data since only some tasks know how to apply FG tables (those with **FLAGVER** as an adverb).

The colors used by **EDITR** are those of the various graphics planes when it begins to run. You may change them with the AIPS verb **GWRITE** to more desirable colors. The planes are:

Plane	Default	RGB	Use
1	1.00	1.00	0.00
2	0.06	1.00	0.00
3	1.00	0.67	1.00
4	0.00	1.00	1.00
5	1.00	0.18	0.18
6	0.60	0.60	1.00
7	1.00	0.80	0.40
			Second uv data set if present

You may wish to change the colors to ones that you can see better.

5.6 Imaging with OBIT

AIPS contains “verbs” that access programs in the **OBIT** software package written by Bill Cotton (bcotton@nrao.edu). If that package is installed on your computer, which is easy for Linux but not Mac systems, then you may use these programs. Two of the verbs are used to translate **EVLA** data into *AIPS* format; see § E.1. The other four verbs provide increasingly elaborate access to **OBIT**’s imager program. They are called **OBITMAP**, **OBITIMAG**, **OBITSCL**, and **OBITPEEL**. The first images a field of view in a single scale, the second allows many more imaging options including multi-scale, the third adds self calibration, and the fourth allows the self-calibration to be direction dependent. The *AIPS* group cannot take responsibility for the **OBIT** installation and programs, but much of the cleverness in *AIPS* was due to Bill and his later work deserves consideration. If nothing else, the tasks will run in a multi-threaded fashion which *AIPS* does not.

6 Displaying Your Data

This chapter is concerned with the ways in which you may display your data. There are a number of tasks for generating “plot files” which contain graphics commands for the making of various displays of your *uv* and image data. All of these now offer a “preview” option to draw the plot directly on the *AIPS* TV, rather than putting the commands into a file. Once the files are created, a variety of tasks may be used to translate them into displays on various devices, such as the *AIPS* TV and graphics windows and PostScript printers. There are also verbs to display and manipulate your images on the *AIPS* TV and a single task [TVCPS](#) to capture that display, if desired, into a PostScript file for printing, recording on film, or even including in your scientific papers.

Several indices of the *AIPS* software are relevant to this discussion. To generate current lists of *AIPS* functions on your workstation window (or terminal) use [ABOUT HARDCOPY](#) C_R, [ABOUT INTERACT](#) C_R, [ABOUT PLOT](#) C_R, [ABOUT TV-APPL](#) C_R, and [ABOUT TV](#) C_R. Recent versions of these indices are reproduced in Chapter 13 of this *CookBook*.

6.1 Getting data into your *AIPS* catalog

By the time you reach this chapter, most of your data will probably already be loaded into your *AIPS* catalog either by reading an external tape or disk or by being generated by some *AIPS* task. Visibility data which are not presently on disk may be read by the *AIPS* tasks [FILLM](#), [UVLOD](#) and [FITLD](#); see § 4.1 and § 5.1 for details. Images that are generated by other imaging systems, (e.g., images from non-NRAO radio telescopes or non-radio images) can be transported to *AIPS* by writing them out of the other imaging system on tape or disk in the standard FITS format. The tasks [IMLOD](#) and [FITLD](#) can then be used to read them into *AIPS*. These tasks are also used to read images saved with [FITTP](#) and [FITAB](#) from previous *AIPS* sessions.

6.1.1 IMLOD and FITLD from FITS-disk

[FITLD](#) and [IMLOD](#) can read FITS-format images from external disk files or magnetic tape into your *AIPS* catalog. The disk-file option is indicated by setting the adverb [DATAIN](#) to a non-blank value. Disk image files must be read in only one at a time per execution of [IMLOD](#), but [FITLD](#) can read more than one FITS-disk file if the file names are identical except for sequential post-pended numbers beginning with 1. [DATAIN](#) is a string of up to 48 characters that must completely specify the disk, directory, and name of the input disk file to your computer’s operating system. See § 3.10.3 for a discussion of FITS-disk files.

One “feature” of *AIPS* complicates this otherwise straightforward disk analog of FITS tape reading. AIPS translates all of your alphabetic inputs to upper case (this was demanded by users who otherwise became confused between upper and lower cases).¹ So if your computer distinguishes upper and lower cases for disk, directory, or file names, you probably should do two things to prepare for this before running *AIPS*. First, restrict your external disk file names to upper-case characters and numbers. Second, set an upper-case “environment variable” or “logical” to point to the disk and directory where your FITS-disk images are stored before you run AIPS. You may need help from your System Manager when doing this for the first time. A common strategy on UNIX machines is to create an upper-case logical name after logging in but before starting up AIPS:

```
% setenv MYLOGICAL myarea CR if using C-shell, or
```

¹If you omit the close quote on the character string, it is not converted to upper case, allowing you to circumvent this *AIPS* limitation. The string without a close quote must, of course, be the last thing on the line.

\$ export MYLOGICAL=*myarea* C_R if using korn, bourne, or bash shells,
 where MYLOGICAL is an all-upper-case string of your choice and *myarea* is the full path name of the disk directory that contains your FITS-disk data. *AIPS* usually provides a public disk area known as FITS which you may use.

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

> **DATAIN** 'MYLOGICAL:IMAGE.DAT' C_R to read in the FITS-disk file *myarea*: IMAGE.DAT.

To check that the file name is correct, type:

> **TPHEAD** C_R

Your terminal will then list information about the image header of the disk file.

> **OUTDI** *n* C_R to specify writing the image to your *AIPS* catalog on disk *n*.
 > **OUTNAME** '*your-chosen-name*' C_R to specify the output disk file name in *AIPS*; the default is the image name on tape if **FITTP** was used to write the image to tape.

The string *your-chosen-name* can be any (\leq 12-character) title that you want to use as the image name within *AIPS* and should be specified for images from other image-processing software systems. **FITLD** also allows you to specify the 6-character image “class” parameter. Use **OUTCLASS** ‘*abcdef*’ C_R, if you wish to change the class from that in your input file as the image is read or if the image comes from a “foreign” system.

> **OUTS** -1 C_R to keep the sequence number the same as that in the file; the default is the highest unique number for images with this name and class in your current *AIPS* catalog.
 > **NCOUNT** *m* C_R in **FITLD** only, to load *m* images consecutively starting with the file specified by **DATAIN** with a 1 post pended. The other files must be the same name but with sequential numbers post pended. If you use this option, do *not* specify the **OUTNAME** unless you want the same name for all the new images in your catalog.
 > **GO FITLD** C_R or **IMLOD**, to run the task.

If **OUTNAME** is left unspecified, it defaults to the “name” of the image read from the FITS header — either the name previously used in earlier image processing or the source name. If **OUTCLASS** is unspecified, it defaults to the Class previously used in earlier image processing or to a compound name (e.g., IMAP, IBEM, QMAP, ICLN) which attempts to describe the image. These defaults are frequently good ones when you are loading multiple consecutive images with **NCOUNT** $>$ 0. You may of course change the *AIPS* image and class names later by using **RENAME** (see § 3.3.3 of this *CookBook*).

IMLOD and **FITLD** can also read images from magnetic tape; see the instructions in § 3.9. Use the verbs **MOUNT** to mount the tape, **AVFILE** to advance the tape to the desired file, and **TPHEAD** to check that the tape is correctly positioned. Set **NFILES**=0, **NCOUNT** to the number of consecutive tape files you wish to load, and then say **GO** to either **FITLD** or **IMLOD**.

6.1.2 Image coordinates

Images of the celestial sphere must be rendered with some sort of coordinate “projection.” *AIPS* pioneered the handling of such coordinates, supporting true projective coordinates called **SIN** (orthographic), **TAN** (gnomic), **ARC** (zenithal equidistant), and **STG** (stereographic) as well as a special version of orthographic suited for East-West interferometers (NCP). In 31DEC13, *AIPS* also supports modern² interpretations of GLS (SFL Sanson-Flamsteed), MER (Mercator), AIT (Hammer-Aitoff), CAR (Plate carrée), MOL (Molweide), and PAR

²Calabretta, M. R. and Greisen, E. W. 2002, “Representations of celestial coordinates in FITS,” *Astr. & Ap.*, 395, 1077.

(parabolic) “projections.” These are normally used to represent large sections of the celestial sphere. Users should be warned that reference pixel values other than zero lead to *oblique* projections.

Fits to the coordinates on optical images often lead to a modest amount of skew in the image. This is represented in the `PCi,j` or `CDi,j` FITS header cards. If these cards contain any significant skew, `IMLOD` and `FITLD` will tell you about it. In such cases, run `DSKEW` in 31DEC13 on the images after they have been loaded into *AIPS*. This is the only task that understands coordinate skew and it will re-grid the images for use by all other *AIPS* tasks.

6.2 Printer displays of your data

The most old fashioned way to look at your data — and the most exact — is simply to print it out and read the numbers. *AIPS* provides a variety of tasks and verbs to print visibility data, image data, tabular data, and miscellaneous other information. All of these tasks and verbs allow you to specify where the printed output goes using two adverbs, `DOCRT` and `OUTPRINT`. If `DOCRT` ≤ 0 and `OUTPRINT` is blank, then the output is placed in a temporary file and queued to the printer you selected when starting *AIPS*; see § 2.2.3. Beginning with 31DEC16, if printing directly to the printer is requested, most tasks and verbs count the number of lines of print and request permission to proceed if the count is large. (Type `PRINTER n CR` to change the line printer selection to that numbered *n*; type `PRINTER 999 CR` to see the devices available to you and their assigned numbers.) If `DOCRT` ≤ 0 , a non-blank `OUTPRINT` specifies a text file into which the output is to be written; see § 3.10.1. The current output is appended to the file if it already exists. Thus, you can combine a number of printed outputs for later editing and/or printing. When `DOCRT` = -1, the output print file will contain full paging commands and headers. To suppress some of this, use `DOCRT` = -2 or to suppress almost all of it, use `DOCRT` = -3. This last is especially helpful when writing programs to read the text file. If `DOCRT` > 0, the output is directed to your workstation window or terminal. All printer verbs and tasks are able to respond to both the width and height of your workstation window. Set `DOCRT` = 1 to use the current width; set `DOCRT` = *n* ≥ 72 , to use *n* as the width of the display window. Since most print tasks display more information on wider windows, we recommend widening your window to 132 characters and specifying `DOCRT` = 1. The print routines will pause whenever the screen is full and offer you the choice of continuing or quitting. Thus, you can start what might be a very long print job, find out what you wanted to know after a few screens full, and quit without using up any trees.

6.2.1 Printing your visibility data

Before beginning calibration, it is a very good idea to make a -summary list of the contents of your data set. `LISTR` with `OPTYPE = 'SCAN'` will list the contents of each scan in the data set. `DTSUM` also produces a listing summarizing the data set in either a condensed or full form.

The most basic display of your visibility data is provided by `PRTUV` which lists selected correlators in the order they occur in the data set:

- > `TASK 'PRTUV' ; INP CR` to review the inputs.
- > `INDI n ; GETN ctn CR` to select the disk and data set to print.
- > `CHANNEL c ; BIF 1 CR` to print starting with channel *c* from IF 1.
- > `BPRINT m; XINC i CR` to print every *i*th visibility starting with the *m*th visibility in the data set.
- > `DOCRT 1 ; GO CR` to run the task with display on the terminal.

When you have seen enough, enter `q CR` or `Q CR` at the page-full prompt.

You may limit the sources, range of projected baselines, and times displayed and may select only one baseline or one antenna. [PRTUV](#) does not apply calibration or flagging tables. To get a similar display with all “standard” calibration, flagging, and data selection (optionally) applied, use the task [UVPRT](#). [LISTR](#) also uses all of the calibration options to list the data in simple lists or in a display showing all the baselines at each time in a matrix form. [SHOUV](#) also lists calibrated visibility data with options to average all channels in each IF and to display closure rather than observed phases. [ANBPL](#) converts baseline-based amplitudes, phases, or weights into antenna-based values and prints and/or plots them. The display of antenna-based weights before and after amplitude self-calibration is a particularly useful tool for spotting calibration/instrumental problems.

There are a number of tasks used to diagnose possible problems in your data and to print information about them. [UVFND](#) examines a data set for excess fluxes, excess apparent V-polarization, or simply any data with a specified fringe spacing and position angle or a specified range in u and v . As it does this, it also checks for bad antenna numbers, bad times, and (optionally) bad data weights. [CORER](#) examines a data set for excessive mean values and rms in each correlator (after applying calibration and flagging and then subtracting a point source at the origin). [RFI](#) examines the rms fluctuations in the real and imaginary visibilities of each correlator looking for (and reporting) periods of apparent RF interference. [UVDIF](#) directly compares two data sets reporting any excess differences. It is useful for determining whether your latest operations (flagging, self-cal) have made a significant (or any) difference.

6.2.2 Printing your image data

The most basic display of an image is a print out of the numbers it contains. Such a display is provided by [PRTIM](#):

- | | |
|---|--|
| > TASK 'PRTIM' ; INP C_R | to review the inputs. |
| > INDI n ; GETN ctn C_R | to select the disk and image to print. |
| > NDIG 3 C_R | to use 3 digits, printing numbers between -99 and 999 with appropriate power of 10 scaling. |
| > FACTOR 10 C_R | to raise the default scaling by a factor of 10, overflowing regions of high values to see low valued regions better. |
| > BLC 0 ; TRC 0 C_R | to see the whole image. |
| > XINC 2 ; YINC 2 C_R | to see every other column and every other row. |
| > DOCRT FALSE ; GO C_R | to print the image on the selected printer. |

Other imaging tasks which can use the printer are [BLSUM](#) and [ISPEC](#), which compute and print spectra by summing over regions of each plane in a data cube (see § 8.6), and [IMFIT](#), [JMFIT](#), and [SAD](#), which fit one or more Gaussians to an image (see § 7.5). [IMTXT](#) writes an ASCII-formatted file containing an image.

6.2.3 Printing your table data

If you have any doubts about the contents of tables in *AIPS*, it is best to resolve them by looking at the contents of the tables involved. [PRTAB](#) is a very general task which will print the contents of any *AIPS* table file. For example, to print flag table version 1:

- | | |
|---|--|
| > TASK 'PRTAB' ; INP C_R | to review the inputs. |
| > INDI n ; GETN ctn C_R | to select the disk and catalog entry to print. |
| > INEXT 'FG' ; INVERS 1 C_R | to select flag table version 1. |
| > BPRINT 0 ; EPRINT 0 ; XINC 1 C_R | to print everything. |
| > DOHMS TRUE C_R | to print times in sexagesimal notation. |

> **DOCRT 1 ; GO** to print the flag table on the terminal.

When you have seen enough, enter **q C_R** or **Q C_R** at the page-full prompt. For a table with a significant number of columns, **PRTAB** shows all rows for the first columns and then loops for the next set of columns. To see all columns for some rows, set a low **EPRINT** value or be very patient. Enter a list of column numbers in **BOX** to see only some of the columns. **NCOUNT**, **BDROP** and **EDROP** control which values are displayed in those columns having more than 1 value per row. Adverb **RPARM** lets you limit the display to rows having specific column values within specified ranges, while **NDIG** controls the format and accuracy used to display floating-point columns.

Some of the tables have specialized printing programs. These include **PRTAN** for antenna tables, **PRTCC** for Clean component tables, **SILPRT** for slice files, and **LISTR** with **OPTYPE = 'GAIN'** for calibration, solution, and system temperature tables. The verb **EXTLIST** will list information about various extension files, particularly slice and plot files (see below), which may be printed with **PRTMSG**. **OFMLIST** is a verb to print the contents of an *AIPS* TV color table. Finally, task **TBDIF** will compare columns of two tables and print information about their differences.

6.2.4 Printing miscellaneous information

There is a variety of miscellaneous information which may also be sent to the printer in the same way. Verb **PRTMSG** prints selected contents of the *AIPS* message file; see § 3.2. Verb **PRTHI** prints selected lines from a history file; see § 3.4. Pseudoverb **ABOUT** prints lists of *AIPS* symbols by category while pseudoverbs **HELP** and **EXPLAIN** print information about a selected symbol; see § 3.8. Task **PRTTP** prints the contents of magnetic tape volumes and pseudo-tape disk files; see § 5.1.1 and § 3.9.4. Task **PRTAC**, which may also be run in a stand-alone mode, prints information selected from the *AIPS* accounting file.

Task **TXPL** will attempt to represent an *AIPS* plot file (see below) on the printer. This will not work well for complicated plots, but, for simple plots, it may be the only way someone running over a slow telephone line can see his/her data in plot form.

6.3 Plotting your data

The basic concept in *AIPS*' plotting is to use some task to create and write a device-independent plot file as a PL extension file to a catalogued image or visibility data set and then to use some device-dependent task to interpret that file for the desired output device. Plot files are not overwritten by subsequent plot tasks. Instead they make new plot files with higher "version" numbers. The device-dependent tasks include **TVPL** (*AIPS* TV devices including **XAS**), **TKPL** (Tektronix graphics devices including *AIPS*' **TEKSRV** server), **TXPL** (line printers), and **LWPLA** (PostScript printer/plotters). Tasks called **PRTPL**, **QMSPL**, and **CANPL** support antique Versatec, QMS, and Canon printer/plotters. To plot on a PostScript printer/plotter:

- > **TASK 'LWPLA' ; INP C_R** to review the inputs.
- > **INDI n ; GETN ctn C_R** to select the disk and catalog entry to print.
- > **PLVER m ; INVERS 0 C_R** to plot the *m*th plot file only.
- > **OUTFILE '' ; GO C_R** to do the plot immediately.

LWPLA offers the option to save the file for later plotting or inclusion as encapsulated PostScript in other documents. It also has options to control scaling, output paper size, width and darkness of lines, and transformation of grey-scale intensities. It can write more than one plot file at a time and can append new plots to existing output files. Multi-plot files are not "encapsulated" but may be printed and viewed with tools such as *gv* or *ghostview*. Note that PostScript files are text files and *AIPS* writes particularly simple PostScript so that it can be modified by the users. See **HELP POSTSCRIPT** for suggestions including

information on deleting and adding labels and arrows and on converting the PostScript to other formats like jpg without loss of resolution.

LWPLA and all plot tasks offer some “coloring” options. These are illustrated in the color pages at the end of this chapter. The grey-scale plotting tasks, including **GREYS**, **PCNTR**, and **KNTR**, can now enhance the grey-scales with a transfer function and then pseudo-color them with a color table. See § 6.4.3 for a short discussion of “output-function memory” tables which may be read into the above tasks or to **LWPLA** using the adverb **OFILE**. Lines plotted on top of grey scales (*e.g.*, contours, polarization vectors, stars) may be “dark” when the grey scale intensity is high. **LWPLA** may be instructed to plot these as bright if adverb **DODARK** is false. All plot programs can draw lines of different types in both bright and dark forms. In **LWPLA**, if **DOCOLOR** is true, the array adverb **PLCOLORS**(*i, j*) controls the red, green, and blue colors (*i* = 1, 2, 3, resp.) of line types *j* = 1 – 10. The normal meanings of these types are:

1. Bright labeling, tick marks, surrounding lines
2. Bright lines, usually contours or model curves
3. Bright lines, usually polarization vectors
4. Bright lines, usually symbols such as stars, visibility samples
5. Dark labeling text inside plot area
6. Dark lines, usually contours
7. Dark lines, usually polarization vectors
8. Dark lines, usually symbols such as stars
9. Bright labeling outside the main plot area, *e.g.* titles, tick values and types, documentation
10. Background for the full plot

Some plot tasks have the ability to control the colors of their line drawing independent of these line types. These colors may be controlled only when making the plot file with the particular task. **PCNTR** and **KNTR** have acquired the ability to color each contour level under control of the adverb **RGBLEVS**. Using system **RUN** file **SETRGBL**, procedures **CIRCLEVS**, **RAINLEVS**, **FLAMLEVS**, and **STEPLEVS** are available to help you set the values of **RGBLEVS**. Examples are shown on the color pages at the end of this chapter. With **LWPLA** and these options one may prepare extremely effective displays — or hopelessly bad ones — for use in talks and, since the prices have become reasonable, even in journals. Note that most journals want color images in CMYK (cyan-magenta-yellow-black) rather than RGB; use **DPARM(9) = 1** in **LWPLA** to get PostScript files with this color convention. Note that these two color representations usually require different “gamma” corrections; the adverb **RGBGAMMA** allows this control in **LWPLA**.

All *AIPS* plot tasks now offer a “preview” option. If you set **DOTV = TRUE** when running any plot task, then the plot appears immediately on the *AIPS* TV display and no plot file is generated. This option allows you to make sure that the parameters of the plot are reasonable and lets you avoid making files and wasting paper for quick-look plots. Additional options allow you to control which graphics channel is used for the line drawing (**GRCHAN**) and to select pixel scaling of the plot at your specified location on the TV screen (**TVCORN**). These two options allow you to view more than one plot at a time on the TV, usually for purposes of comparison. Each graphics channel on the TV has a different color and a complementary color is used when two or more channels are on at the same point. This allows for a fairly detailed and effective comparison of plots, all of which may be captured with task **TVCPS** (see below). Be aware that most tasks now interpret **GRCHAN = 0** as an instruction to use graphics channels 1 through 4 for line types 1 through 4 and graphics channel 8 for dark vectors. The comparison function is only achieved by specifying **GRCHAN**. Tasks that produce multiple plot files pause for 30 seconds at the end of each plot when **DOTV = TRUE**. This

6.3. Plotting your data

6. Displaying Your Data

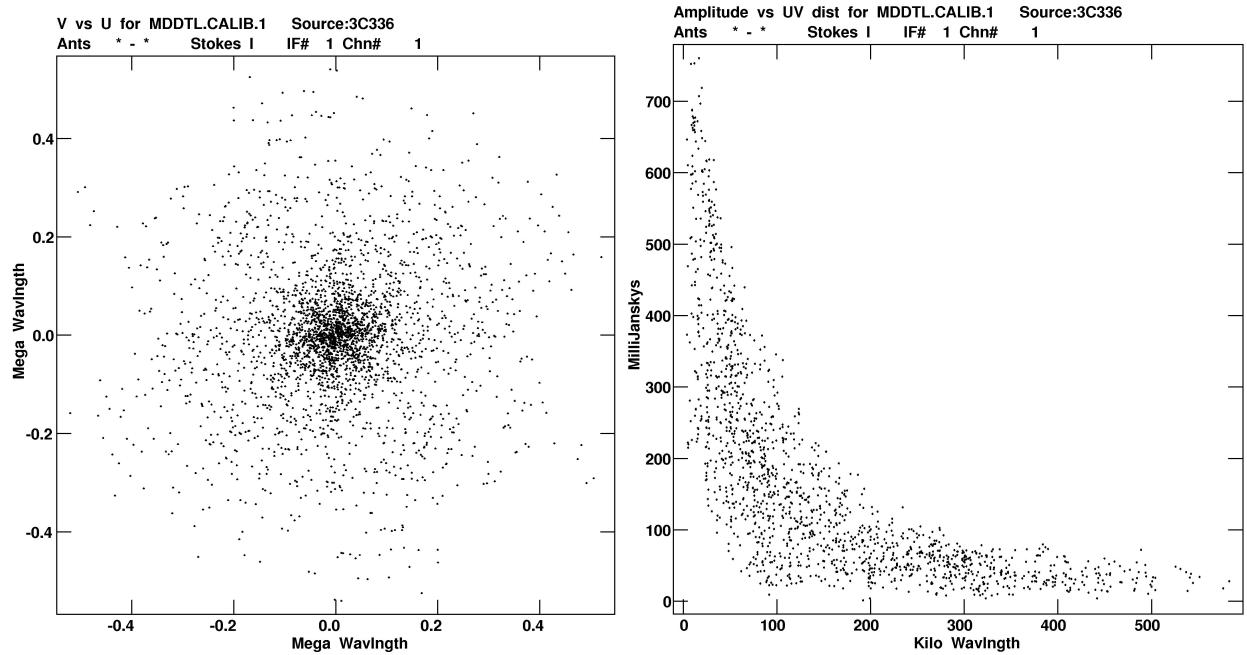


Figure 6.1: UVPLT displays of where the data were observed in the uv plane (left) and of the visibility amplitudes as a function of baseline length (right). The data on 3C336 were provided by Alan Bridle from observations made with the [VLA](#) on 6 December 1987.

allows you to stop the task (TV button D), hurry it along (TV buttons B or C), or make it pause indefinitely (TV button A) until another TV button is pressed.

You can review the parameters of the plot files associated with a given image or visibility data set by typing:

- > **INDI** *n* ; **GETN** *ctn* C_R to select the disk and catalog entry to print.
- > **INEXT** 'PL' ; **EXTLIST** C_R to list summaries of the plot file contents.
- > **PLVER** *m* ; **PLGET** C_R to recover all adverb values used when making the specified plot file.

Plot files (and other “extension files”) are automatically deleted when an image is deleted by **ZAP**. However, large plot files should be deleted as soon as they are no longer needed:

- > **INP EXTDEST** C_R to review the inputs required.
- > **INEXT** 'PL' ; **INVERS** *m* C_R to set the type to PL (plot) and the version number to be deleted to *m*. *m* = -1 means all and *m* = 0 means the most recent (highest numbered).
- > **EXTDEST** C_R to do the deletion.
- > **INVERS** 0 C_R to reset the version number to its default — usually advisable.

Plot files are not amenable to the FITS format and so are not written by **FITTP** and **FITAB**. They may be copied from one catalog entry to another with **TACOP**.

6.3.1 Plotting your visibility data

The most basic plot program for visibility data is called **UVPLT**. It allows you to select the *x* and *y* axes of the plot from real, imaginary, amplitude, log of amplitude, phase and weight of the visibility, time, hour angle, elevation, azimuth, and parallactic angle, and projected baseline length, position angle, *u*, *v*, *w*, frequency,

and spectral channel number. It offers all of the usual calibration and data selection options and it plots the selected points individually and/or in a controlled number of bins along the x axis. It can display multiple IFs and spectral channels at once, including averaging groups of spectral channels. For example, to plot calibrator phases as a function of time for all baselines to one antenna:

> TASK 'UVPLT' ; INP C _R	to review the inputs.
> INDI <i>n</i> ; GETN <i>ctn</i> C _R	to select the disk and catalog entry of the data set.
> BPARM = 11 , 2 C _R	to plot time in hours on the x axis and phase in degrees on the y axis.
> SOURCES '' ; CALCODE '*' C _R	to select all calibrator sources.
> XINC 4 C _R	to plot only every fourth selected sample.
> ANTENNA 2,0 ; BASELINE 0 C _R	to do all baselines with antenna 2.
> DOCRT = -1 ; GO C _R	to make a plot file of these data.
After UVPLT is running, or better, after it has finished:	
> PLVER 0 ; GO LWPLA C _R	to plot the latest version on a PostScript printer/plotter.

There are several other tasks to plot your visibility data. **VPLOT** plots all of the parameters offered by **UVPLT**, but one baseline at a time with multiple baselines per page (*i.e.*, per plot file) and multiple pages per execution. **CLPLT** is a similar task but restricted to plotting closure phases around baselines involving 3 antennas as a function of time. **CAPLT** is a similar task but restricted to plotting closure amplitudes around groups of 4 baselines as a function of time. All three of these can also plot a source model (based on Clean components) as well as the observations. **ANBPL** can create a similar plot of amplitudes, phases, or weights converted to antenna-based quantities; this function is helpful in diagnosing problems with your data. For spectral-line users, **POSSM** plots visibility spectra averaged over selected baselines and time intervals. It can also plot bandpass calibration tables and the Fourier transform of visibility spectra (the auto- and cross-correlation functions). For VLBI users (primarily), **FRPLT** plots visibilities versus time or, more importantly, the fringe rate spectrum. To examine the statistical distribution of your data, try **UVHGM** which plots histograms showing the number of samples or weights versus a wide variety of parameters. Optionally, it will fit a Gaussian to the histogram and plot the result. The task **UVIMG** will grid visibility data into an image on a variety of axes, while the task **UVHIM** will make an image of a two-dimensional histogram of your *uv* data. All image display functions below may then be used to view the results. For example, if the axes of the histogram image are the real and imaginary parts of your data, then the image will demonstrate the amplitude and phase stability present (or missing) in your data.

Two plot programs actually convert visibility data to the image plane for plotting. Observers of point objects which might vary with time either intrinsically or by scintillation (*e.g.*, stars, masers) might wish to try **DFTPL**, which plots the Fourier transform of the data shifted to a selected position as a function of time. In 31DEC17, task **DFTIM** makes an image of such data as a function of frequency and time. VLBI spectral-line observers may need to use **FRMAP**, which performs imaging via fringe-rate inversion and plots the loci of possible source positions.

6.3.2 Plotting your image data

Image data may be drawn in a variety of ways including contours, grey or color levels, row tracing, and statistical. All plots are drawn with labeled tick marks although these may be suppressed with the **LTYPE** parameter. This **LTYPE** parameter is used to control the type of axis labels, the degree of extra labeling, the presence of the line giving the plot time and version, and also how metric scaling is done on the axis units. Read **HELP LTYPE** for details. For plots having significantly non-linear coordinate axes, *e.g.*, wide-field images, it is sometimes useful to draw a full, non-linear coordinate grid rather than just short lines at the edges of the plot. Tasks like **CNTR** and even the verb **TVLABEL** offer this option; enter **DOCIRC TRUE** C_R.

Plot symbols (e.g., plus signs) may be drawn on the plots produced by **CNTR**, **PCNTR**, **GREYS**, **KNTR** and several of the other tasks mentioned below. In these tasks, the parameters which controls the plotting are **STFACTOR**, a scale factor for the symbols, and **STVERS**, a choice of the desired stars extension file. When using this option, there must be a table of “star” positions associated with the image being plotted. To create one, enter **EXPLAIN STARS** **C_R** to learn the format of the input data file and the parameters for the task. See also Appendix Z or your local equivalent for instructions on editing text files. A star file may also be created by **MF2ST** from a model fit file produced by task **SAD** (see § 7.5.3 and § 10.4.4). Tasks **IMFIT**, **JMFIT**, and **SAD** may now also produce “star” extension files directly.

Example outputs of the following three tasks are given in Figure 6.2.

6.3.2.1 Contour and grey-scale plots

The most basic contour drawing task is **CNTR**. In addition to the usual image selection parameters, you may specify:

- | | |
|---|---|
| > TASK 'CNTR' ; INP C_R | to tell you what you may specify. |
| > BLC 250 , 230 C_R | to set the bottom left corner of plot at 250, 230 (in pixels with 1,1 at extreme bottom left of the image). |
| > TRC 300, 330 C_R | to set the top right corner of plot at 300, 330. |
| > CLEV 0 ; PLEV 1 C_R | to get contour levels at 1% of the peak image value. |
| > PLEV 0 ; CLEV .003 C_R | to get contour levels at 3 mJy. |
| > LEVS -1, 1, 2, 4, 6 C_R | to get actual contours at -1, 1, 2, 4, and 6 times the basic level set by PLEV or CLEV . LEVS need not be integers, but very fine subdivisions cannot be represented accurately on the plot. |

N.B., if you request more than one negative level with the **LEVS** input, you *must* use commas between the negative levels. Otherwise the minus sign(s) will be treated as subtraction symbols and the desired levels will be combined into a single negative level by the AIPS language processor. **BLC** and **TRC** can be initialized conveniently from the TV display using the cursor with the **TVWIN** instruction (see § 6.4.4). Then check:

- | | |
|----------------------------|--|
| > INP C_R | to review what you have specified. |
| > GO C_R | to run the task when you’re satisfied with the inputs. |

This generates a plot file as an extension to your image file, with the parameters you have just specified. Watch the *AIPS* monitor (which, on some systems, is your terminal) to see the progress of this task. If the “number of records used” in the plot file is over 200, the contour plot will be messy (unless the field is also large). In this case, check that you have not inadvertently set **PLEV** or **CLEV**, for example, to unrealistically low values. Printing a large, messy plot file on the printer can take a considerable length of time and will inconvenience other users. Consider plotting directly on the TV first (**DOTV = TRUE**) to check on your selection of contours.

PCNTR plots polarization vectors on top of contours and/or grey-scales. You may make a polarized-intensity image and a polarization position-angle image from the Q and U images (see § 7.1.2) or use the Q and U images themselves. Then:

- | | |
|--|---|
| > TASK 'PCNTR' ; INP C_R | to review the input parameters. |
| > INDI n1 ; GETN ctn1 C_R | where <i>n1</i> and <i>ctn1</i> select the disk and catalog numbers of the image to be contoured. |
| > IN2DI n2 ; GET2N ctn2 C_R | where <i>n2</i> and <i>ctn2</i> select the Q or polarized intensity image. |
| > IN3DI n3 ; GET3N ctn3 C_R | where <i>n3</i> and <i>ctn3</i> select the U or position-angle image. |
| > IN4DI n4 ; GET4N ctn4 C_R | where <i>n4</i> and <i>ctn4</i> select the grey-scale image. |
| > PCUT nn C_R | to blank out vectors less than <i>nn</i> in the units of polarized intensity. |

> ICUT <i>mm</i> C _R	to blank out vectors at pixels where the total intensity (image 1) is less than <i>mm</i> in the units of image 1.
> DOFRAC 1 C _R	to convert the polarization lines to fractional polarization, dividing by the I image, before plotting.
> FACTOR <i>xx</i> C _R	to set the length of a vector of 1 (in units of total or fractional polarization) to <i>xx</i> cell widths.
> DOCONT 1 ; DOVECT 1 ; DOGREY 4 C _R	to request vectors plus contours of image 1 and grey scale of image 4.
> PIXRAN <i>T_{min}, T_{max}</i> ; FUNCTYP '' C _R	to scale linearly the grey-scale values from <i>T_{min}</i> to <i>T_{max}</i> .
> OFMFILE 'RAINBOW' C _R	to use the standard rainbow-colored OFM table to pseudo-color the grey scales.
> CBPLOT 1 C _R	to plot the Clean beam in the lower left corner. See HELP CBPLOT for numerous options.
> INP C _R	to review your inputs and remind you of others. Most are similar to those in CNTR and sensibly defaulted.
> GO C _R	to generate the plot file, which can then be routed to output devices via TKPL , TVPL , LWPLA etc.

Unless images 2 and 3 are of Q and U polarization, the lengths of the vectors are controlled by image 2 while the directions of them are controlled by image 3. Clearly this program can also be used for other combinations of images, so long as one of them represents an angle. [PCNTR](#) and following tasks have the option to draw an image of the Clean beam under control of adverb [CBPLOT](#). Polarization vectors may be plotted with the color representing the angle. The value of [POL3COL](#), if greater than zero, is that angle represented in pure red from 0 to 180 degrees. A color “spray” is plotted to calibrate the eye. The ability to plot multiple spectral planes in colored contour or polarization vectors was also added. The adverb [CON3COL](#) controls this function. The ability to select the colors of each contour level with [RGBLEVS](#) has been added. These color functions are displayed at the end of this chapter in the color pages.

[GREYS](#) creates a plot file of the grey-scale intensities in the first input image plane and, optionally, a contour representation of a second input image plane. Like the other grey-scale plotting tasks, [GREYS](#) can interpret a true-color (RGB) image cube in its “true” colors. Unlike the others, it can construct the true-color image from 3 separate image planes. A sample set of inputs could be:

> TASK 'GREYS' ; INP C _R	to review the inputs.
> DOCOLOR 1 C _R	to specify that a “true-color” image is to be plotted.
> INDISK <i>n1</i> ; GETN <i>ctn1</i> C _R	to select the red image.
> IN3DISK <i>n3</i> ; GET3N <i>ctn3</i> C _R	to select the green image.
> IN4DISK <i>n4</i> ; GET4N <i>ctn4</i> C _R	to select the blue image.
> PIXRAN <i>T_{minr}, T_{maxr}</i> ; FUNCTYP 'SQ' C _R	to scale by a square-root function red values from <i>T_{minr}</i> to <i>T_{maxr}</i> .
> APARM <i>T_{ming}, T_{maxg}, T_{minb}, T_{maxb}</i> C _R	to scale green and blue values similarly over ranges <i>T_{ming}</i> to <i>T_{maxg}</i> and <i>T_{minb}</i> to <i>T_{maxb}</i> , respectively.
> BLC 250 , 250 , 3 C _R	to select the lower left corner and the plane in the first image.
> TRC 320 , 310 , 12 C _R	to select the upper right corner in the first image and, with TRC(3) , the plane in the second image.
> DOCONT TRUE C _R	to specify that contours are to be drawn.
> IN2D <i>n2</i> ; GET2N <i>ctn2</i> C _R	to select the contour image.
> PLEV 0 ; CLEV 0.005 C _R	to select 5 mJy/beam contour increments.
> LEVS -3 , -1 , 1 3 10 30 100 C _R	to plot contours at -15, -5, 5, 15, 50, 150, and 500 mJy/beam.
> DOWEDGE 2 C _R	to plot a 3-color step-wedge along the right-hand edge; 1 for along the top and 0 for no wedge.

6.3. Plotting your data

6. Displaying Your Data

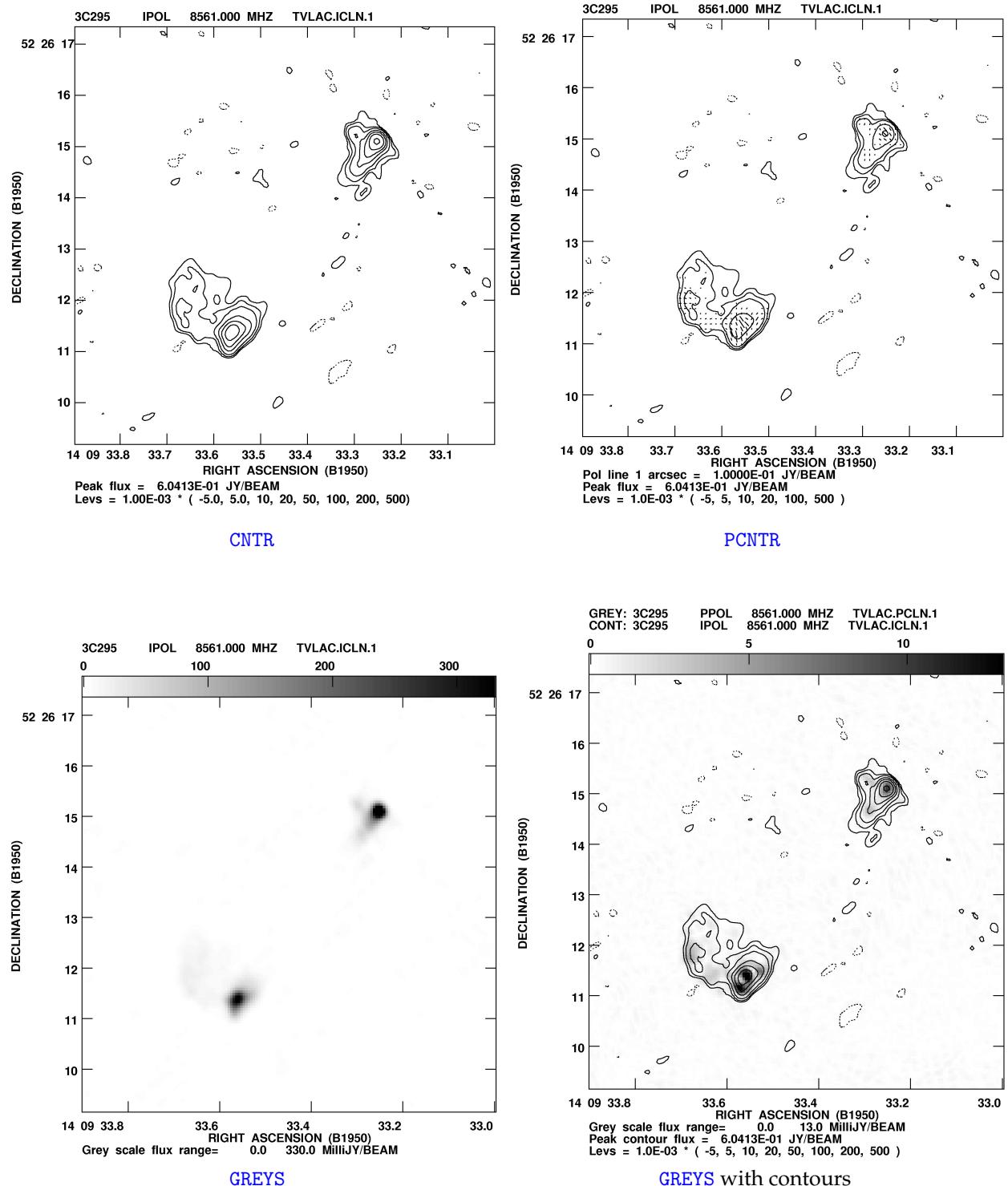


Figure 6.2: Contour, polarization, and grey-scale plots of an image

> **DOTV FALSE ; GO C_R** to make the plot file.

When **GREYS** has finished, run **LWPLA** to view the plot file. Note that **LWPLA** has a variety of options which control the plotting and scaling of the grey-scale images without having to rerun **GREYS**. In this example case, you should remember to set **FUNCTYPE** = ' ' and **DPARM** = 0 (or at least the first 4 values to 0) in **LWPLA** to avoid additional scaling. You may wish to color the labeling, contours, and background with **DOCOLOR**=1 and **PLCOLORS** with **LWPLA**. The procedure **TVCOLORS** will set **PLCOLORS** to match the TV graphics-plane colors; **DEFCOLOR** will set **PLCOLORS** to the standard TV colors even in the absence of a TV display. See examples on the color pages at the end of this chapter.

There are two other contour drawing tasks which offer additional options. **KNTR** is able to draw multiple contour, polarization, and/or grey-scale images in a single plot file, primarily to show multiple planes of a spectral-line cube; see § 8.5.4. **KNTR** uses a different, and probably superior, method of drawing the contour lines. It also can use color to represent different spectral channels and/or polarization angles or simply different colors for the different contour levels. **CCNTR** is virtually identical to **CNTR** except that it can draw extra symbols on the plot representing the locations and intensities of source model components found in CC (Clean component) or MF (model fit Gaussians from **SAD**, see § 7.5.3 and § 10.4.4).

6.3.2.2 Row tracing plots

There are a number of tasks which plot rows directly. Two of these are for use with single image planes while others are more intended for use with, *e.g.*, spectral-line data cubes transposed into velocity-ra-dec order. Of the former, **PLROW** is the simpler. It makes a plot file of all selected rows in an image plane. Each row is plotted as a slice offset a bit from the previous row. Low intensities which are “obscured” by foreground (*i.e.*, lower row number) bright features are blanked to keep the plot readable. Example inputs would be:

> TASK 'PLROW' ; INP C_R	to review the inputs.
> INDISK n ; GETN ctn C_R	to select the image on disk <i>n</i> catalog slot <i>ctn</i> .
> BLC 100 ; TRC 300 C_R	to select the sub-image from (100, 100) to (300, 300).
> YINC 3 C_R	to plot only every 3 rd row.
> PIXRANGE -0.001 0.050 C_R	to clip intensities outside the range –1 to 50 mJy.
> OFFSET 0.002 C_R	to set the intensity scaling such that 2 mJy separates rows of equal intensity.
> INP C_R	to check the inputs.
> GO C_R	to run PLROW .
> GO LWPLA C_R	to display the plot file on the laser printer after PLROW has finished.

The plot files produced by **PLROW** are a simple, special case of those produced by **PROFL**. This task makes a plot file of a “wire-mesh” representation of an image plane complete with user-controlled viewing angles and correct perspective. Enter **EXPLAIN PROFL C_R** for a full description. Both of these tasks are especially useful where the signal-to-noise ratio is high and examples of them are given in Figure 6.3.2.2.

In Chapter 8 we discuss the computation and use of “slices,” one-dimensional profiles interpolated along any line in an image plane. Once a slice has been computed, it may be plotted by **SL2PL** on the TV or into a device-independent plot file.

Three other row-plotting tasks, **PLCUB**, **ISPEC**, and **BLSUM**, are designed primarily for spectral-line and other data “cubes” (see § 8.5.4 and § 8.6). **PLCUB** makes one or more plot files showing the intensities in each selected row. The row subplots are positioned in a matrix in the coordinates of the 2nd and 3rd axes of the cube. **ISPEC** averages rectangular areas in each plane of a cube and plots the resulting spectrum. It can also save the output in a SLice file. **RSPEC** does the same except that it plots the robust rms in each plane rather

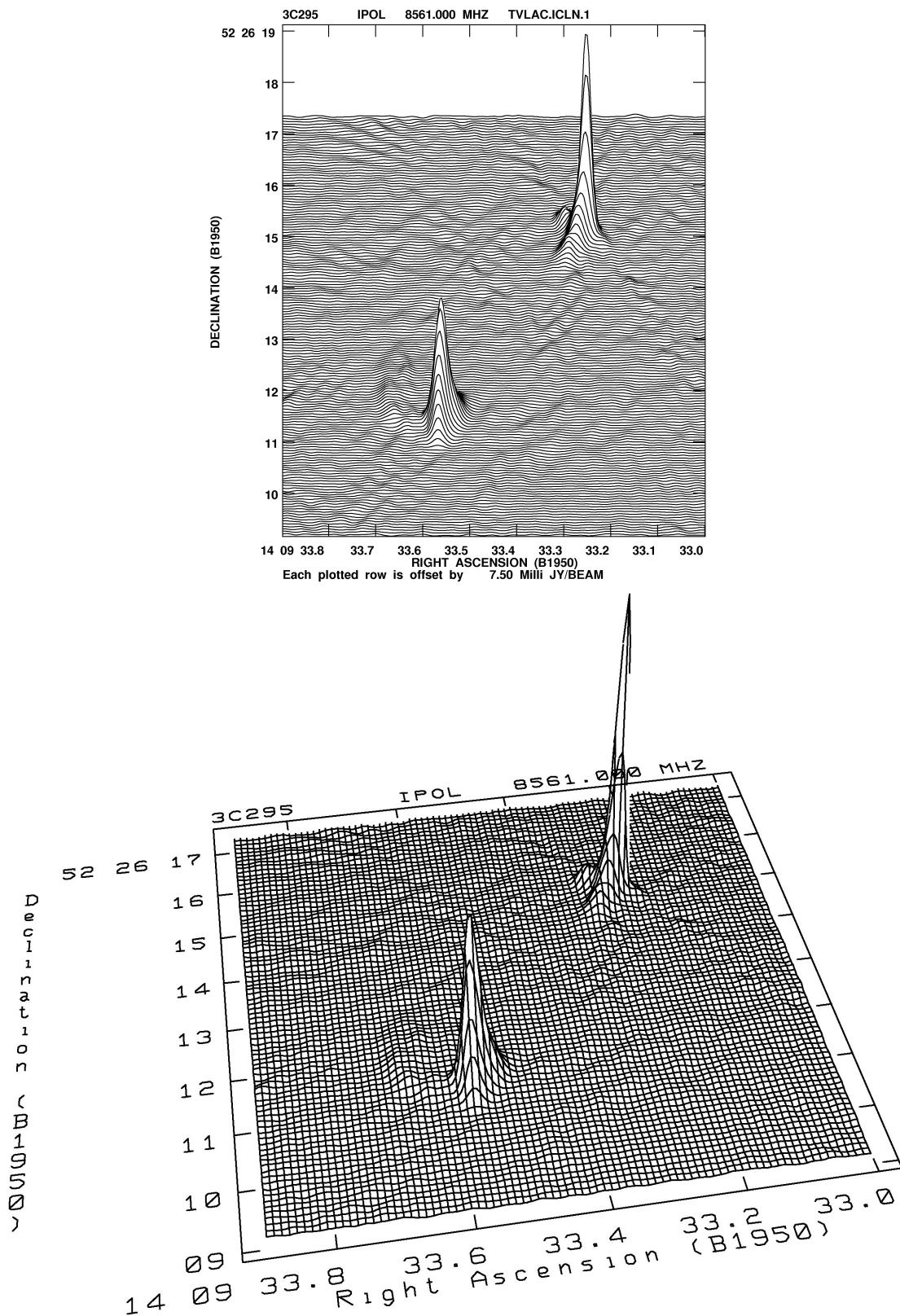


Figure 6.3: Row plots of an image with (bottom, [PROFL](#)) and without (top, [PLROW](#)) perspective

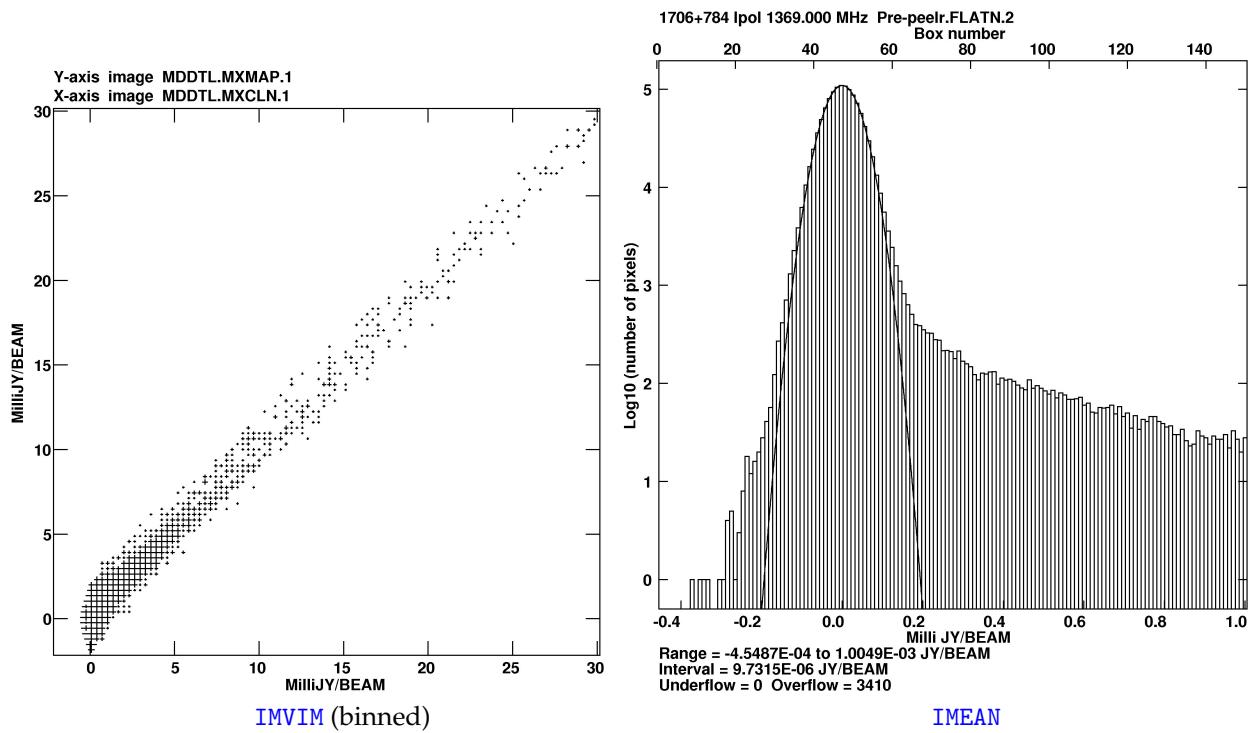


Figure 6.4: Plots of statistical parameters of an image.

than the data. **RSPEC** has options to write out a signal-to-noise image and/or a text file of channel weights as well. **BLSUM** can plot spectra from irregular regions selected on the TV with a “blotch” algorithm.

6.3.2.3 Miscellaneous image plots

IMVIM allows a variety of image comparisons by plotting the pixel values of one image against the pixel values of another image. The special options include binning the values (and plotting symbols proportional to the number of samples in a bin) and shifting one of the images in *x* and/or *y* with respect to the other. The former reduces large scatter diagrams to more manageable sets of numbers while the latter allows cross-correlation functions to be developed.

IMEAN prints the mean, rms, and extrema inside or OUTSIDE a user-specified window in an image. It also prints the intensity at, and rms of, the noise peak in the histogram and returns these values as adverbs to AIPS. Beginning with 31DEC12, it also returns an array in which **TRIANGLE**(*i*) is the brightness level above which there are *i* per cent of the image pixels. This is often useful in setting **PIXRANGE** for optical images. Optionally, **IMEAN** plots histograms of image intensities over the window using a user-specified number of summing cells over a user-specified range of intensities. Optionally, it also plots the fit to the noise peak on the user-specified histogram. An example of this is also shown in Figure 6.4.

6.3. Plotting your data

6. Displaying Your Data

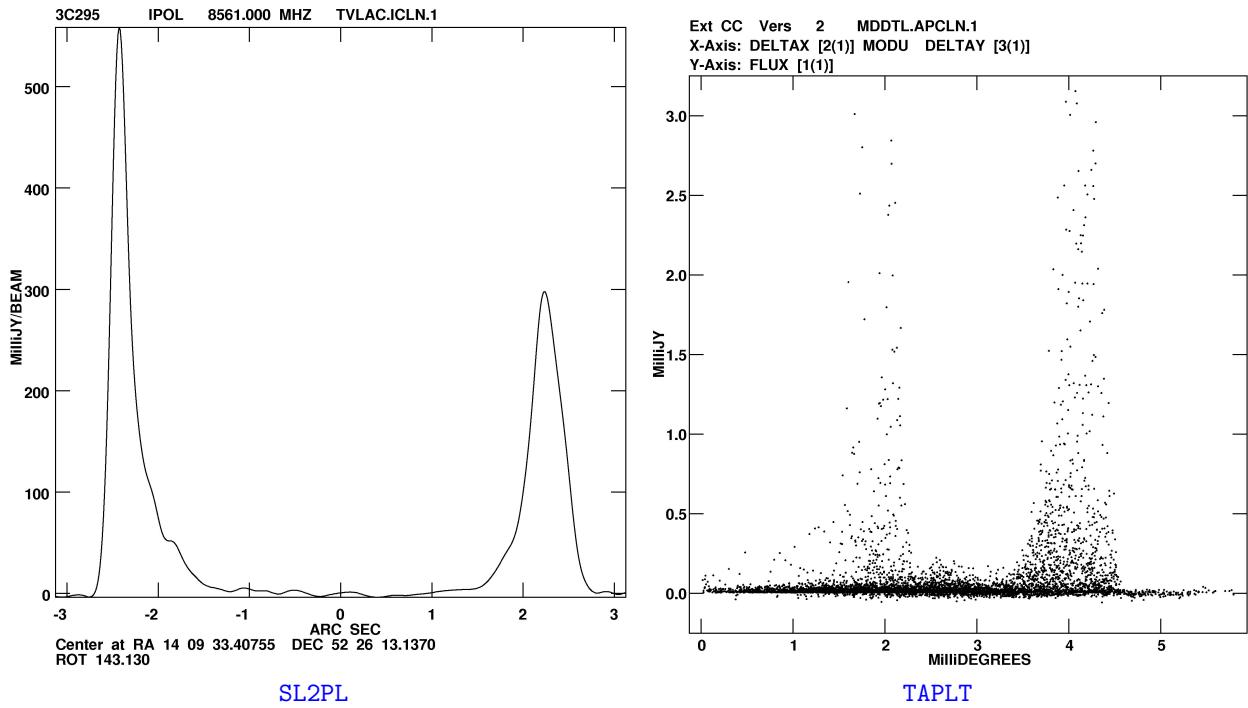


Figure 6.5: Slice and table plots.

6.3.3 Plotting your table data

TAPLT is a very general task to plot information from *AIPS* table extension files. It can plot a histogram of a function of the values in one or two columns of the table and it can plot a function of one or two columns against another function of another one or two columns. The latter can be summed or averaged in bins or have every point plotted. At first blush, the inputs seem rather complicated, but the results may well justify some effort to understand them. For example, to plot Clean component fluxes as a function of radius from the image center:

- > **TASK 'TAPLT' ; INP C_R** to review the inputs.
- > **INDISK n ; GETN ctn C_R** to select the image on disk *n* catalog slot *ctn*.
- > **INEXT 'CC' ; XINC 1 C_R** to select every row in the Clean components file.
- > **BCOUNT 1 ; ECOUNT 0 C_R** to select all rows in the table.
- > **APARM = 2 , 1 , 3 , 1 , 16 , 1 , 1 C_R** to plot the modulus of columns 2 and 3 on the *x* axis (*i.e.*, $\sqrt{\Delta_x^2 + \Delta_y^2}$) and column 1 on the *y* axis (*i.e.*, component flux).
- > **BPARM = 0 ; CPARM = 0 C_R** to use self-scaling of the plot and no scaling of the column values.
- > **DOTV TRUE ; GO** to plot the fluxes on the TV screen.

You may need to set the scaling with **BPARM** after seeing the preview plot in order, for example, to bring out the details in the low-level components.

There are a few tasks intended to make plotting specific kinds of tables rather easier. **SNPLT** plots calibration, solution, system temperature and **EVLA** SysPower tables for selected antennas as a function of time, antenna elevation, hour angle, azimuth, or sidereal time. It will make several plots per page (one antenna per plot) and multiple pages if needed, or you can plot the most discrepant value over all antennas in

a single plot. In one execution, polarizations and IFs may be plotted on separate plots or together on the same plots, optionally separated by color. In 31DEC12, **SNPLT** also has the option to separate sources by color instead. In 31DEC13, you can also plot multiple parameter types in the same execution. **SNIFS** is similar to **SNPLT** but is intended to compare solutions at different IFs, plotting the IF on the *x* axis with a variety of binning options.

POSSM will plot bandpass tables when **APARM(8)** is set to 2. Values 3 – 8 plot bandpass-like tables from **BLCHN** (BD) and **PCAL** (PD antenna polarization and CP source polarization). **POSSM** can also do multiple plots per page with all the usual data selection adverbs. **BPLOT** also plots BP and the other similar tables tables, with multiple times for one antenna or multiple antennae for one time on each plot. Combined with difference and coloring options, this allows close comparison of bandpasses as functions of time or antenna. **WETHR** plots the data in a WX weather table including parameters computed from those data such as relative humidity. **FGPLT** plots the times of selected flags from a flag table.

6.3.4 Plotting miscellaneous information

There are a few other tasks which create plot files, but which do not fit into the categories above. The most general of these is **PLOTR** which can plot up to ten sets of (*x,y*) points input from a text file with coloring options. **CONPL** is a task which plots *AIPS* convolving functions (used by various *uv*-data gridding tasks such as **IMAGR**) and their Fourier transform, expected signal-to-noise ratio, or convolution with a user-specified Gaussian. **IRING** integrates an image in concentric annuli about the user-specified object center with specified major axis position angle and inclination. The results may be placed in a plot file for later display. **GAL** calculates the orientation and rotation curve parameters of a galaxy from an image of the predominant velocities. The observed rotation curve is plotted together with the fitted model curve. **LOCIT** fits antenna location corrections to SN tables; the residual phases may be plotted as a function of sample number.

There are several tasks which create RGB cubes for later display by tasks such as **GREYS** and **KNTR**. These include **RGBMP** which does a weighted sum of the planes of a data cube, **HUINT** which uses two images as hue and intensity to construct an RGB cube, and **TVHUI** which interactively uses three images as intensity, hue, and saturation to construct an RGB cube by a different algorithm. Task **SCLIM** will scale and clip image planes to be used as inputs to **LAYER**, which produces an RGB cube from the colored sum of up to 10 input image planes using a complicated and general algorithm.

6.4 Interactive TV displays of your data

The *AIPS* TV display allows you to look at your image data in detail and to set parameters by pointing at interesting features visible on the screen. Although *AIPS* will run on a variety of hardware display devices (e.g., *I²S* Models 70, 75 and IVAS and DeAnza), it is now used almost exclusively with the X-Windows TV-simulation program called **XAS**. See § 2.3.2 for information on using this basic *AIPS* tool. Information on TV verbs and tasks may be found by entering **ABOUT TV C_R**, **ABOUT INTERACT C_R**, and **ABOUT TV-APPL C_R**, or by consulting the corresponding sections of Chapter 13 of this *CookBook*.

6.4.1 Loading an image to the TV

The simplest way to load an image from your catalog to the TV and then to manipulate the display is with the procedure called **TVALL**:

- > **INP TVALL C_R** to review the input parameters.
- > **INDI n ; GETN ctn C_R** to select the disk and image name parameters from the catalog.

Use one of the following commands to specify the initial transfer function that converts your image file intensities to display pixel intensities:

- > **FUNC 'LN' C_R** linear—this is the default.
- > **FUNC 'LG' C_R** logarithmic.
- > **FUNC 'L2' C_R** "extra" logarithmic.
- > **FUNC 'SQ' C_R** square-root, often a good compromise
- > **FUNC 'NE' C_R** negative linear.
- > **FUNC 'NG' C_R** inverse logarithmic.
- > **FUNC 'N2' C_R** inverse "extra" logarithmic.
- > **FUNC 'NQ' C_R** negative square root.
- > **PIXRA x1, x2 C_R** to load only image intensities x_1 to x_2 in the units of the image.
The default is to load the full range of intensities in the image.

(The slopes and intercepts of the display transfer functions can be modified later, but the above options let you choose initially between linear, square-root, logarithmic, and negative displays and restrict the range of intensities that is loaded). Then:

- > **TVALL C_R** to load the selected image.

The image should appear on the TV screen in black-and-white. If you see a new image but it is not what you expected, hit button D to end **TVALL** and then review your inputs with:

- > **INP TVALL C_R**

If no image appears, make sure that AIPS started up with your workstation assigned as the display; see § 2.2.3 for information on starting up AIPS and assigning the TV display.

After your image has been displayed on the TV by **TVALL**, your trackball with its buttons (which should be labeled A, B, C, and D) — or your workstation mouse with keyboard buttons A, B, C, and D) — can be used to modify the display transfer functions, coloring and zoom. Pressing button A alone enables black-and-white and color-contour coding of the image intensities, successively. Adjust the cursor position on the TV (using the trackball or mouse) to vary the slope and intercept of the display transfer function. **TVALL** will superpose a calibrated horizontal wedge on the image. This should help you to choose the optimum cursor setting for the display. Black-and-white displays are generally much more suitable than color for high-dynamic-range images, while color contouring may be used to accentuate interesting features. (Note also that a much wider range of image-coloring options is available outside **TVALL** by invoking **TVPSEUDO**, **TVPFLAME**, and **TVHELIX**.) Pressing buttons B and C adjusts the zoom of the display: B to increase the magnification and C to decrease it. When these buttons are enabled, the cursor controls the position of the center of the zoomed field of view. Magnification factors of 1 through 16 are available on most workstations. Note that your terminal issues instructions when buttons are pressed, but that it is in the death-like grip of **TVALL** otherwise until you press button D to exit from it.

The size of the image which can be displayed depends on the size (in pixels) of your workstation screen or TV display and some other parameters which can vary from site to site. A typical Sun workstation can display up to 1024 by 900 pixels. Then, if the image is larger than about 800 pixels or so in the y -direction, portions of the labeling of the wedge (the units) will be omitted or superposed on top of the wedge (the tick numeric values). A useful technique for displaying large images is to load only alternate pixels. The command:

> **TXINC** 2 ; **TYINC** 2 C_R to load every other *x* and *y* pixel.
 before **TVALL** would do this. Also use:
 > **TBLC** = *n*₁, *n*₂, *n*₃, ... C_R bottom left pixel to load.
 > **TRTC** = *m*₁, *m*₂, *n*₃, ... C_R top right pixel to load.
 to limit the displayed field. A small image may be interpolated to fill the TV screen by setting **TXINC** = -1 ; **TYINC** = -1 C_R. Recent versions of **XAS** allow the verb **TVROAM** to function again. This verb loads adjacent portions of an image to as many as 16 memories and then “roams” with a split screen to allow you to view any contiguous portion of the loaded image. **EROAM** resumes roaming an already loaded roam-mode image, perhaps after adjustment of the transfer function of all channels with **TVTRANSF** and **TVCHAN** = 0. The verb **ROAMOFF** allows you to convert the final split-screen image into a normal, more useful single-plane image.

TVALL is a procedure that insures that the desired graphics and TV channels are on and cleared and the others off, then loads the image with verb **TVLOD**, loads a wedge with **TVWEDGE**, and labels the wedge with **TVWLABEL**. You do not have to use **TVALL**, which can be rather slow, simply to load a new image on the TV. Use **TVLOD** instead. **TVWEDGE** has a variety of options concerning the width of the wedge and its position; type **HELP TVWED** C_R for details.

In 31DEC15, the task **TVHLD** was completely revised to load images to the TV with histogram equalization. The task has interactive controls over the intensity range and method of computing the histogram and can write out a histogram-equalized image in arbitrary, non-physical units for display by the various plotting tasks.

6.4.2 Manipulating the TV display

There are a number of verbs which allow you to manipulate the display, including:

- > **TVINIT** C_R to initialize the entire TV and TV image catalog. The image catalog is now kept in **XAS** and so is always current.
- > **TVON** *n* C_R to turn on TV grey channel(s) with *n* being a bit pattern of the desired channels: $2^{(i-1)}$. Thus **TVON** 4 turns on channel 3. You may have to turn off other channels to see the desired channel, since the sum of two images may be rather odd.
- > **TVOFF** *m* C_R to turn off channel *m*, where *m* is a bit pattern as with **TVON**.
- > **GRON** *n* C_R to turn on one or more of the 8 graphics channels, where *n* is a bit pattern.
- > **GROFF** *m* C_R to turn off one or more graphics channels.
- > **TVCHAN** *n* ; **TVCLEAR** C_R to zero a TV channel.
- > **GRCHAN** *m* ; **GRCLEAR** C_R to zero a graphics channel.
- > **TVZOOM** C_R to set the zoom magnification and center interactively, follow instructions on the screen.
- > **OFFZOOM** C_R to reset the zoom and zoom center to null.
- > **GREAD** C_R to read the current color of a specified graphics overlay channel into **RGBCOLOR**.
- > **GWRITE** C_R to change the color of a graphics overlay channel to that specified by **RGBCOLOR**. This may be done for aesthetic reasons or because the default colors may not show up well when captured by **TVCP** and printed on a color printer.
- > **TVPOS** C_R to read the TV cursor position, returning adverbs for use in procedures or other verbs.

> **IM2TV** C_R to convert an image pixel in **PIXXY** to the corresponding TV pixel.

6.4.3 Intensity and color transfer functions

The *AIPS* model of a TV postulates two intensity transfer functions, called the LUT and the OFM, which are basically multiplicative. In most circumstances, the LUT is used for black-and-white enhancements and the OFM for coloring, but both can be used for either. To manipulate the LUT interactively, while leaving the pseudo-coloring alone, use the **TVTRAN** verb. The cursor position controls the slope and intercept of the transfer function, buttons A and B switch a plot of the transfer function on and off, button C switches the sign of the slope, and button D (as always) exits. To turn the LUT back to normal, enter **OFFTRAN**.

A rich zoo of color coding is available with **TVPSEUDO**, which alters the OFM while leaving the LUT alone. Repeated hits on button A select a variety of color triangles, button B selects a circle on hue, and repeated hits on button C select a variety of color contours. First-time users should experiment with the *AIPS* coloring options until they develop an intuitive feel for the effects of cursor settings on the image appearance. The wedge displayed by **TVALL** adjusts to the alternative colorations selected with **TVPSEUDO**, and it is helpful to watch changes in both the wedge and the image. A flame-like coloring is available with **TVPFLAME**, or variations on the scheme with repeated hits on buttons A or B. **TVHELIX** does a helix in color space attempting to make a monotonic increase in perceived intensity. The buttons change direction, number of rotations, and saturation. In these verbs the cursor position controls aspects of the coloring such as enhancements, richness, or cycles of hue. To turn off pseudo-coloring, enter **OFFPSEUD**.

A set of less well-known verbs is available to allow you to create, manipulate, and save desirable versions of the OFM table. **OFMSAVE** allows you to save a named OFM, **OFMDIR** lists all saved OFMs belonging to you or generally available from the *AIPS* distribution, **OFMGET** loads a named OFM, **OFMZAP** deletes a named OFM, and **OFMLIST** prints the current OFM. **OFMCONT** is an elaborate interactive verb which allows you to set the hue, intensity, and saturation of the OFM divided up into a number of color contours. Each of these contours can be a constant level or a step wedge. **OFMADJUS** is another elaborate interactive verb to alter pieces of the OFM, while **OFMTWEAK** is a simpler verb to stretch the OFM.

6.4.4 Setting parameters with the TV

One reason to load the image to the TV is to set adverbs for use by other verbs and tasks. Verbs which use the TV cursor to set adverbs include:

> **TVNAME** C_R to set **INDISK**, **INNAME**, etc. to the name parameters of the image currently visible. If there is an ambiguity, you will be asked to move the cursor to the desired image and press a button.

> **TVWIN** C_R reads pixel coordinates from the next two cursor positions at which a trackball button is depressed. The TV graphics shows the current shape and position of the window. Button A allows you to switch to (re)setting the other corner while the other buttons exit after both corners have been set. **TVWIN** uses the pixel coordinates to set up the bottom left (**BLC**) and top right (**TRC**) corners of an image subsection, *e.g.*, for input to the contouring programs **CNTR** and **PCNTR**, to the mean/rms calculator **IMEAN**, and to many other tasks.

> **SETXWIN** (*dx,dy*) C_R reads pixel coordinate of the center of a *dx*-pixel by *dy*-pixel window and sets the adverbs **BLC** and **TRC**.

- > **SETSLICE** C_R works like **TVWIN** above to set **BLC** and **TRC**. Instead of a rectangle however, the display shows a diagonal line which is useful for setting the ends of slices.
- > **TVBOX** C_R is similar to **TVWIN** above except that it is used to set up pixel coordinates to define rectangular or circular Cleaning areas for the *AIPS* Clean tasks. The adverbs **NBOXES** and **CLBOX** are set. The circular option appeared in the 15JUL95 release and is not supported by **APCLN** and **MX**.
- > **REBOX** C_R allows revision using the TV of the Cleaning areas set previously with **TVBOX**. Revises **NBOXES** too.
- > **DELBOX** C_R allows deletion using the TV of the Cleaning areas set previously with **TVBOX**. Revises **NBOXES** too.
- > **FILEBOX** C_R is **REBOX** for boxes in the text file used with **IMAGR**; multiple fields and many more boxes are allowed.
- > **DFILEBOX** C_R is **DELBOX** for boxes in the text file used with **IMAGR**; multiple fields and many more boxes are allowed.
- > **MFITSET** C_R allows you to set the adverbs for **IMFIT** and **JMFIT**, namely the image name parameters, window, and initial guesses for the Gaussians from the image on the TV.

Task **FILIT** performs **FILEBOX** and much more on a set of images, usually the multiple facets of a wide-field imaging. It offers a wide range of image display options including handling images larger than the TV, allows editing of the box information, creates boxes via the auto-boxing algorithm, and checks the multiple facets for overlapping boxes. This new task may be the quickest and easiest way to review a set of facet images.

6.4.5 Reading image values from the TV

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

- > **IMPOS** C_R displays the two coordinate values (e.g., RA and Dec) from the cursor position when any button is depressed. Adverbs **TVBUT** and **COORDINA** are returned.
- > **IMXY ; IMVAL** C_R displays the image intensity and the two coordinate values (e.g., RA and Dec) from the cursor position when any button is depressed. Adverbs **PIXXY**, **TVBUT**, **PIXVAL**, and **COORDINA** are set.
- > **CURVALUE** C_R continuously displays (in the upper-left corner of the TV) the pixel coordinates and the image intensity in user-recognizable units at the position selected by the TV cursor.
- > **TVFLUX** C_R displays image intensities and coordinates whenever a TV button is pressed, looping until button D is pressed. Adverbs for the first image name are set as well as **PIXXY**, **TVBUT**, **PIXVAL**, and **COORDINA** for the last pixel selected.
- > **TVDIST** C_R displays the angular length and position angle of the spherical vector between two pixels in one or two images shown on the TV. Name adverbs for input files 1 and 2 are set as well as adverbs **PIXXY**, **PIX2XY** and **DIST**.

- > **TVMAXFIT** C_R whenever a TV button is pressed, fits a quadratic function to the image to find the position and strength of an extremum, looping until button D is pressed. Adverbs for the first image name are set as well as **PIXXY**, **TVBUT**, **PIXVAL**, and **COORDINA** for the last object selected.
- > **COPIXEL** C_R to convert between pixel and astronomical coordinates for an image; allows determination that a coordinate is not inside an image.
- > **TVSTAT** C_R determines the mean, rms, extrema and integrated intensity (if appropriate) in user-defined “blotch” regions within the image currently displayed on the TV. The regions are irregular polygons selected with the TV cursor. Type **EXPLAIN TVSTAT** C_R for details. Adverbs **PIXAVG**, **PIXSTD**, **PIXVAL**, **PIXXY**, **PIX2VAL**, and **PIX2XY** are set.

6.4.6 Labeling images on the TV

There are a number of facilities for labeling images on the TV including:

- > **CHARMULT** /CR to scale the character size used by the TV by 1 through 5 times the standard size which is 7 by 9 pixels.
- > **TVLABEL** C_R to draw standard axis labels around the visible image. You may control the type of labeling and whether coordinates are shown as a grid or short tick marks. If more than one image is visible, you will be asked to indicate which one you want with the cursor and any button.
- > **TVWLABEL** C_R to draw axis labels around the visible intensity wedge.
- > **TVANOT** C_R to draw a text string into a grey-scale channel or a graphics plane at a location specified via an adverb or via the TV cursor.
- > **TVLINE** C_R to draw a straight line into a grey-scale channel or a graphics plane at locations specified in part or in whole via adverbs or via the TV cursor.
- > **TVILINE** C_R to draw a straight line into a grey-scale channel or a graphics plane between two image pixel coordinates.
- > **COSTAR** C_R to plot a “star” positions at a user-specified coordinate on the TV image.
- > **TVSTAR** C_R to plot “star” positions from an ST file on top of the visible image; see § 6.3.2.

6.4.7 Comparing images on the TV

It is often useful to compare two images, e.g., to decide whether one contains artifacts that are not present in another at the same frequency, or to look for frequency-dependent features at constant resolution. AIPS provides several tools for such image comparisons.

The first tool is a capability for loading multiple images to the same plane (or channel) of the TV device. The parameter **TVCORN** specifies where the bottom left corner of the image or image subsection will be positioned in the TV frame by **TVLOD** or **TVALL**. If **TVCORN** is left at zero, **TVLOD** and **TVALL** adjust it to center the displayed image. You may however use **TVCORN** to control loading successive images to different regions

of the display with successive executions of **TVLOD**. For example, the following commands would load two 512 by 512 pixel images from slots 1 and 2 on disk 1 *side-by-side* on channel 1 of a 1024 by 900 TV display:

- | | |
|--|---|
| > INDI 1; GETN 1 C_R | to select the input disk and the first image. |
| > TVCH 1 C_R | to select TV channel 1 for the loading. |
| > TVCORN 1 193 ; TVLOD C_R | to load the first image. |
| > GETN 2 C_R | to select the second image. |
| > TVCORN 513 193; TVLOD C_R | to load the second beside the first. |

You could then adjust the color coding, transfer function, etc. for both images simultaneously with **TVFIDDLE** or **TVTRAN**. You may load as many as 256 images to a single TV plane with this technique, which is therefore a powerful method for making “montages.” The number of simultaneous images is limited mostly by your image sizes, the need to avoid overlaps (which are allowed if you want) — and your ability to do the arithmetic for appropriate **TVCORN** settings! You are also limited by the need for all the images in one plane to share that plane’s transfer function. Judicious use of the **PIXRANGE** and **FUNC** inputs to **TVLOD** permits making useful montages of disparate images, however.

A second tool is the classic “blink” technique from optical astronomy. **TVBLINK** allows you to load images to two different planes of the TV memory and then to alternate the display rapidly between the two. The two images described above could be “blinking” against each other by the following command sequence:

- | | |
|--|--|
| > INDI 1; GETN 1 C_R | to select the input disk and first image. |
| > TVINIT ; TVCORN 0 | to clear the TV and restore the default positioning. |
| > TVCH 1; TVLOD C_R | to load the first image on plane 1. |
| > GETN 2 C_R | to select the second image. |
| > TVCH 2 ; TVLOD C_R | to load the second image on plane 2. |
| > TVCH 12 ; TVBLINK C_R | to blink planes 1 and 2. |

The rate and duty cycle of the blinking, and the transfer functions applied to the planes, are controlled interactively with the TV cursor. Instructions for these operations appear on your terminal while **TVBLINK** is running.

The task **PLAYR** provides a menu-driven method to enhance and blink two images and to develop and save TV color tables (“OFMs”).

For data cubes (*e.g.*, frequency or time sequences of images), the verbs **TVMOVIE** and **TVCUBE** combine the two previous techniques. These are described in more detail in §8.5.4. Both verbs load one or more image planes with as many planes from the cube as possible (and as requested). Then they display each frame in sequence with interactive controls over the frame rate in movie mode, the chosen frame in single-frame mode, and the brightness, contrast, and color of the displayed images. **TVMOVIE** makes a somewhat more efficient movie sequence, but **TVCUBE** makes a better montage by using a more normal arrangement of the image planes. The **DOALL** adverb allows these verbs to loop over more than just the third axis if the “cube” has more than three multi-pixel dimensions.

Certain real TV displays used to provide powerful tools to compare images using color as well as intensity to represent real information. **XAS** implements some of these tools on workstations capable of displaying a virtually unlimited number of colors. Perhaps the most powerful of these is the “hue-intensity” display in which one image controls the intensity of the displayed image and the other controls the hue. A number of uses for this algorithm are obvious, including spectral-line moment images (intensity from total line integral and velocity setting color), polarization images (intensity from the total or polarized intensity image and polarization angle setting color in a circular scheme), and depolarization observations (color set by a two-frequency depolarization image). The best implementation of this algorithm has been in the interactive verb **TVHUEINT**:

- | | |
|--|--|
| > INDI <i>d1</i> ; GETN <i>ctn1</i> C_R | to select the intensity image. |
| > TVINIT ; TVCORN 0 | to clear the TV and restore the default positioning. |

- | | |
|--|--|
| > TVCH 1; TVLOD C_R | to load the first image on plane 1. |
| > INDI d2; GETN ctn2 C_R | to select the hue image. |
| > TVCH 2 ; TVLOD C_R | to load the second image on plane 2. |
| > TVCH 1 ; TV2CH 2 ; TVHUEINT C_R | to display a full-color view where the intensity is controlled by image 1 and the hue by image 2 and to interactively adjust that display. |

Instructions for altering transfer functions and reversing the roles of the two images appear on your terminal while **TVHUEINT** is running. You may create a labeled two-direction step wedge to accompany the hue-intensity display with the verb **HUEWEDGE**.

Until the 31DEC14 release, the only way to capture this image was with **TVCPS**, which is fundamentally limited by the size and resolution of the TV display. Control over labeling, character font, number of pixels, and the like is severely limited. To overcome these limitations, a new task called **HUINT** was written. It performs the functions of **TVHUEINT** with the option of saving the resulting image and a two-dimensional step wedge as *AIPS* catalogued images. The task enters an interactive mode in which the user selects options from a menu shown on a graphics overlay plane. Options are to enhance each of the images individually, to select linear, square-root, or one of 2 logarithmic transfer functions for intensity or hue, and to exit with or without final updates and an output file. The output image(s) can then be made into plot files with standard plot tasks like **GREYS** and **KNTR** and then converted into PostScript by **LWPLA**. **HELP POSTSCRIPT** contains some guidance on how to combine PostScript renderings of the color image and the color step wedge into a single file. Note that the images saved by **HUINT** are not in any recognizable physical units. Also note that **KNTR** should be run with **DOCOLOR=1; DOWEDGE=0 FUNCTYPE=' '** and **LWPLA** should be run with **RGBGAMMA=1** and no extra enhancements.

The two functions above require a display that can do an almost unlimited number of colors. Unfortunately, some older workstations can display only 256 simultaneous colors (or even fewer). **XAS** supports both kinds of workstation. For these limited workstations, there are two tasks, also discussed in § 8.6, which attempt to recover much of this capability by trying to optimize color assignments over the limited range available. The first of these, **TVHUI** produces a composite display in which the intensity is set by one image, the hue is derived from another image, and the saturation is optionally derived from a third image. An interactive menu allows you to enhance each of the images individually, to select linear, square-root, or logarithmic transfer functions for the intensity and hue images, to select the sub-image used during interactive enhancements, to repaint the full image, and to exit with or without writing out the final three-color image. The task has the option of writing an image of a hue-intensity step-wedge as well. A number of uses for the saturation portion of this task are obvious, including spectral-line moment images (line width setting saturation) and polarization images (polarization intensity setting saturation). **TVHUI** is also useful on full-color displays, but for such displays, you may do much of the work of this task in **HUINT**. **TVHUI** attempts to write out the full range of input intensities scaled by the selected transfer functions. Thus, its output images are different from those of **HUINT** and may, or may not, be preferable in particular cases.

There is also an *AIPS* task called **RGBMP** which writes three-color cubes using weighted sums over a data cube. Three-color cubes also arise when digitizing color photographs of real scenes. The second task, **TVRGB**, can be used to display these three-color cubes or to generate a three-color display from any three *AIPS* image planes. Common examples of the latter are the superposition of radio continuum and/or line data on optical or X-ray images, and color-coding of effective temperatures or spectral indices from 3-channel continuum data. **TVRGB** can also be used to color-code different types of depolarization effects from multi-frequency polarimetry. Like **TVHUI**, **TVRGB** offers a simple menu to enhance each of the images individually or all together, to select the window specifying the sub-image which is used during interactive enhancements, to repaint the full image on the TV, and to exit. **TVRGB** does not write an output image per se, but it can be instructed to write out a full 24-bit color PostScript plot file to be sent to a color printer. Its display (or any other TV display including that of **TVHUI**) can be captured and sent to a color printer; see § 6.4.10 below.

On full-color workstations, three-color images may be displayed by loading each color plane to a separate TV memory. Then each memory is turned on in the desired color only using **TVON** with the usually ignored **COLORS** adverb. If the red image is in TV channel 1, the green in 2 and the blue in 3, the verb **TV3COLOR** is a short-cut for all the parameter setting.

> **FOR TVCH=1:3; TBLC(3)=TVCH; TVLOD; END; TV3COLOR C_R**

6.4.8 Slice files and the TV display

In Chapter 8 we discuss the computation and use of “slices,” one-dimensional profiles interpolated along any line in an image plane. Once a slice has been computed, it may be plotted by **TVSLICE** on the TV display in your choice of graphics channel. A second slice may be plotted on top of the first with **TVASLICE**. The TV graphics display is used to prepare initial guesses for **SLFIT**, which fits Gaussians to slices. The verbs involved are:

> NGAUS n ; TVSET C_R	to set the number of Gaussians to be fitted to <i>n</i> and then to prepare an initial guess at the parameters by pointing at the peaks and half width points on a graphics plot of the slice.
> TV1SET j C_R	to revise the initial guess for the <i>j</i> th Gaussian.
> TVGUESS C_R	to plot the initial guess of the model on the TV graphics, erasing any previous plot.
> TVAGUESS C_R	to add a plot of the initial guess of the model to the current slice plot on the TV graphics.
> TVMODEL C_R	to plot the fit model on the graphics device, erasing any previous plot.
> TVAMODEL C_R	to add a plot of the fit model to the current slice plot on the TV graphics.
> TVRESID C_R	to plot the data minus the fit model on the TV graphics, erasing any previous plot.
> TVARESID C_R	to add a plot of the residuals (data minus model) to the current slice plot on the TV graphics.
> TVCOMPS C_R	to plot the individual components including baseline of the fit model on the TV graphics, erasing any previous plot.
> TVACOMPS C_R	to add a plot of the individual components including baseline of the fit model to the current slice plot on the TV graphics.

The units for slice model parameters are those of the plot, so it is convenient to set them with these verbs. These same operations may also be done on the TEK graphics device (§ 6.5.2), but modern X-Windows emulations of such devices seem to have problems with cursor reading. The TV verbs allow multiple colors for plot comparisons using different **GRCHAN**s.

6.4.9 Other functions using the TV

There are a number of tasks which use the TV to give the user real interactive input to the operation based on the images displayed by the task on the TV. These include **BLANK** (§ 8.6) to blank out non-signal portions of an image, **BLSUM** (§ 8.6) to sum images over irregular blotch regions plotting (in 31DEC14) or printing out summed spectra (and saving SLice files), **TVFLG** (§ 4.4.3) to edit visibility data based on grey-scale displays of some function of the visibility with baseline on the *x* axis and time on the *y* axis, **SPFLG** (§ 8.1, § 10.2.2) to edit visibility data based on grey-scale displays of some function of the visibility with spectral channel for all IFs on the *x* axis and time on the *y* axis, **EDITR** (§ 5.5.2) to edit visibility data based on plots of

visibility versus time, 1–11 baselines at a time, **EDITA** (§ 4.4.2) to edit visibility data based on plots of system temperature (TY or SY tables) or antenna gains (SN or CL tables), **WIPER** to edit visibility data using **UVPLT**-like displays, and **SNEDT** to edit TY, SY, SN, and CL tables themselves. In 31DEC16 new TV-based tasks appeared to edit visibility data displayed on a $u - v$ grid (**UFLAG**), to edit visibility data based on graphical displays of bandpass solutions (**BPEDT**), and two tasks to edit pulse-cal tables (**PCFLG**, **PCEDT**).

The imaging tasks, **IMAGR**, **SCIMG**, and **SCMAP** display the results of the computation at its current stage on the TV and provide a menu of interactive options to the user. The menu includes the usual display enhancements, the ability to choose among the images being computed, the ability to set Clean windows in those images, and the ability to end the computation at its current stage. The computation will resume when instructed via the menu or after a period of inactivity. A number of older iterative tasks use the TV to display the results of the computation so far and then prompt the user to hit button D within some number of seconds to stop the computation. Tasks that do this include **APCLN**, **MX**, **SDCLN**, **VTESS**, and several less significant tasks. **UVMAP** uses the TV simply to draw a picture indicating which cells are sampled in the uv plane. All of these tasks are described in Chapter 5.

In 31DEC14, the multi-source fitting task **SAD** was provided with a companion task **TVSAD** which allows interactive setting of the initial guess parameters for each island being fit. It allows re-trying the fits and adjusting the island boundaries, and can be run non-interactively until the fit finds a problem, after which the interactive mode resumes. *AIPS Memo 119*³ describes the operation of this task in detail.

There is a set of related tasks for analysis of data cubes transposed so that the first axis is the one on which baselines or Gaussians are to be fit. In the case of a spectral-line cube, the image would be transposed so that velocity is the first axis. In 31DEC16, **TVSPC** is an interactive task which displays spectra from one or two transposed cubes at pixels chosen on the TV from some image of the overall field. A third cube may also be displayed with the spectral plane chosen by the cursor on the displayed spectra. See the associated *AIPS Memo*⁴. In earlier releases, use **XPLOT** (with **DOTV** 1) first to get an idea of what the profiles really look like. It uses a flux cutoff to determine which profiles to display and prompts you for permission to continue after each plot. **XBASL** is used to remove n^{th} -order polynomial baselines from each spectrum. It has a batch mode of operation and an interactive mode which uses the graphics display to plot each spectrum and to accept guidance on which channels to use in determining the baselines.

XGAUS fits up to 8 Gaussians plus a linear baseline to each spectral profile. In 31DEC13, it builds a table of solutions at each pixel and may be re-started until you are satisfied, after which images of the fit Gaussian parameters and their uncertainties may be written to disk. In its interactive mode, it plots each selected spectrum on the TV graphics planes and accepts guidance on the initial guesses for the Gaussians. After all pixels have been done, it displays images of the fit parameters and offers a variety of options for editing and correcting the results. In 31DEC13, tasks similar to **XGAUS** have been written to fit polarization spectra (**RMFIT**) and to fit for Zeeman splitting (**ZEMAN**) of the line viewed simply or as a set of Gaussians found by **XGAUS**. Plot tasks **XG2PL** and **RM2PL** have been written to display these spectral fits in standard *AIPS* plot files. See *AIPS Memo 118*.⁵ In 31DEC17, corresponding tasks **AGAUS** and **ZAMAN** became available to perform similar analysis of absorption-line image cubes. See *AIPS Memo 122*.⁶

6.4.10 Capturing the TV

Having done all the work to prepare the absolutely perfect display on your TV screen, it would be a good idea to capture it before someone, such as the local power company, does a **TVINIT**. See § Z.1.2.2 for a discussion of Unix tools to do this. We recommend, however, **TVCPS** to capture the image on your TV,

³Greisen, E. W. 2014, “**TVSAD**: interactive search and destroy” AIPS Memo 119, <http://www.aips.nrao.edu/aipsdoc.html>

⁴Greisen, E. W. 2017, “Exploring Image Cubes in *AIPS*,” AIPS Memo 120 revised, <http://www.aips.nrao.edu/aipsdoc.html>

⁵Greisen, E. W. 2017, “Modeling Spectral Cubes in *AIPS*,” AIPS Memo 118 revised, <http://www.aips.nrao.edu/aipsdoc.html>

⁶Greisen, E. W. 2017, “Modeling Absorption-line Cubes in *AIPS*,” AIPS Memo 122, <http://www.aips.nrao.edu/aipsdoc.html>

including graphics overlay channels if they are on, and to write the result to an encapsulated PostScript file. This file can be printed immediately on a black-and-white or color printer or on any other device which understands PostScript. It can also be saved for later printing or inclusion in other documents. **TVCPS** was used to make the picture on the title page of this *CookBook*, the picture of **TVFLG**'s display in Chapter 4, and the picture of a right ascension - velocity - declination data cube in Chapter 9. **TVCPS** bases its picture on the current size of your TV display. If you are using a workstation with **XAS**, be sure to adjust the size of the display window to encompass all of your image plus a modest border. If you leave a large border, you will get a large border in your output. **TVCPS** understands both pseudo- and full-color **XAS** displays. Your **TVCPS** session could look like:

> TASK 'TVCPS' ; INP C _R	to review the inputs.
> OUTFILE 'MYAREA:TV.PIX' C _R	to save the output in a file called TV.PIX in an area defined by the logical MYAREA; see § 3.10.1.
> OPCODE 'COL' C _R	to make a color picture.
> APARM 8.5 , 11 C _R	to set the output device size to 8.5 by 11 inches, appropriate to standard quarto paper.
> GO C _R	to run the task when the inputs are set.

TVCPS has an option to add a character string below the image with adverb **REASON**. If your image is too large to fit on the TV, you can instruct **TVCPS** to read the image from disk with **DOTV** = -2 C_R, using adverbs **TBLC**, **TTRC**, **TXINC**, and **TYINC**. When doing this, you should turn off the graphics display (**GROFF** 32767 C_R) since it is not aligned properly. This option works when multiple images are visible at the same time, but only when they are in separate TV planes as fully overlapped 3-color images. Some color printers and recorders have rather different transfer characteristics than the workstation screen. **TVCPS** offers the option to remove the “gamma correction” used for your workstation and to apply a different one appropriate to your color recorder. **TVCPS** now offers the option to represent colors using the CMYK (cyan-magenta-yellow-black) used in printing rather than the familiar RGB colors system. CMYK displays often require different gamma corrections from those used for RGB. To see the effects of the gamma correction, try the verb **GAMMASET** in AIPS. You may need to use **GWRITE** to select better colors for the graphics overlay planes as well.

§ Z.1.2.2 contains a discussion of Unix tools to capture the TV screen which work even when the *AIPS* display task has control of the display. The **import** program is perhaps the more useful of these.

6.5 Graphics displays of your data

In the dim dark past, Tektronix invented some nice graphical display devices and an inconvenient but functional way to talk to them. This communication language became so embedded in software that workstation vendors now provide X-Windows windows that understand it. *AIPS* also arose from this dim dark past and once upon a time talked to those lovely green screens. To retain the graphics capability, we now provide a **TEKSRV** server which will provide a Tektronix-like graphics screen on which certain *AIPS* tasks and verbs plot. When AIPS starts up on workstations, it brings up a window called **TEKSRV**. Leave this window in its iconic state; it is only a marker for the presence of the server. The first time you write to **TEKSRV**, it will create and open a window called **TEKSRV** (Tek) in which the plot is done. You can resize this window within some limits and the plot will automatically resize itself. When the workstation cursor is in the Tek window, it changes to a diagonal arrow pointer. When an *AIPS* task or verb tries to read from the Tek window, this pointer becomes a plus sign. You should position the pointer to the desired location *without* touching the keyboard or the mouse buttons. When the pointer is exactly where you want it, press any mouse button or any key (*except* **RETURN**) to return the pointer position to the program. Note that the functions using the graphics display are not quite as friendly as those on the TV. This is due to the inability to erase a piece of a plot without erasing all of it. You can erase the full screen with **TKERASE**, which will keep the window from redrawing a big plot on every expose event. In recent years, we have found

unfortunately that cursor reading from Tektronix emulation screens can be unreliable. Therefore, every TK function described below also has a TV version, described in a previous subsection. The TV has many other advantages over the Tektronix emulator and now is nearly as fast.

6.5.1 Plotting data and setting values with the graphics display

[TKPL](#) interprets AIPS plot files to the graphics window or device. The graphics screen can be used to read back data values and set adverbs, much like the TV:

- > [TKXY](#) C_R to read the graphics cursor position, setting adverb [PIXXY](#); requires a contour or comparable image to be shown on the screen.
- > [TKXY](#) ; [IMVAL](#) C_R to read the graphics cursor position and return the image value and coordinates of the selected position.
- > [TKPOS](#) C_R to read the graphics cursor position and return the image coordinates of the selected position.
- > [TKWIN](#) C_R to read the graphics cursor position twice, first setting [BLC](#) and then setting [TRC](#).
- > [TKBOX](#)(*i*) C_R to read the graphics cursor position twice, first setting the lower left and then the upper right of Clean box *i*.
- > [TKNBOXS](#)(*n*) C_R to set [NBOXES](#) to *n* and then set all *n* Clean boxes using the graphics cursor.

All of these verbs require a graphics display of a plot file produced by [CNTR](#), [PCNTR](#), [GREYS](#), or [SL2PL](#). The window procedures don't make much sense with a slice plot, but they will work.

6.5.2 Slice files and the graphics display

In Chapter 8 we discuss the computation and use of "slices," one-dimensional profiles interpolated along any line in an image plane. Once a slice has been computed, it may be plotted by [TKSLICE](#) on the graphics display. A second slice may be plotted on top of the first with [TKASLICE](#). The graphics display is used to prepare initial guesses for [SLFIT](#), which fits Gaussians to slices. The verbs involved are:

- > [TKVAL](#) C_R to return the flux level pointed at by the graphics cursor.
- > [NGAUS](#) *n* ; [TKSET](#) C_R to set the number of Gaussians to be fitted to *n* and then to prepare an initial guess at the parameters by pointing at the peaks and half width points on a graphics plot of the slice.
- > [TK1SET](#) *j* C_R to revise the initial guess for the *j*th Gaussian.
- > [TKGUESS](#) C_R to plot the initial guess of the model on the graphics device, erasing any previous plot.
- > [TKAGUESS](#) C_R to add a plot of the initial guess of the model to the current slice plot on the graphics device.
- > [TKMODEL](#) C_R to plot the fit model on the graphics device, erasing any previous plot.
- > [TKAMODEL](#) C_R to add a plot of the fit model to the current slice plot on the graphics device.
- > [TKRESID](#) C_R to plot the data minus the fit model on the graphics device, erasing any previous plot.
- > [TKARESID](#) C_R to add a plot of the residuals (data minus model) to the current slice plot on the graphics device.

The units for the slice model parameters are fairly problematic, so we recommend using these graphical input and output functions. At least, they all have the same strange ideas. See § 6.4.8 for the verbs that allow this same processing using the TV display.

6.5.3 Data analysis with the graphics display

There is a set of related tasks for analysis of data cubes transposed so that the first axis is the one on which baselines or Gaussians are to be fit. In the case of a spectral-line cube, the image would be transposed so that velocity is the first axis. It is a good idea to use `XPLT` first to get an idea of what the profiles really look like. It uses a flux cutoff to determine which profiles to display and prompts you for permission to continue after each plot. `XBASL` is used to remove n^{th} -order polynomial baselines from each spectrum. It has a batch mode of operation and an interactive mode which uses the graphics display to plot each spectrum and to accept guidance on which channels to use in determining the baselines. `XBASL` can be asked to write images of the baseline parameters. Unfortunately, this task requires you to do the full cube in a single execution, which is rather an endurance contest.

6.6 Additional recipes

6.6.1 Banana mallow pie

1. Combine 2 cups **vanilla wafer** crumbs and 1/3 cup melted **butter**. Press into 9-inch pie plate and bake at 375° F for 8 minutes.
2. Prepare a 3 1/8 ounce package **vanilla pie filling** using 1 3/4 cup **milk**. Cover surface with transparent wrap and chill.
3. Fold 1 1/2 cups **mini-marshmallows** and 1 cup **Cool Whip** into pie filling.
4. Slice 2 **bananas** into pie crust, pour filling over bananas, and chill several hours or overnight.

6.6.2 Sopa de Plátano

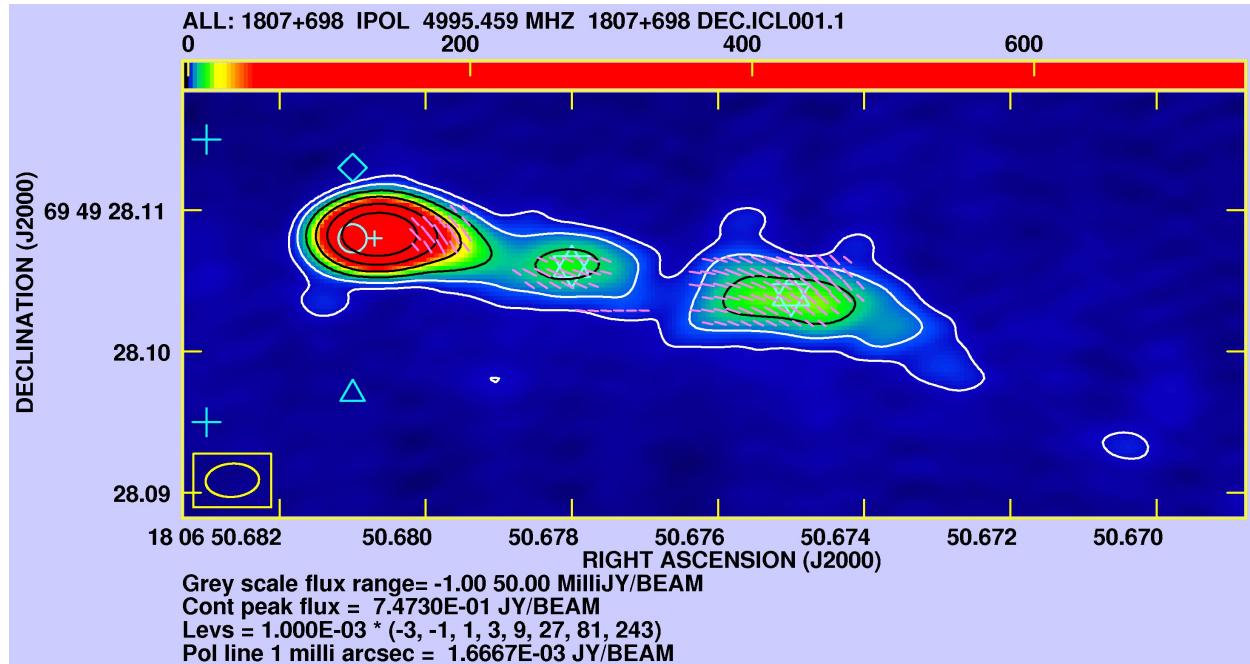
1. Cook 10 whole red-skinned under-ripe **bananas** in one quart of water over low heat.
2. Peel and mash bananas with 1/4 teaspoon **cloves**, 1/4 teaspoon **orégano**, and 1 teaspoon **powdered cinnamon**.
3. Knead the mixture, add a pinch of **salt**, and fry in 4 tablespoon **shortening** until slightly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

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

6.7 Examples of color plotting

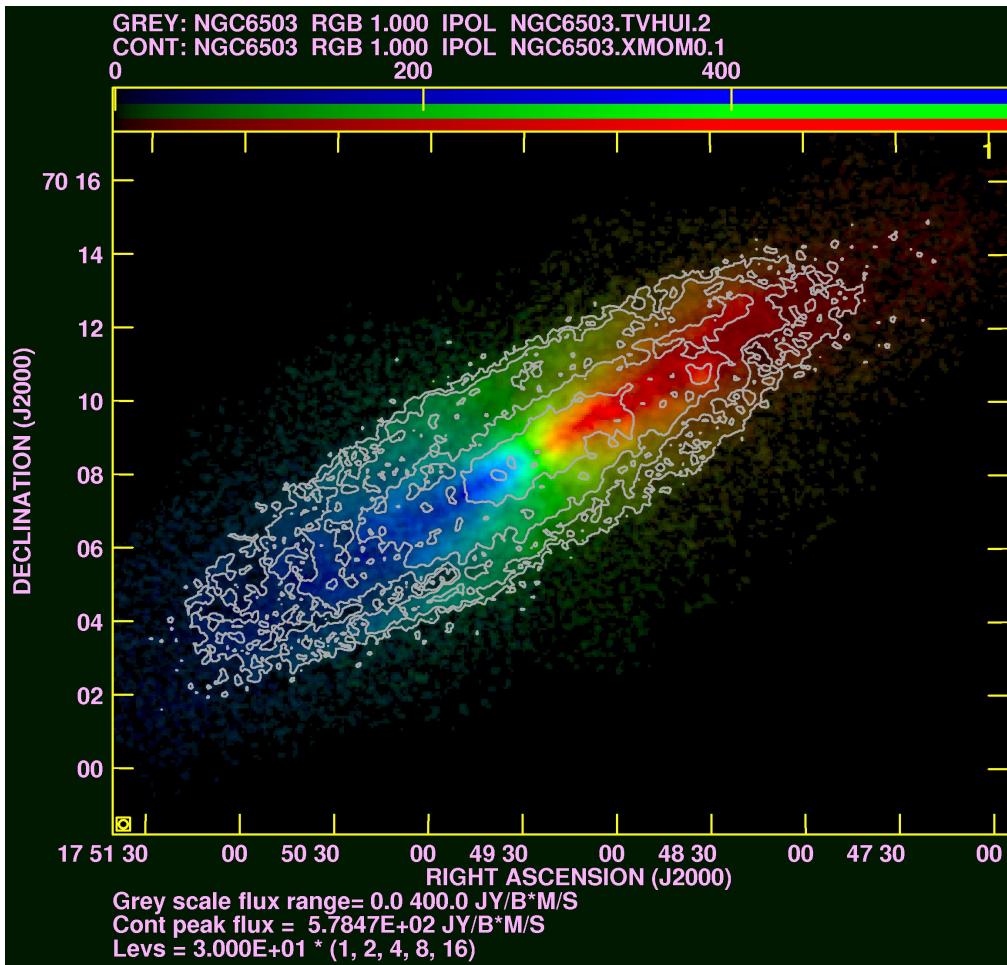
6.7 Examples of color plotting

6. Displaying Your Data



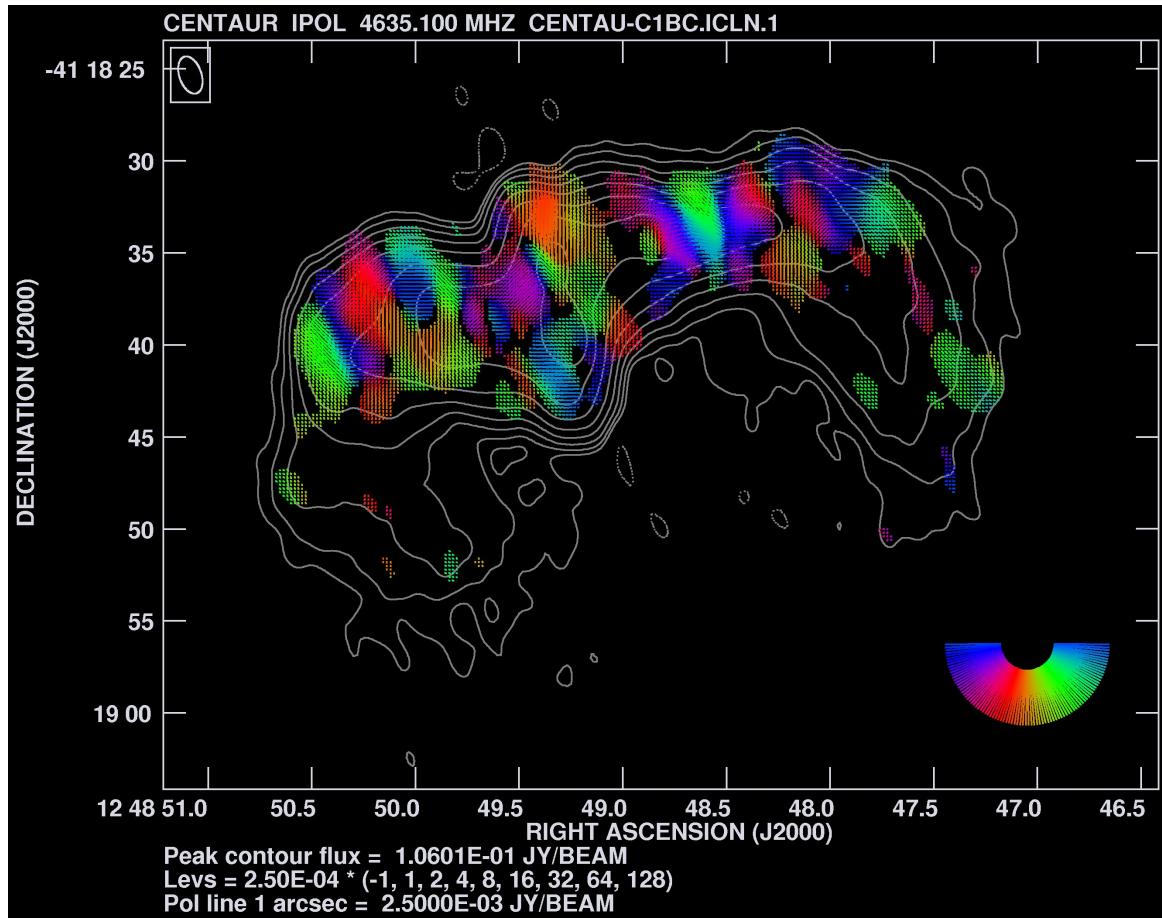
KNTR:	Task to generate a plot file for a contour and grey plot			
DOCONT	1	> 0 => do contours (1 or 2 => which name)	PCUT 0.001 ICUT 0.001	Pol. vector cutoff. P units. Int. vector cutoff. I units.
DOGREY	1	> 0 => do grey scale (1 pr 2 => which name)	DOWEDGE 3	= 3 => put on top using full range of image values
DOVECT	1	> 0 => do polarization vectors (1 or 2 => which is IPOL) Contour or grey or IPOL	STFACTOR 1 CBPLOT 1	Scale star sizes: 0 => none. Position for beam plot:& 1: lower left (default)
INNAME	'1807+698 DEC'	First image name (cube?)	DARKLINE 0.33	Switch to dark lines when grey-scale > DARKLINE 0-1
INCLASS	'ICL001'	First image class		
INSEQ	1	First image seq. #		
INDISK	3	First image disk drive #		
IN3NAME	'1807+698 DEC'	Polarization intensity image: (name) blank => INNAME	LWPLA: Sends plot file(s) to a PostScript printer or file	
IN3CLASS	'QCL001'	(class) blank => 'PPOL'	INNAME '1807+698 DEC' INCLASS 'ICL001'	Image name (name) Image name (class)
IN3SEQ	1	(seq. #) 0 => high	INSEQ 1	Image name (seq. #)
IN3DISK	3	Disk drive #, 0 => any	INDISK 3	Disk drive #
IN4NAME	'1807+698 DEC'	Polarization angle image: (name) blank => INNAME	PLVER 2	Version # of PL file. 0=>last
IN4CLASS	'UCL001'	(class) blank => 'PANG'	FUNCTYPE , ,	'NE','LG','NG','SQ','NQ' else linear
IN4SEQ	1	(seq. #) 0 => high	DPARM *all 0	(1,2) Clip recorded grays
IN4DISK	3	Disk drive #, 0 => any	DODARK 1	Paint dark vectors as "dark"
PIXRANGE	-1.00E-03	Min,Max of image intensity	OFMFILE 'RAINBOW'	Color grey scales....
FUNCTYPE	'SQ'	Image intensity transfer func	DOCOLOR 1	Use PLCOLORS ?
OFMFILE	,		PLCOLORS 1	Line, character, background
LTYPE	3	Type of labeling: 3 standard	0 1	colors - see HELP.
DOALIGN	1	> 0 => images must line up	1 1 1 0.5	
CLEV	0.001	Absolute value for levs	1 0 1 1	
LEVS	-3 -1 1 3 81 243	Contour levels (up to 30). 9 27 *rest 0	0 0 0 0 0 0	
FACTOR	1000	Mult. factor for Pol vector	0 0 0 0 0 0.8	
XINC	3	X-inc. of Pol vectors. 0=>1	0.8 1	
YINC	3	Y-inc. of Pol vectors. 0=>1		

Figure 6.6: **KNTR** does polarization lines, contours, and grey-scale. Then **LWPLA** converts the grey-scale to pseudo-color and colors the lines making dark contours dark but dark polarization lines and stars bright. Data courtesy of Greg Taylor.



KNTR:	Task to generate a plot file for a contour & grey plot	CBPLOT	1	Position for beam plot:
DOCONT	2	> 0 => do contours (1 or 2 => which name)	DODARK	1
DOGREY	1	> 0 => do grey scale (1 pr 2 => which name)	DARKLINE	0.33
INNAME	'NGC6503 ,	First image name (cube?)	LWPLA:	Sends plot file(s) to a PostScript printer or file
INCLASS	'TVHUI '	First image class	INNAME	'NGC6503 ,
INSEQ	2	First image seq. #	INCLASS	Image name (name)
INDISK	3	First image disk drive #	INSEQ	'TVHUI '
IN2NAME	'NGC6503 ,	Contour or grey or IPOL	INDISK	Image name (class)
IN2CLASS	'XMOM0 ,	Second image name	FUNCTYPE	Image name (seq. #)
IN2SEQ	1	Second image class	DPARM	Disk drive #
IN2DISK	3	Second image seq. #	DODARK	'NE','LG','NG', 'SQ', 'NQ'
PIXRANGE	0 400	Second disk drive #	OFMFILE	else linear
FUNCTYPE	'SQ'	Min,Max of image intensity	*all 0	(1,2) Clip recorded grays
		Image intensity transfer func	*	Paint dark vectors as "dark"
		'SQ' Square root	DOCOLOR	Color grey scales....
DOCOLOR	1	Do RGB images as 3-color?	PLCOLORS	Use PLCOLORS ?
LTYPE	-3	Type of labeling: 3 standard	1	Line, character, background
		<0 -> no date/time	0	colors - see HELP.
DOALIGN	1	> 0 => images must line up (see HELP DOALIGN)	0.67	0.67 0 0
DOBBLANK	-1	Draw boundary between blanked	0	0 0 0 0.67
		areas and good areas?	0.67	0.67 0 0
DOWEDGE	3	= 3 => put on top using full	0	0 0 0 0
		range of image values	1	0.7 1 0
			0.1	0

Figure 6.7: KNTR interprets the output of `TVHUI` as a three-color RGB image and overlays moment-0 contours. `LWPLA` adds coloring to the lines, using a less than pure white for both bright and dark contours so that they are not so dominant. Data courtesy of Eric Greisen, Kristine Spekkens, and Gustaaf van Moorsel.

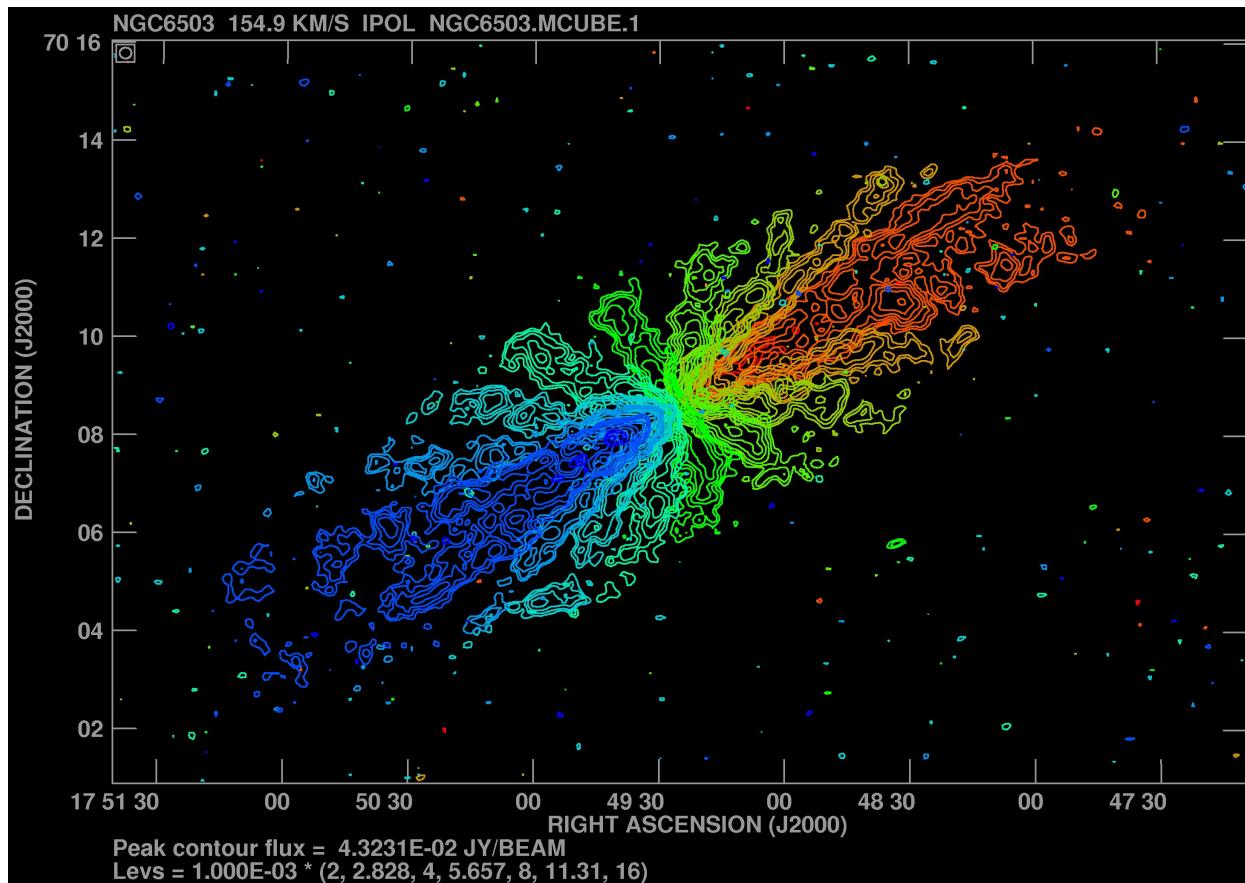


```

PCNTR: Task to generate plot file for contour plus pol. vectors      ROTATE      0      Angle to rotate Pol vector
DOCONT   1              Draw contours? > 0 => yes                  (in degrees)
DOVECT   1              Draw pol. vectors? > 0 => yes             XINC        1      X-inc. of Pol vectors. 0=>1
DOGREY   -1             Draw grey-scale image?                   YINC        1      Y-inc. of Pol vectors. 0=>1
INNAME   'CENTAU-C1BC '  Image name (name)                      PCUT        1.250E-04  Pol. vector cutoff. P units.
INCLASS  'ICLN '        Image name (class)                     ICUT        2.500E-04  Int. vector cutoff. I units.
INSEQ    1               Image name (seq. #)                    POL3COL     160      Color polarization vectors
INDISK   1               Disk unit #                         CBPLOT      4       value in degrees = red
                           Polarization intensity image:          Position for beam plot:
IN2NAME  'CENTAU-C1BC '  (name) blank => INNAME                 4: upper left
IN2CLASS 'QCLN '        (class) blank => 'PPOL'
IN2SEQ   1               (seq. #) 0 => high
IN2DISK  0               Disk drive #, 0 => any
                           Polarization angle image:           LWPLA: Sends plot file(s) to a PostScript printer or file
IN3NAME  'CENTAU-C1BC '  (name) blank => INNAME                 RGBGAMMA    1       1      Gamma correction to apply
IN3CLASS 'UCLN '        (class) blank => 'PANG'                2.5
IN3SEQ   1               (seq. #) 0 => high
IN3DISK  0               Disk drive #, 0 => any
                           Type of labeling:                   DPARM      *all 0      (1,2) Clip recorded grays
LTYPE    -3              <0 -> no date/time                  DOCOLOR     1      before FUNCTYPE (0 to 1)
                           Absolute value for levs            PLCOLORS   0.85      Use PLCOLORS ?
CLEV    2.500E-04        Contour levels (up to 30).           0.9       0.5      Line, character, background
LEVS    -1      1          0.5       0.5      colors - see HELP.
                  2      4          0.9       0.5
                  32     64          0.0       0.0
FACTOR  1000            Mult. factor for Pol vector         0.0       0.0
                           (see HELP)                      0.85      0.85      *rest 0

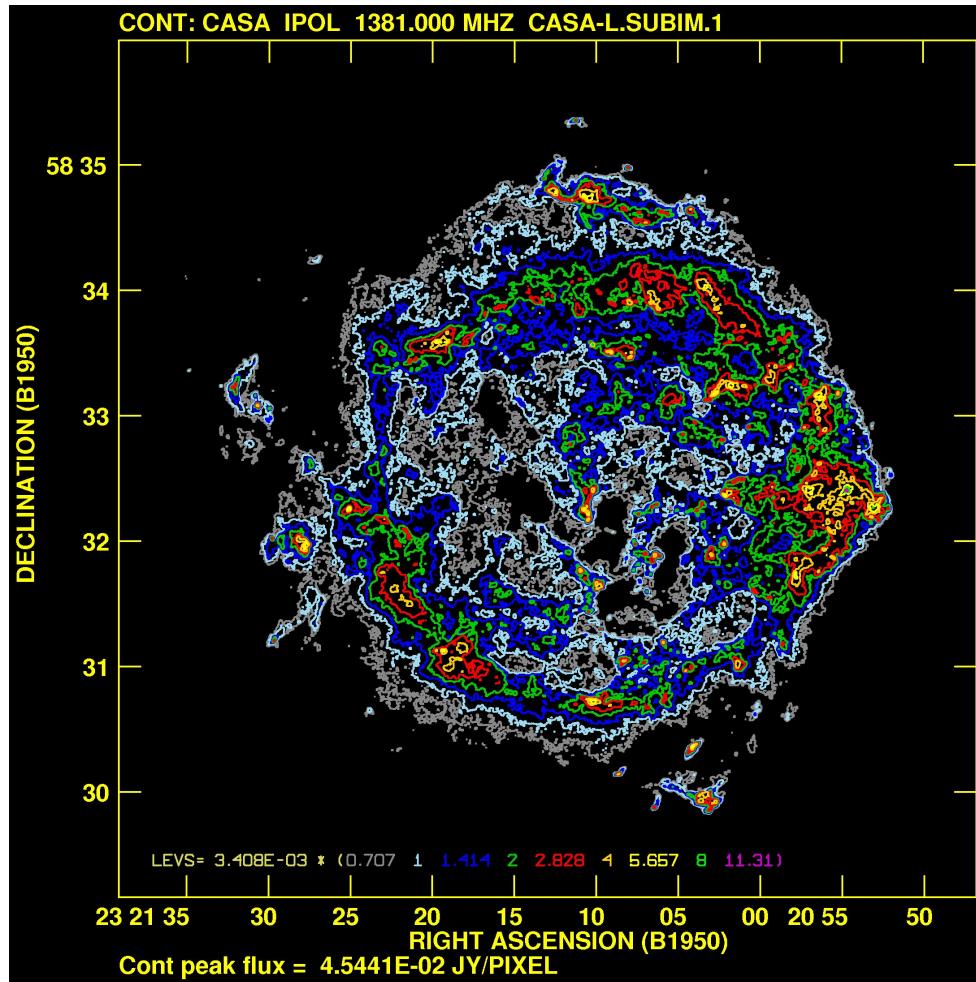
```

Figure 6.8: **PCNTR** plots contours and polarization vectors of Centaurus A. Color is used to show the complex changes in polarization position angle since the angles of short lines cannot be seen accurately. Data courtesy of Greg Taylor. For a discussion of this amazing pattern see Taylor, G.B., Fabian, A.C., & Allen, S.W. 2002, MNRAS, 334, 769, astro-ph/0109337 "Magnetic Fields in the Centaurus Cluster."



PCNTR:	Task to generate plot file for contour plus pol. vectors	LWPLA:	Sends plot file(s) to a PostScript printer or file
DOCONT	1	LPEN	3
DOVECT	-1	RGBGAMMA	1 1
BLC	51 167		Gamma correction to apply
	39 *rest 0	2.5	
TRC	397 393	DPARM	*all 0
	89 *rest 0	DOCOLOR	1
CON3COL	5	PLCOLORS	0.6 0.6 0.06275
	> 0 => overplot contours in		(1,2) Clip recorded grays
	color of multiple planes		Use PLCOLORS ?
	ZINC is CON3COL.		Line, character, background
CLEV	0.001		colors - see HELP.
LEVS	2 2.8284	1 0 1 0.6706	
	4 5.6569	0 0 0 0	
	16 *rest 0	0 0 0 0	
CBPLOT	4	0.6 0.6 0.6 *rest 0	
	Position for beam plot: 4: upper left		

Figure 6.9: PCNTR plots contours every fifth plane from a data cube using colors related to the velocity. LWPLA adds coloring to the labeling and background and applies a gamma correction to blue. Data courtesy of Eric Greisen, Kristine Spekkens, and Gustaaf van Moorsel.

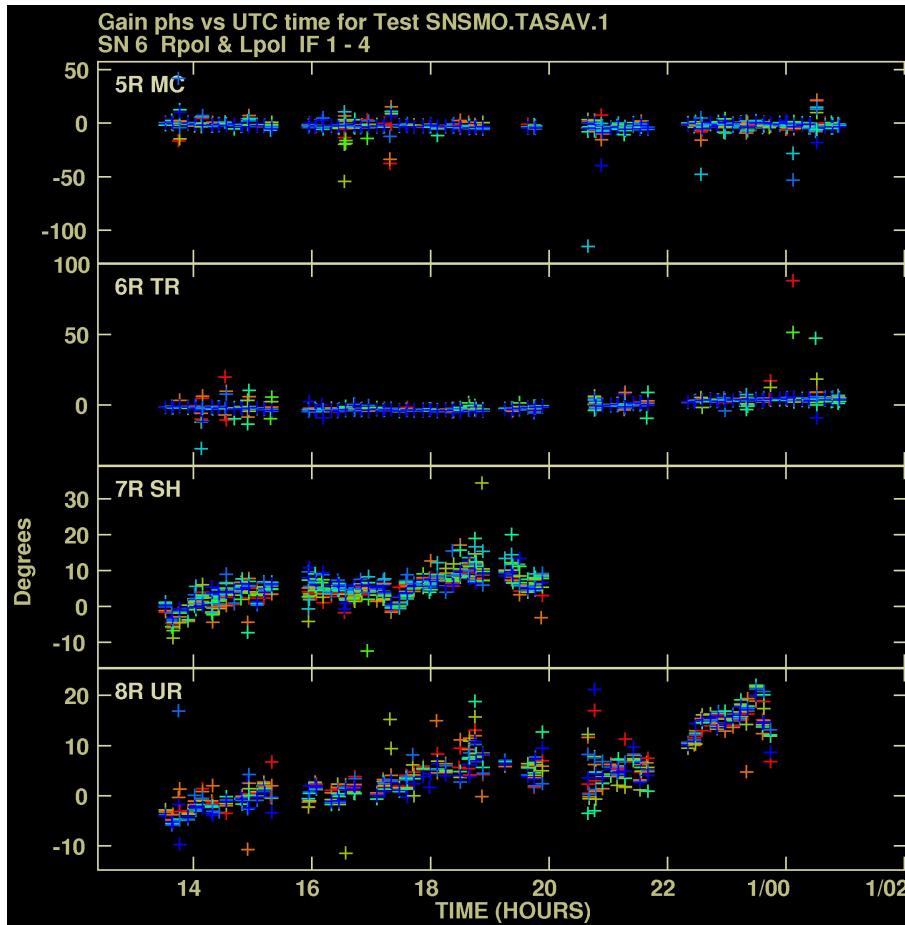


```

KNTR: Task to generate a plot file for a contour & grey plot      RGBLEVS      0.2562      0.2562      Color each value of LEVS
DOCONT    1           > 0 => do contours      0.2562      0.3511      0.7297      0.9035
          (1 or 2 => which name)
DOGREY   -1           > 0 => do grey scale      0           0           1           0
          (1 pr 2 => which name)
DOVECT    -1           > => do polarization vectors      0.6205      0           1           0
          (1 or 2 => which is IPOL)
LTYPE     -3           Type of labeling: 1 border,      0.8503      0           0           0
          2 no ticks, 3 standard, 4 rel
          to center, 5 rel to subim cen
          6 pixels, 7-10 as 3-6 with
          only tick labels      0.6594      *rest 0
          <0 -> no date/time
          special values for RGBLEVS
          Absolute value for levs      OFMFILE      *all ,      Color grey scales....
          (used only if PLEV = 0).      DOCOLOR      1           Use PLCOLORS ?
CLEV      0.003408      Contour levels (up to 30).      PLCOLORS      1           1           Line, character, background
          0.7071      1           0           0.06275      colors - see HELP.
          1.4142      2           2.8284      4           1           0           0.6706
          5.6569      8           11.3137      *rest 0           0           0           0
CON3COL    0           Color the contours by plane      0           0           0           0
          1           1           1           0           0           0
          0           0           0           0           0           0
          1           1           *rest 0

```

Figure 6.10: **KNTR** plots contours of Cassiopeia A with each contour level separately colored under control of adverb **RGBLEVS**. The values of **RGBLEVS** were set by a procedure call **STEPLEVS(10)** made available by **RUN SETRGBL**. Image is from the *Images from the Radio Universe CD*, 1992, NRAO with the particular image from Anderson M., Rudnick, L., Leppik, P., Perley, R. & Braun, R. 1991, ApJ, 373, 146.



```

SNPLT: Task to plot selected contents of SN, TY, PC or CL file.      FACTOR      1.5
INEXT   'SN'           Input 'SN', 'TY', 'PC', 'CL'
INVERS   6             Input table file version no.    LTYPE       -3
STOKES   ,             Stokes type to plot: R, L,
                  RR, LL, RRLL, DIFF, RATO
BIF      0             First IF to plot, 0=>1.
EIF      0             Last IF to plot 0 -> highest
ANTENNAS 5             6             Antennas to plot 0=>all
                  7             8             *rest 0
NPLOTS   4             Number of plots per page
XINC     1             Plot every XINC'th point
OPTYPE   'PHAS'        Data to be plotted:
                  'PHAS', 'AMP', 'DELA', 'MDEL',
                  'RATE', 'TSYS', 'TANT', 'ATM',
                  'GEO', 'DOPL', 'SNR', 'SUM'
                  'CCAL', 'DDLY', '='phas.
OPCODE   'ALSI'        Type of plot:
                  'IFDF' => diff BIF and EIF
                  'IFRA' => ratio BIF and EIF
                  'ALIF' => combine all IFs
                  'ALST' => combine all Stokes
                  'ALSI' => all IFs & Stokes
DO3COL   1             > 0 use 3-color symbols for
                  ALIF, ALST, ALSI OPCODEs and
                  SUM OPTYPE.
XAXIS    0             Variable data is to be
                  plotted against, 0=>time.
                                             LWPLA: Sends plot file(s) to a PostScript printer or file
                                             PLVER    0             Version # of PL file. 0=>last
                                             INVERS   0             PL file version #, upper
                                             limit if > PLVER
                                             ASPMM    0             Arc sec. per mm. 0=self scale
                                             LPEN     3             Pen width (dots).
                                             RGBGAMMA 1             1             Gamma correction to apply
                                             DPARM    *all 0          (1,2) Clip recorded grays
                                             before FUNCTYPE (0 to 1)
                                             Use PLCOLORS ?
                                             PLCOLORS 0.85         0.85         Line, character, background
                                             0.65         0.6           colors - see HELP.
                                             0.6           0             0             0
                                             0             0             0             0
                                             0.75         0.75         0.5           0
                                             0             0             0             0
                                             0.75         0.75         0.5           *rest 0

```

Figure 6.11: SNPLT plots phases for four antennas with color indicating polarization and IF channel. Stokes 1, IF **BIF** is pure red changing through yellow, green, and cyan to Stokes 2, IF **EIF** as pure blue. When all symbols lie on top of each other, the last one (pure blue) will dominate.

7 Analyzing Images

In order to obtain useful astronomical information from the data, software exists for the analysis of images, combining of images, estimating of errors, etc. Only a few of the programs are described in any detail here; the others should be self-explanatory using the **HELP** and **INPUTS** files for the tasks listed in Chapter 13. A complete list of software in *ATPS* for the analysis of images may also be obtained at your terminal by typing **ABOUT ANALYSIS C_R**.

7.1 Combining two images (COMB)

The task **COMB** is a general purpose program for combining two images, pixel by pixel, to obtain a third image. Many options are available and, as a first example, we illustrate inputs to subtract a continuum image from a spectral line image cube.

7.1.1 Subtracting a continuum image from an image cube

A common method to obtain a spectral data cube containing only line signal without any continuum emission is to create a line-free continuum image C , and subtract it from the data cube L . For a more general discussion and alternative methods see § 8.3. **COMB** can be used to this purpose as follows:

- | | |
|---|---|
| > TASK 'COMB' ; INP C_R | to review the required inputs. |
| > INDI 0 ; MCAT C_R | to help you find the catalog numbers of C and L . |
| > INDI n1 ; GETN ctn1 C_R | to select the L image cube from disk $n1$ catalog slot $ctn1$. |
| > IN2D n2 ; GET2N ctn2 C_R | to select the C image from disk $n2$ catalog slot $ctn2$. |
| > OUTN 'xxxxx' C_R | to specify $xxxxx$ for the name of the continuum-free image cube. |
| > OUTC 'ccc' C_R | to specify ccc for the class of the continuum-free image cube, e.g., LCUBE. |
| > OPCODE 'SUM' C_R | to select the addition algorithm. |
| > APARM 1, -1 C_R | to specify that we want $+1 \times L - 1 \times C$. |
| > GO C_R | to compute the continuum-free, line-only output cube. |
| Once COMB task has terminated with the message COMB: APPEARS TO END SUCCESSFULLY , you should find the requested image in your catalog: | |
| > MCAT C_R | to list the images in your catalog. |

7.1.2 Polarized intensity and position angle images

As a second example, we derive the polarization intensity and angle from the Q and U Stokes parameter images. To compute a polarized intensity image, enter:

- | | |
|--|--|
| > TASK 'COMB' ; INP C_R | to review the required inputs. |
| > INDI 0 ; MCAT C_R | to find the catalog numbers of the Q and U images. |
| > INDI n1 ; GETN ctn1 C_R | to select the Q image from disk $n1$ catalog slot $ctn1$. |
| > IN2D n2 ; GET2N ctn2 C_R | to select the U image from disk $n2$ catalog slot $ctn2$. |

- > **OUTN** 'xxxxx' C_R to specify *xxxxx* for the name of the polarized intensity image.
 - > **OUTC** 'ccc' C_R to specify *ccc* for the class of the polarized intensity image, e.g., PCLN.
 - > **OPCODE** 'POLC' C_R to select the $\sqrt{Q^2 + U^2}$ algorithm with correction for noise.
 - > **BPARM** *ns1 , ns2* C_R to specify the noise levels of the 2 images.
 - > **GO** C_R to compute the corrected, polarized intensity image.
- AIPS will write the message **TASK COMB** BEGINS followed by a listing of the POL. INTENSITY algorithm. While it is running, you can prepare the inputs to make a polarization position angle image. Type:
- > **OUTN** 'yyyyy' C_R to specify *yyyyy* for the name of the polarization angle image.
 - > **OUTC** 'ddd' C_R to specify *ddd* for the class of the polarization angle image, e.g., PSIMAP, CHICLN.
 - > **OPCODE** 'POLA' C_R to select the $\frac{1}{2} \tan^{-1} \left(\frac{U}{Q} \right)$ algorithm.

Once **COMB** has finished, enter:

- > **GO** C_R to compute the polarization angle image.

7.1.3 Other image combination options

COMB may also be used to rescale images, and to compute spectral indices, optical depths, etc. Type:

- > **HELP COMB** C_R to review the available options.

The **OPCODE** options are:

'SUM'	Addition	$a_1 M_1 + a_2 M_2 + a_3$
'SUMM'	Addition	$a_1 M_1 + a_2 M_2 + a_3$ except blanked pixels replaced with 0
'MEAN'	Average	$a_1 M_1 + a_2 M_2$ except M_j where M_i blanked
'MULT'	Multiplication	$a_1 M_1 M_2 + a_2$
'DIV'	Division	$a_1 M_1 / M_2 + a_2$
'SPIX'	Spectral Index	$a_1 \ln(M_1/M_2) / \ln(\nu_1/\nu_2) + a_2$ where $M_1 > a_3$ and $M_2 > a_4$
'OPTD'	Opacity	$a_1 \ln(a_3 M_1/M_2 + a_4) + a_2$ where $M_1 > a_5$ and $M_2 > a_6$
'POLI'	RMS sum	$a_1 \sqrt{M_1^2 + M_2^2} + a_2$
'POLC'	RMS sum	$a_1 C(M_1, M_2) \sqrt{M_1^2 + M_2^2} + a_2$ where <i>C</i> is a noise-based correction for Ricean bias
'POLA'	Arctangent	$a_1 \tan^{-1}(M_2/M_1) + a_2$ where $\sqrt{M_1^2 + M_2^2} > a_3$
'REAL'	Real part	$a_1 M_1 \cos(a_2 M_2) + a_3$
'IMAG'	Imaginary part	$a_1 M_1 \sin(a_2 M_2) + a_3$
'RM'	Rotation measure	$a_1 \text{RM}(M_1, M_2) + a_2$
'CLIP'	Clipping	M_1 except blanked where $a_1 > M_2 > a_2$ or $a_2 > a_1 > M_2$ or $M_2 > a_2 > a_1$

where the a_i are user-adjustable parameters — specified by **APARM** — and M_1 and M_2 are the images selected by **INNAME**, etc. and by **IN2NAME**, etc., respectively. **COMB** may be instructed to write an image of the estimated noise in the combination in addition to the direct result of the combination. These noise images may be used as inputs to **COMB** and, e.g., **BLANK** to control later computations. When combining numerous images together, set **DOHIST**=-2 to prevent history files from becoming much larger than the images.

CONVL can compute the cross-correlation of two images and report the location of the peak cross-correlation.

7.1.4 Considerations in image combination

COMB can use images of the uncertainties in the first two input images to control the computation of the output. The new task **RMSD** may be used to create an image of the rms in an image, computing the rms self-consistently (robust, histogram, and median absolute deviation methods available) in windows surrounding each pixel. Task **FLATN** can compute weight and noise images corresponding to mosaiced images.

For some applications of **COMB**, undefined pixel values may occur. For example, if the spectral index is being calculated and the intensity level on either image is negative, the index is undefined. In this case, the pixel value is given a number which is interpreted as undefined or “blanked.” Blanking also arises naturally in operations of division, opacity, polarization angle, and clipping and, of course, the input images may themselves be blanked. In addition, the output image can be blanked (set **BPARM(4) = 0**) whenever either $M_1 < \text{APARM}(9)$ or $M_2 < \text{APARM}(10)$. Alternatively, blanking may be done on the basis of the estimated noise (set **BPARM(4) = 1**) or signal-to-noise ratio (set **BPARM(4) = 2**) in the combination. See **HELP COMB** C_R for a description of these options and certain limitations in their use. With **APARM(8) = 1** C_R , the user may specify that all undefined pixels are to be assigned an apparently valid value of zero, rather than the “magic” undefined-pixel value. Alternatively, the task **REMAG** can be used to replace blanked pixels in the output image with a user-specified value.

When combining two or more images, **COMB**, **PCNTR**, *et al.* must decide which pixels in the 2nd image go with which pixels in the 1st image. The user input parameter **DOALIGN** controls this process. A value of 1 requires the two headers to be correct and sufficiently similar that an alignment by coordinate value is possible. A value of -2 tells the programs to ignore the headers and align by pixel number. Enter **HELP DOALIGN** C_R for details and intermediate options. In some cases, the images may have been created on different grids which are correctly described in the headers. The observations, for example, could have differed in the phase reference position or projective geometry used or the imaging could have been done with different axis increments. Such images should *not* be combined directly. Instead, the header of one should be used as a template for re-gridding the other. tasks **HGEOM** and **OHGEO** provide this service with good interpolation methods. See § 7.6.1 and type **EXPLAIN HGEOM** C_R or **EXPLAIN OHGEO** C_R for more information.

7.2 Combining more than two images (SUMIM, SPIXR, STACK)

The task **SUMIM** is used to sum or average any number of images. Since *AIPS* has only a limited number of adverbs of the kind **INNAME**, **IN2NAME**, etc., **SUMIM** requires that all input images have identical **INNAME** and **INCLASS**. The input images are then specified by **INSEQ** (the sequence number of the *first* input image), **IN2SEQ** (the sequence number of the *last* input image), and **IN3SEQ** (the increment in sequence number). All input images have to reside on the *same* disk.

- | | |
|--|--|
| > TASK 'SUMIM' ; INP C_R | to review the required inputs. |
| > INDI 0 ; MCAT C_R | to help you find the catalog number of the first input image. |
| > INDI n ; GETN ctn C_R | to select the first input image from disk <i>n</i> catalog slot <i>ctn</i> . |
| > IN2SEQ s C_R | to specify the sequence number of the last image to be included. |
| > IN3SEQ 0 | to specify the increment in sequence number (= 1). |
| > OUTN 'xxxxx' C_R | to specify <i>xxxxx</i> for the name of the output image. |
| > OUTC 'ccc' C_R | to specify <i>ccc</i> for the class of the output image. |
| > FACTOR f C_R | to specify the factor with which to multiply each image before adding. <i>f</i> = 1 leads to summation, <i>f</i> = 0 defaults to the inverse of the number of input images (average) |

> **GO** C_R

to start **SUMIM**.

This is a very noisy way to make a line-sum image. For more serious work, use **BLANK** (§ 7.4) and **XMOM** (§ 8.6) instead.

A popular analysis technique has been developed in which images centered upon objects of some chosen kind are simply added up even though the objects' emission is less than the noise. The belief is that, if there is real emission just below detectability in individual images, the summed image will reduce the noise and reveal the weak emission. In 31DEC15, the task **STACK** attempts such an analysis doing a weighted mean or median of a matched set of images ignoring the coordinates.

The task **SPIXR** is intended to fit spectral indexes optionally including curvature to a cube of image planes. This cube is build with **FQUBE** to make an FQID axis out of irregularly spaced frequencies. The cube is transposed by **TRANS** to put the FQID or, if build with **MCUBE**, frequency axis first. Then **SPIXR** will do a least squares fit for spectral index. Again, the results will be noisy unless the initial images have been blanked and converted to similar spatial resolution.

7.3 Image statistics and flux integration

The task **IMEAN** is used to determine the statistics of the image inside, or outside a specified rectangular or circular area. It derives the minimum and maximum value and location, the rms, the average value and, if the image has been Cleaned, an approximate flux density within the area. A typical run might be:

- > **TASK 'IMEAN'** ; **INP** C_R to list the input parameters.
- > **INDI** n ; **GETN** ctn C_R to select the image file from disk n catalog slot ctn.
- > **BLC** n1, n2 ; **TRC** m1, m2 C_R to set the window from (n1,n2) to (m1,m2) — or use **TVWIN** with the cursor on the TV.
- > **DOHIST TRUE** C_R to make a plot file of the pixel histogram.
- > **PIXRANGE** x1, x2 C_R to set the range of the histogram from x1 to x2.
- > **NBOXES** n C_R to set the number of boxes in the histogram.
- > **GO** C_R to run the task.

A circular aperture may be specified with **BLC** = -1, radius ; **TRC** = X_c, Y_c. **IMEAN** attempts to determine the true noise of the image by fitting the peak of the histogram and reports both that result and the one found by including all pixels within the window. **IMEAN** now gets the initial guess for the true noise from a robust computation over the input window and returns the adverbs **PIXSTD** and **PIXAVG** from the histogram fit to the AIPS program. **IMEAN** now fills an additional output adverb such that **TRIANGLE**(i) is the brightness level exceeded by only i per cent of the image.

The statistics will appear in the **AIPS** window. For a hard copy type:

- > **PRTASK 'IMEAN'** ; **PRTMSG** C_R with **PRI0** ≤ 5.

To see the histogram of the intensities, an example of which is shown in § 6.3.2.3, type one of:

- > **GO TKPL** C_R to display the histogram in the TEK window.
- > **GO LWPLA** C_R to display the histogram on a PostScript printer.

The verbs **TVSTAT** and **IMSTAT** provide similar functions to **IMEAN** without the histogram and true rms options. Both return their results as AIPS parameters **PIXAVG** (mean), **PIXSTD** (rms), **PIXVAL** (maximum), **PIXXY** (pixel position of the maximum), **PIX2VAL** (minimum), **PIX2XY** (pixel position of the minimum). **IMSTAT** uses the same file name, **BLC**, and **TRC** parameters as **IMEAN** including the circular aperture convention. It is useful to prepare the initial rms guess for that task although the **PIXSTD** it returns will often be a factor of several too large. **TVSTAT**, however, works on the image plane currently displayed on the TV and is not limited to a single rectangular area. Instead, the TV cursor is used to mark one or more

polygonal regions over which the function is to be performed. Type **EXPLAIN TVSTAT** C_R for a description of its operation.

The interactive task **BLSUM** employs a method similar to that of **TVSTAT**. The TV cursor is used to mark a region of interest in a “blotch” image. Then **BLSUM** finds the flux in that region not only in the blotch image but also in each plane (separately) of a second image. More than one region of interest may be done in any given execution of the task. In spectral-line problems, the blotch image is often the continuum or the line sum while the second image is the full “cube” in almost any transposition. The spectrum obtained may be saved as a SLice file for further analysis and display. In 31DEC14, the spectra may be saved as true plot files as well as printer plots and the flux summation may be weighted by the values (or their squares) found in the blotch image. Numerous continuum applications also exist (e.g., polarization, comparison across frequency). Type **EXPLAIN BLSUM** C_R for a description of the operation.

The verb **IMDIST** is used to measure the angular distance and position angle between two pixel positions in up to two images. The separation is returned as adverb **DIST**. Verb **TVDIST** allows you to select the two pixels interactively from the TV display.

The verb **IMCENTER** may be used to determine the intensity-weighted centroid of a rectangular or circular portion of an image. The verb returns adverbs **PIXXY**, **COORDINA**, and **ERROR** giving the pixel and physical coordinates of the centroid and an indicator of success or failure.

7.4 Blanking of images

In order to determine accurate flux values in images, or moments of velocity profiles, it is desirable to restrict the integrations to pixels that contain emission, or, in other words, to exclude pixels that contribute only noise. If this is not done, the inclusion of noisy pixels will increase the rms in the derived integrated value to an unacceptable extent. The task **BLANK** gives the user the opportunity to replace pixels containing pure noise with values that *AIPS* and its tasks interpret as *undefined*. The decision whether a certain pixel contains pure noise, or carries some emission, can be made subjectively (using the TV) or in a more objective fashion (see below for an example). In all cases, **BLANK** creates an output image which is a copy of the input image with some pixels replaced by undefined values, or — if the user specifies it — by the value zero.

The most straightforward use of **BLANK** is to apply a cutoff to the input image, e.g. let **BLANK** replace with an undefined value every pixel in the input image that lies below a specified. e.g., 3σ noise level. This effectively removes almost all noisy pixels. The disadvantage is that this method also removes any *signal* below the 3σ noise level. Since a substantial fraction of the total flux may be “hidden” in pixels below 3σ , this method prevents an accurate total flux determination. Another straightforward use of **BLANK** is to remove all pixels outside a user-specified radius. This allows blanking regions for which the primary-beam corrections, and hence the noise levels, are large.

A better way to perform the blanking is one which is not based on the pixel values in the input image itself, but on those in a *second* input image. Typically this is a convolved (spatially and/or in velocity) version of the input image, which has a higher signal to noise for extended emission than the input image. In the example given here we have the input image I_1 of full spatial resolution, and a convolved version of this input image I_2 with a linear beam size roughly twice full resolution. Careful inspection of this second image has shown that there are no outlying noise peaks above f mJy/beam. **BLANK** is then run as follows:

- > **TASK 'BLANK'** ; **INP** C_R to review the required inputs.
- > **INDI 0** ; **MCAT** C_R to help you find the catalog numbers of I_1 and I_2 .
- > **INDI n1** ; **GETN** $ctn1$ C_R to select I_1 from disk $n1$ catalog slot $ctn1$.
- > **IN2D n2** ; **GET2N** $ctn2$ C_R to select I_2 from disk $n2$ catalog slot $ctn2$.
- > **OUTN 'xxxxx'** C_R to specify $xxxxx$ for the name of the blanked output image.

- > **OUTC** 'ccc' C_R to specify *ccc* for the class of the blanked output image.
- > **OPCODE** 'IN2C' C_R to specify that the blanking is performed using pixel values in a *second* input image.
- > **DPARM(3)** *sim* – *f*, *f* C_R to set **DPARM(3)** and **DPARM(4)** to specify that all pixels with fluxes in the second input image in the interval (*-ff*) should be blanked.
- > **GO** C_R to compute the blanked output image.

The task **REMAG** can be used to replace blanked pixels by a value to be specified by the user.

The *AIPS* TV display may be used to do a more subjective blanking with this task. Set **OPCODE** 'TVCU' C_R to display the image, one plane at a time in any transposition. You will be prompted to set "blotch" regions (much like **TVSTAT** and **BLSUM**) to define the areas to be blanked. This is one method for having different regions of signal at different spectral channels. There are also four windowing methods for blanking spectral-line cubes which have been transposed to have the frequency axis be first. In these methods, a window (range of spectral channels) about the peak signal in each spectrum is retained.

The task **RMSD** may be used to write a version of the input image blanking pixels below *N* times the rms in the image, computing the rms self-consistently in windows surrounding each pixel.

7.5 Fitting of images

There are three tasks and two verbs which estimate the position and intensity of a component on a two-dimensional image. The simplest and fastest methods are the verb **IMCENTER** and **MAXFIT**. The latter fits a two-dimensional parabola to the maximum within a few pixels of an image position, and gives the peak and its position. The tasks **IMFIT** and **JMFIT** are similar and fit an image subsection with up to four Gaussian components with error estimates. Tasks **SAD** and **TVSAD** attempt to automate the process of finding and fitting Gaussian components in an image. Additionally, in one dimension, the task **SLFIT** fits Gaussian components to slice data and the task **XGAUS** fits Gaussian components to each row of an image.

7.5.1 Centroid fits (IMCENTER) and parabolic fit to maximum (MAXFIT)

You may determine a centroid for a region in an image with the verb **IMCENTER**. Set the name parameters for the desired image and then define the region with adverbs **BLC** and **TRC**, perhaps using **TVLOAD**; **TWIN** C_R. Set **FLUX** if you wish to limit the computations to pixel values greater than **FLUX**. Then

- > **IMCENTER**; **IMVAL** C_R to find the value at the centroid.

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

- > **EXPLAIN MAXFIT** C_R to get a good explanation of the algorithm.

The inputs should be self-explanatory. The **IMSIZE** parameter can be important in crowded fields. **MAXFIT** can be used conveniently by first displaying the image on the TV and then typing:

- > **IMXY** ; **MAXFIT** C_R

First the cursor will appear on the TV. Move it close to a maximum, press the left mouse button, and hit button A, B, C, or D. The fit will appear in your *AIPS* window. Adverb values **PIXXY**, **PIXVAL**, **COORDINA**, and **ERROR** will be set appropriately. Adverb **FSHIFT** will also be set to suggest the shift needed to place the source directly on a pixel.

7.5.2 Two-dimensional Gaussian fitting (IMFIT)

A more sophisticated least-squares fit of an image is obtained with **IMFIT**, which fits an image with up to four Gaussian components and attempts to derive error estimates. A linear or curved, two-dimensional “baseline” may also be fitted. A sample set-up is as follows:

> TASK 'IMFIT' ; INP C _R	to list the input parameters.
> INDI n ; GETN ctn C _R	to select the image from disk n catalog slot ctn.
> BLC n1, n2 ; TRC m1, m2 C _R	to set the area to be fitted as (n1,n2) to (m1,m2) — or use TVWIN with the cursor on the TV.
> NGAUSS 2 C _R	to set the number of components to be fitted to 2.
> CTYPE 1, 1 C _R	to have both components be Gaussians.
> GMAX 0.34 0.20 C _R	to give estimates of peak intensity in Jy.
> GPOS 200, 100, 210, 110 C _R	to give estimates of the pixel locations of each component.
> GWID 6 4 20 6 4 20 C _R	to give estimates of component sizes in pixels. In this case, each component has a FWHM of 6 by 4 pixels with the major axis at position angle 20 degrees.
> DOWID FALSE C _R	to hold all of the widths constant (if required).
> STVERS 0 C _R	to have the results places in a new “stars” extension file for later plotting.
> INP C _R	to review inputs.
> GO C _R	to run the task.

To improve accuracy, include as small an area as possible in the fit. In some cases, it is useful to hold some of the parameters constant, particularly when fitting a complex clump of emission with several components. The parameters can interact. Error estimates are given for each component. **IMFIT** will sometimes fail to converge in complicated regions. When this happens, you might try using the task **JMFIT**, which is very similar in function, but uses a different mathematical method to minimize the rms. Comparison of the results of **IMFIT** and **JMFIT** will sometimes be instructive. The tasks will correct the results for the effects of the primary beam and bandwidth smearing if you wish. These tasks return adverbs **FMAX**, **FPOS**, and **FWIDTH** with the answers to the fit, **DOMAX**, **DOPOS**, and **DOWIDTH** with the uncertainties in the fits, and **FSHIFT** with the shift needed to place the first component directly on a pixel. It is wise to treat the results of **MAXFIT**, **IMFIT** and **JMFIT** with caution. The estimates of the errors, in particular, are based on theory and on trials of deconvolution over a range of widths.

In 31DEC14, the verb **MFITSET** may be used to set many of the input adverbs for **IMFIT** and **JMFIT** from the image displayed on the TV. The verb sets the image name adverbs, then directs you to set the fitting window (like **TVWINDOW**), and then to point at the peak of Gaussian component 1 (sets **GMAX**(i) and **GPOS**(*,*)), then its major axis half-power point (sets **GWIDTH**(1,i) and **GWIDTH**(3,i)), then its minor axis half-power point (sets **GWIDTH**(2,i)). Push buttons A, B, or C to set each of these points or push button D to stop the process and set **NGAUSS**. Instructions will appear on your terminal.

In older versions, or alternatively, use **RUN INPFIT C_R** (see § 12.2.1) to obtain a procedure which will help to supply input parameters to **IMFIT**. This **RUN** file loads a procedure called **INPFIT** into AIPS. To invoke it, load the image which you want to fit onto the TV with **TVALL** and type **INPFIT (3) C_R** to specify three components. The procedure will prompt you to set the desired sub-image window with the TV cursor (it uses verb **TVWINDOW**) and then to point the TV cursor at the peaks of each of the Gaussians, click the left mouse button when the cursor is correctly placed, and push button A, B, C, or D. The inputs **GMAX**, **GPOS**, **BLC**, and **TRC** are set in this way.

7.5.3 Source recognition and fitting (SAD, TVSAD)

The task **SAD** (§ 10.4.4) attempts to find all sources in a sub-image whose peaks are brighter than a given level. It searches the sub-image specified by **BLC** and **TRC** for all points above this level and merges such points in contiguous “islands.” For each island, initial estimates of the strength, size, and number of components are generated. Then the fitting algorithm used in **JMFIT** is called to determine the least square Gaussian fit. Solutions which fail to meet certain criteria can be retried as two components and, if they still fail, rejected. **SAD** is a task with many adverbs, a full description of which would be beyond the scope of this *CookBook*. Enter **EXPLAIN SAD** \mathcal{C}_R for a full description of this task and its parameters. The effects of bandwidth smearing and the primary beam may be corrected. **SAD** produces a Model-Fit extension file which may be converted to a stars file (§ 6.3.2) with **MF2ST**. The MF file may be printed with **MFprt** in formats suitable for **STARS** and in formats which may be used, with task **BOXES**, to prepare Clean boxes for input to the imaging tasks. **SAD** can now directly attach a “stars” extension file to the input image containing the fit Gaussians. Plot tasks can then overlay them on images.

In 31DEC14, a version of **SAD** called **TVSAD** appeared and an *AIPS Memo*¹ was written describing its use in detail. Like **SAD** it finds source islands and makes an initial guess. At this point it displays an interpolated image of the area surrounding the island and marks it with the island boundary and the initial guesses (plotted as ellipses at the half-power point). You may choose from a TV menu to enhance the display, to change the island boundary, to change one of the initial guesses, to change the number of Gaussians guessed and set their values (like verb **MFITSET**), to have the fit attempted, or to reject this island and go on to the next. After the fit, a residual image is displayed with the island boundary and fit Gaussians marked. At this stage you may accept the fit, go back to the previous menu to try again, or reject this island and go on the next island. You may turn off the TV interaction and have the task run automatically. It will turn the TV interaction back on if it finds a fit that does not meet the acceptance criteria. Finally, **TVSAD** writes out the residual image, printer displays, and extension files exactly like **SAD**.

7.5.4 Gaussian fits to slices (SLFIT)

You can generate a one-dimensional slice (profile) through any plane (characterized by the first two coordinates) of an image file using the *AIPS* task **SLICE**. The output file is appended to the image file as an SL extension file. Slices are computed along lines in the two-dimensional image joining any valid pair of points selected by **BLC** and **TRC**. A slice along the third axis, at a single pixel (not necessarily integers) in the first two axes, may also be prepared. Slices directly along any single axis may be saved without interpolation if desired. If the slice is directly along an FQID axis, **SLICE** will output a slice which is linearly sampled in frequency. Tasks **ISPEC**, **BLSUM**, and **TVSPC** may also save slice extension files. These are somewhat different in that they can represent a sum over an area in the sky as a function of frequency. Nonetheless, the slice file display and fitting software is of considerable use with these as well. The set of software dealing with slice file analysis and display can be obtained on your terminal by typing **ABOUT ONED** \mathcal{C}_R . The list is also given in Chapter 13.

To generate a slice:

> **TASK 'SLICE' ; INP** \mathcal{C}_R to review the inputs to **SLICE**.

Use **INDISK** and **GETNAME** to select the input image. The beginning (**BLC**) and ending (**TRC**) points for the slice can be specified conveniently using the TV cursor if the image to be sliced is first displayed on the TV with **TVLOD** or **TVALL**. To set these points with the TV, type:

> **SETSLICE** \mathcal{C}_R

then set the TV cursor to the desired beginning point for the slice, press the left mouse button, and repeat for the ending point for the slice. Note that, for slices, **BLC** need not be below or to the left of **TRC**. Finally:

¹Greisen, E. W. 2014, “**TVSAD**: interactive search and destroy” AIPS Memo 119, <http://www.aips.nrao.edu/aipsdoc.html>

> **GO** C_R to generate the slice file.

Slice files may be output as ASCII text files using the **OUTTEXT** adverb. They contain detailed pixel and coordinate information as well as the interpolated slice values. In 31DEC16, they may also be printed later by **SLPRT** including any one of the models fit to it. Slice files are archived in your disk catalog as SL extensions to the image file from which they were derived. Running **SLICE** again with new parameters does not overwrite the slice file, but makes another with a higher “version” number. To review and/or delete slice files, follow the instructions for **EXTLIST** and **EXTDEST** of plot files in § 6.3 above, but use **INEXT** 'SL' C_R in place of **INEXT** 'PL' C_R.

When **SLICE** has terminated, the file may be plotted in the TV display on your workstation using:

> INP TVSLICE C _R	to review the inputs to verb TVSLICE .
> INEXT 'SL'; EXTL C _R	to find the intensity range and number of points in the interpolated slice.

The default scales will plot all slice points on a vertical scale from the slice minimum to the slice maximum. You can alter the part of the slice that is plotted and the vertical scale by specifying, for example:

> BDROP 100 ; EDROP 225 C _R	to drop 100 points from the beginning and 225 points from the end of the plotted portion of the slice.
> PIXRANGE -0.001 0.004 C _R	to set the range of the vertical axis to be -1 to 4 mJy/beam.
> TVSLICE C _R	to plot the slice in the TV window.

Note: several slices may be put on one TV plot. Use **TVASLICE** C_R for the additional ones. Multiple colors may be achieved by using different graphics channels (**GRCHAN**).

Slice files may be converted into plot files by:

> **GO** **SL2PL** C_R

The resulting plot files may then be output by:

> GO LWPLA C _R	to display the plot file on a PostScript printer.
> GO TKPL C _R	to display the plot file in the TEK window.
> GO TVPL C _R	to display the plot file on a TV graphics plane.

The task **SLFIT** fits Gaussian components to one-dimensional data in slice files. Assuming that the usual **GETNAME** step has been done, a typical session would go like:

> INEXT 'SL'; EXTL C _R	to list the parameters of the slice files.
> INVERS m C _R	to select the m th file for analysis.
> TVSLICE C _R	to plot the slice in the TV window.
> EDROP 840 ; BDROP 700 C _R	to select a subsection to fit.
> TVSLICE C _R	to re-plot just the subsection.
> ORDER 1 C _R	to fit a linear baseline, no baseline up to quadratic are allowed.
> NGAUSS 2 C _R	to fit 2 Gaussians.
> TVSET C _R	

This verb will prompt you to POSITION CURSOR AT CENTER & HEIGHT OF GAUSSIAN COMP 1. Move the cursor to the requested position and hit any button. Then you are asked to POSITION CURSOR AT HALFWIDTH OF GAUSSIAN COMP 1. Move the cursor to the half-intensity point of the component and click any button. Continue until all components have been entered. (Note: these operations are also available on the TEK device with verbs beginning with TK. We recommend the TV versions since cursor reading in X-Windows emulations of TEK devices appears to be unreliable.) Then type:

> **TVAGUESS** C_R to plot the guess on top of the slice plot.

If everything looks ok, then:

> **GO** **SLFIT** C_R to run the task.

When the task gets an answer, the solution will be displayed as *AIPS* messages, recorded in the message file, and recorded in the slice file itself. To get a hard copy of the results:

> **PRTASK 'SLFIT'** ; **PRTMSG C_R** to print the message file.

and, to display the results in the TV window, enter:

> **TVSLICE C_R** to re-plot the slice.

> **TVAMODEL C_R** to add the model results to the plot.

> **TVACOMPS C_R** to add the individual components of the model to the plot.

> **TVARESID C_R** to add the residuals (data – model) to the plot.

The TV plot may be easier to examine if you use different **GRCHAN** for the different types of plot. To get a higher quality plot of the results, an example of which is shown in § 6.3.2.1, type:

> **DORES TRUE** ; **DOMOD 3 C_R** to request the residuals, the model, and the model components.

> **DOSLICE FALSE C_R** to leave the slice data out of the plot.

> **TASK 'SL2PL'** ; **GO** ; **WAIT C_R** to make a plot file and wait for it to be complete.

7.5.5 Spectral parameter fitting (TVSPC, XGAUS, ZEMAN, RMFIT, FARS)

In 31DEC16, task **TVSPC** has appeared to assist users in getting to know their data cubes. It uses a 2-dimensional image plane of some parameter that is useful in visualizing the full field. A moment-zero image is one obvious choice. The user selects a pixel on this image using the TV display and a spectrum at that coordinate is displayed. The task is highly interactive allowing a quick choice of an interesting pixel. The spectrum, or a portion of the spectrum, at that point may be fit by up to 4 Gaussians plus baseline and may be saved as a SLice file. The corresponding spectrum from a second transposed cube may also be displayed, allowing for example the I and V or the Q and U polarizations to be displayed simultaneously. Optionally, a third spectral cube may be displayed with the spectral plane selected interactively from the spectral plots. See the associated *AIPS* Memo². In earlier releases, use **XPLOT** with **DOTV 1** to familiarize yourself with your image cube.

XGAUS is an interactive task which can fit up to eight Gaussians and a linear baseline to each row of an image. In 31DEC13, it was redesigned so that it begins by setting up a table of the peak image intensities and fit results, so that it may be re-started multiple times with the same table, and so that it provides multiple means to edit the results before it finally writes its results as a set of $n - 1$ dimensional image files. Although **XGAUS** was designed for use primarily on transposed spectral-line cubes (see § 8.5.2), it has a wide variety of other applications. The interaction is optional and uses the TV window on your workstation. The data, initial guess, model fit (total and components), and the residual for each row may be plotted on the TV screen. If the number of Gaussians being fit is larger than one, you may choose for each row to enter a revised initial guess using the cursor in the TV window. This process is similar to that of **TVSET** described above (§ 7.5.4). If things go well, you may turn off the interactive model setting; the task will turn it back on if there is a bad fit at some pixel. After all pixels have been fit, the task enters an image and edit mode. Images are computed for all Gaussian parameters and their uncertainties. A menu is shown offering displays of the images and several methods to revisit the solutions for selected pixels. This includes swapping the solutions between components and re-doing a list of individual pixels or all pixels for which the solutions or uncertainties exceed specified ranges. When you are satisfied with the result, you may instruct the task to write out images of the Gaussian parameters and their uncertainties. The use of **XGAUS** as well as **ZEMAN**, **RMFIT**, and the plotting and modeling tasks (below) is discussed in detail in *AIPS* Memo 118.³

In 31DEC13, **ZEMAN** was written to fit for Zeeman splitting. The basic function fit to the V Stokes spectrum

²Greisen, E. W. 2017, "Exploring Image Cubes in *AIPS*," AIPS Memo 120 revised, <http://www.aips.nrao.edu/aipsdoc.html>

³Greisen, E. W. 2017, "Modeling Spectral Cubes in *AIPS*," AIPS Memo 118 revised, <http://www.aips.nrao.edu/aipsdoc.html>

is a constant time the total intensity plus another constant times the derivative of the total intensity with frequency. The task is similar to `XGAUS` in that it first sets up a table of image intensities and fit results, then fits all pixels above specified intensities, and then offers the image display and edit modes. Unlike previous Zeeman programs (in other packages), `ZEMAN` also offers the option to use the results of `XGAUS` on the total intensity cube to fit the V Stokes parameter with a constant times the total intensity plus a separate constant for the derivative of each Gaussian component with frequency. This allows for multiple magnetic fields at the same pixel, each corresponding to a separate Gaussian spectral component.

In 31DEC17, tasks `AGAUS` and `ZAMAN` appeared to perform the same functions but on absorption-line spectral cubes. The fit is done to the observations, but the parameter that is Gaussian is the optical depth. These tasks and the essential mathematics are described in *AIPS Memo 122*.⁴

In 31DEC15, task `XG2PL` was written to plot the solution for individual spectra from `XGAUS`. The corresponding `ZEMAN` fit may also be plotted in a separate panel. Absorption line and optical depth models may also be plotted in 31DEC17.

Polarization spectra are often fit for rotation measure. The old task `RM` reads a transposed cube of the polarization angle made with multiple `COMBS`, then `FQUBE` or `MCUBE` followed by `TRANS`. It fits the rotation measure and the intrinsic magnetic field direction using one of two possible methods to resolve lobe ambiguities.

The task `FARS` reads two similar cubes of Q and U polarization images and performs a “rotation-measure synthesis.” The output of `FARS` is an image cube of the real part and another image cube of the imaginary part of the Fourier transform on the λ^2 axis, with or without one-dimensional Cleaning. Task `AFARS` reads these cubes to produce images of the maximum rotation measure and of the amplitude or phase at the maximum. Task `RFARS` applies the rotation measure image to the `FARS` input cubes to produce de-rotated cubes of Q and U. These may be `SQASH`’ed to make an image of the average un-rotated polarization. The `RUN` file/procedure `DOFARS` assists with the whole process, doing matrix transpositions, finding channel weights, running `FARS`, transposing the outputs, and cleaning up.

Rotation-measure synthesis remains an experimental procedure at present. To test the handling of spectra in Q and U, the task `TARS` was written. It reads a text file giving the spectrum of Q and U at some single coordinate and performs the `RM` synthesis using methods from `FARS` plus an additional experimental form of complex Clean. Up to 20 model components may be added to the Q/U spectral data at specified rotation measure, amplitude, and phase. Task `QUXTR` was written to extract a spectrum from a pair of Q and U polarization cubes for input to `TARS`. The text-file output of `TARS` may then be plotted with `TARPL`, which allows multiple `RM` spectra from the same `TARS` output file to be plotted on top of each other.

It has been found that rotation-measure synthesis is severely limited in its ability to separate multiple `RM` components. Therefore, another `XGAUS`-like task called `RMFIT` was written. It uses the `FARS` rotation-measure output cubes to provide initial guesses for fits to the Q and U polarization spectral cubes. Like `XGAUS`, it first builds a table of image intensities to hold the fit results and then displays the `FARS` spectrum at each pixel to take an initial guess with which to fit the Q and U spectra. They are displayed and, if the fit is accepted, the task goes on to the next pixel. `RMFIT` is restartable and has the same sort of image/edit stage in which images are displayed and the fits may be revisited by means similar to those in `XGAUS`. An option to fit a spectral index in Q and U is offered, but, in general, the input cubes should have been corrected by the total intensity spectral index and the channel-dependent primary beam before being used in `RMFIT`. An additional spectral index in Q and U may then indicate “thick” rotation measure screens rather than a true source spectral index. Other models of thickness may be fit instead in 31DEC14. The task may be re-started repeatedly and output images written only when desired. Despite rather simple fitting methods, `RMFIT` has shown the ability to separate closely spaced components not separable by rotation measure synthesis. In 31DEC15, task `RM2PL` was written to make plot files from the solutions for individual Q/U spectra.

⁴Greisen, E. W. 2017, “Modeling Absorption-line Cubes in *AIPS*,” AIPS Memo 122, <http://www.aips.nrao.edu/aipsdoc.html>

7.6 Image analysis

Image analysis is a very broad subject covering essentially all that *AIPS* does or would like to do plus specialized programs designed to analyze a user's particular image in the light of his favorite astrophysical theories. *AIPS* provides some general programs to perform geometric conversions, image filtering or enhancement, and model fitting and subtraction. These are the subjects of the following sections. Specialized programs for spectral-line, VLBI, and single-dish data reduction are described in Chapter 8, Chapter 9, and Chapter 10, respectively. Chapter 11 of *Synthesis Imaging in Radio Astronomy*⁵ covers the topic of image analysis in more detail.

7.6.1 Geometric conversions

The units of the geometry of an image are described in its header by the coordinate reference values, reference pixels, axis increments, axis dimensions, and axis types. The types of coordinates (celestial, galactic, etc.) and the type of tangent-plane projection (**SIN** from the **VLA**, **TAN** from optical telescopes, **ARC** from Schmidt telescopes, **NCP** from the **WSRT**) are specified in the *AIPS* headers by character strings. See § 6.1.2 and *AIPS* Memo No. 27 for details of these projections. A “geometric conversion” is an alteration of one or more of these geometry parameters while maintaining the correctness of both the header and the image data. The *AIPS* tasks which do this interpolate the data from the pixel positions in the input image to the desired pixel positions in the output image.

The simplest geometric conversion is a re-gridding of the data with new axis increments and dimensions with no change in the type of projection or coordinates. The task **LGEOM** performs this basic function and also allows rotation of the image. One use of this task is to obtain smoother displays by re-gridding a sub-image onto a finer grid. To rotate and blow up the inner portion of a 512^2 image, enter:

- > **TASK 'LGEOM'; INP C_R** to review the inputs.
- > **INDISK n ; GETN ctn C_R** to select the image.
- > **BLC 150 ; TRC 350 C_R** to select only the inner portion of the image area.
- > **IMSIZE 800 C_R** to get an 800^2 output image. This will allow the sub-image to be blown up by a factor of 3 and rotated without having the corners “falling” off the edges of the output image.
- > **APARM 0 C_R** to reset all parameters to defaults.
- > **APARM(3) = 30 C_R** to rotate the image 30° counterclockwise (East from North usually).
- > **APARM(4) = 3 C_R** to blow up the scale (axis increments) by a factor of 3.
- > **APARM(6) = 1 C_R** to use cubic polynomial interpolation.
- > **INP C_R** to check the inputs.
- > **GO C_R** to run the program.

LGEOM allows shifts of the image center, an additional scaling of the y axis relative to the x axis, and polynomial interpolations of up to 7th order. **OGEOM** is similar to **LGEOM**, but handles blanked pixels in a manner that does not increase the blanked area.

A much more general geometric transformation is performed by **OHGEO** and **HGEO**, which convert one image into the geometry of a second image. The type of projection, the axis increments, the rotation, and the coordinate reference values and locations of one image are converted to those of a second image. One of these tasks should be used before comparing images (with **COMB**, **KNTR**, **PCNTR**, **BLANK**, **TVBLINK**, etc.) made with different geometries, *i.e.*, radio and optical images in different types of projection or **VLA** images taken

⁵*Synthesis Imaging in Radio Astronomy*, A collection of Lectures from the Third NRAO Synthesis Imaging Summer School, eds. R. A. Perley, F. R. Schwab and A. H. Bridle, Astronomical Society of the Pacific Conference Series Volume 6 (1989)

with different phase reference positions. Use **EXPLAIN OHGEO** C_R to obtain the details and useful advice. **SKYVE** regrids images from the Digital Sky Survey (optical DSS) into coordinates recognized by *AIPS*.

A potentially very powerful transformation is performed by **PGEOM**. In its basic mode, it converts between rectangular and polar coordinates. An example of this operation is illustrated in Figure 7.1. However, **PGEOM** can also “de-project” elliptical objects to correct for their inclination and “unwrap” spiral objects. Type **EXPLAIN PGEOM** C_R for information.

7.6.2 Mathematical operations on a single image

The task **MATHS** allows the user to do a mathematical operation on a single image on a pixel by pixel basis. Currently supported mathematical operators are: **SIN**, **COS**, **TAN**, **ASIN**, **ACOS**, **ATAN**, **LOG**, **LOGN**, **ALOG**, **EXP**, **POLY**, **POWR**, and **MOD**. An example of **MATHS** follows, in which the output image (**OUT**) is computed in terms of the natural logarithm of the input image (**IN**) as follows: $OUT = 4 + 2 \times (\log(3 \times IN) - 1)$

- | | |
|------------------------------------|--|
| > TASK 'MATHS' ; INP C_R | to review the required inputs. |
| > INDI 0 ; MCAT C_R | to help you find the catalog number of the input image. |
| > INDI n ; GETN ctn C_R | to specify the image, on disk n catalog slot ctn as the input. |
| > OUTN 'xxxxx' C_R | to choose $xxxxx$ as the name for the output image. |
| > OUTC 'ccc' C_R | to choose ccc as the class for the output image. |
| > OPCODE 'LOGN' C_R | to specify the operation to be performed (a natural logarithm). |
| > CPARM 4, 2, 3, -1 C_R | to specify the coefficients. |
| > GO C_R | to start MATHS . |

Undefined output pixels (in the current example, all pixels in the input image ≤ 0) are either blanked (**CPARM(6) ≤ 0**) or put to zero (**CPARM(6) > 0**). Type **EXPLAIN MATHS** C_R for further information on the available operators and the meaning of **CPARM** for any particular operator.

7.6.3 Primary beam correction

PBCOR allows correction for the attenuation due to the shape of the primary beam. Its use is straightforward:

- | | |
|------------------------------------|--|
| > TASK 'PBCOR' ; INP C_R | to review the required inputs. |
| > INDI 0 ; MCAT C_R | to help you find the catalog number of the input image |
| > INDI n ; GETN ctn C_R | to select the input image from disk n catalog slot ctn . |
| > OUTN 'xxxxx' C_R | to specify $xxxxx$ for the name of the output image. |
| > OUTC 'ccc' C_R | to specify ccc for the class of the output image |
| > PBPARM 0 C_R | to use the VLA or ATCA beam parameters fit for the particular receiver. |
| > COORDIN 0 C_R | to use the pointing position from the image header. |
| > GO C_R | to start PBCOR . |

The default behavior requested above uses the position in the header as the pointing position and uses the empirically determined shape of the evla, **VLA** or ATCA primary beam; **PBCOR** will scale the primary beam shape according to the frequency provided in the image header and use the parameters associated with the particular antenna feed. These defaults can be overridden by specifying particular values of **COORDIN** and **PBPARM**. In 31DEC16, the **EVLA** parameters have been added. They are frequency dependent within each band, so **PBCOR** determines the beam value at the two tabulated frequencies nearest to the image frequency and then interpolates.

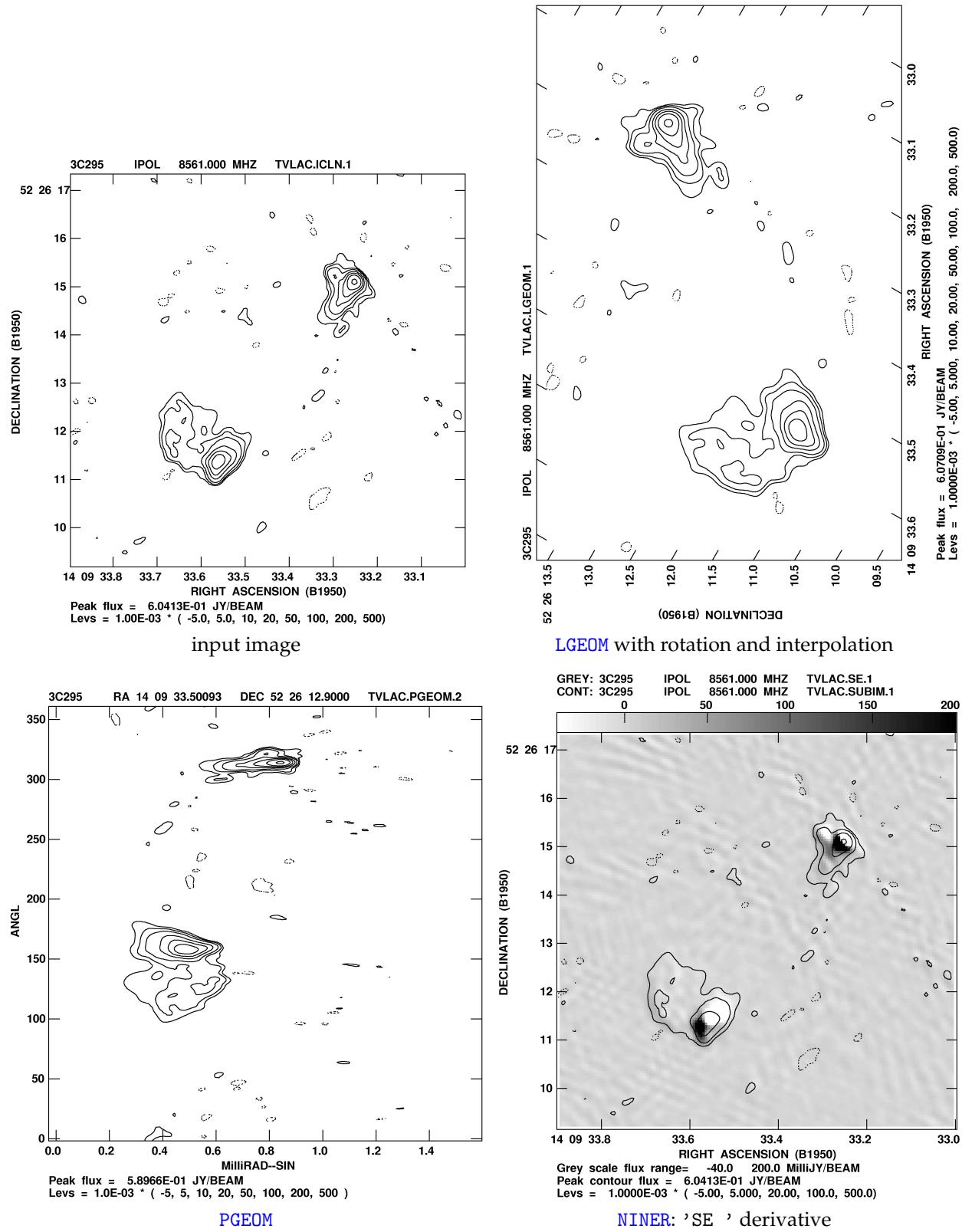


Figure 7.1: Geometric and other functions on an image.

In 31DEC12, the task **SPCOR** may be used to apply corrections both for primary beam and for spectral index. The latter are based on images of spectral index and spectral index curvature such as those used in **IMAGR** (§ 5.3.4.4). Such corrections may be significant in Faraday-rotation synthesis (§ 7.2, § 7.5.5).

An image of the primary beam may be generated with the task **PATGN** using **OPCODE** 'BEAM' C_R with other adverbs to give the antenna type, frequency, cell size, image size, and, optionally, the parameters of the beam shape. Beam parameters for the **EVLA**, old **VLA**, and the ATCA are known.

7.6.4 Changing the resolution of an image

CCRES allows you to change the resolution of a Cleaned image by removing any Clean components in the image and then restoring Clean components with your choice of resolution. The task may also be used to remove Clean components to create a residual image or to restore Clean components to an existing residual image. **CCRES** allows you to smooth or hyper-resolve your image. Unlike **RSTOR**, **CCRES** rescales the residual image to put it into units of Jy/beam for the new beam. This may be a superior way to image with a beam that does not replicate the central portion of the dirty beam. **IMAGR** leaves the residual image in units of Jy per dirty beam while restoring the Clean components in units of Jy per the Clean beam given in the header.

Note that **CCRES** only changes the Clean components, not the residual image resolution. Furthermore, **CCRES** does not take into account the varying resolutions of the many planes in an image cube. **CONVL** on the other hand, with **OPCODE** 'GAUS', will convolve both the Clean components and the residual to the requested resolution, taking into account the change in input resolution as a function of frequency.

7.6.5 Filtering

For our purposes here, we can define “filtering” as applying an operator to an image in order to enhance some aspects of the image. The operators can be linear or nonlinear and do, in general, destroy some of the information content of the output image. As a result, users should be cautious about summing fluxes or fitting models in filtered images. (Technically, these remarks can also be made about Clean and self-calibration.) However, filtered images may bring out important aspects of the data and often make excellent, if unfamiliar-looking, displays of particular aspects.

NINER produces an image by applying an operator to each cell of an image and its 8 nearest cells. The task offers three nonlinear operators which enhance edges (regions of high gradient in any direction). It also offers linear convolutions with a 3×3 kernel which can be provided by the user or chosen from a variety of built-in kernels. Among the latter are kernels to enhance point sources and kernels to measure gradients in any of 8 directions. The 'SOBL' edge-enhancement filter can bring out jets, wisps, and points in the data, while the gradient convolutions produce images which resemble a landscape viewed from above with illumination at some glancing angle (as when viewing the Moon). Both are very effective when displayed on the TV or by the **KNTR** / **LWPLA** combination (see Figure 7.1). Enter **EXPLAIN NINER** C_R for additional information.

MWFLT, at present, applies any one of six non-linear, low-pass filters to the input image. Each filter is applied in a user-specified window surrounding each input pixel. One of the operators is a “normalization” filter designed to reduce the dynamic range required for the image while bringing out weaker features. Two of the operators are a “min” and “max” within the window. When applied in succession, they produce a useful low-pass filtered image (Rudnick, L. 2002, PASP, 114, 427). Other operators produce, at each pixel, the weighted sum of the input and the median, the “alpha-trimmed” mean, or the alpha-trimmed mode of the data in the window surrounding the pixel. These filters can be turned into high-pass filters by subtracting the output of **MWFLT** from the input with **COMB**. Type **EXPLAIN MWFLT** C_R for further information.

Histogram equalization provides another form of non-linear filtering. **HISEQ** converts the intensities of the full input image to make an output image with a nearly flat histogram. This magnifies small differences in the heavily occupied parts of the histogram (usually noise) and diminishes large differences in the less occupied parts (often real signal). **TVHLD** is an interactive task that loads an image to the TV with histogram equalization and then allows the intensity range and method of computing the histogram to be modified. It can write out at the end an equilized image in arbitrary (non-physical) units. **AHIST** does an “adaptive” histogram equalization on each pixel using a rectangular window centered on that pixel. This will magnify small differences in a more local sense, bringing out structures in smooth areas of different brightness. **SHADW** generates a shadowed image as if a landscape having elevation proportional to image value were illuminated by the Sun at a user-controlled angle. Although these tasks magnify noise, they are likely to elucidate real structures in large areas of nearly constant brightness.

7.7 Modeling in the image and uv planes

Special considerations for analysis and modeling of spectral-line image cubes are discussed in § 8.6. Models of galaxy rotation are fit by **GAL** to images of the predominant velocity (first moment images usually), while the entire data cube may be fit by task **CUBIT**.

The addition of model data to an image or *uv* data set is often useful either to simplify later processing steps or to study processing steps using a “source” of known structure. For example, the removal of the response to an appropriate uniform disk from the *uv* data for a planet will leave Clean the task of deconvolving only the remaining fine-scale structure to which it is well suited. The removal of a few bright point sources of known position and strength may allow imaging with significant tapers in a numerically smaller field. The tasks **MODIM**, **MODSP**, **UVMOD**, **MODAB**, and **SPMOD** will add (or subtract) up to 9999 point, Gaussian, disk, rectangular, spherical, or exponential sources to the (scaled) input image or *uv* data, respectively. These tasks can also add noise and allow the original data to be replaced by the model. **UVMOD** can include a spectral index for the sources, **SPMOD** includes spectral lines for the sources, **MODIM** can include both a spectral index, a rotation measure, and even a rotation measure thickness for each component, and **MODSP** can include polarized spectral lines with spatially varying spectral structure. Type **EXPLAIN MODIM ; EXPLAIN UVMOD ; EXPLAIN SPMOD ; EXPLAIN MODAB ; EXPLAIN MODSP** **Q** for details or see AIPS Memos 118⁶ and 122⁷. For simple cases, the older task **IMMOD** may be easier to use than **MODIM**.

The task **CCMOD** will create a clean-components file representing the chosen Gaussian or disk model. Clean may then be “restarted” with the model as its initial set of components.

The task **UVFIT** may be useful for fitting Gaussian or uniform-sphere models to *uv* data sets with a maximum of 2.5 million visibilities. Up to 60 source components may be fit along with up to 30 antenna gains. Provision for getting initial guesses for the specified spectral channel from a text file with guesses for many channels is now available as is a compact text file output option for the solutions. Task **OMFIT** is a more complicated and difficult to use task which fits source components in a *uv* data set along with antennas gains. It allows multiple different model types and even a full self-calibration. The results from **OMFIT** are often well worth the extra effort to obtain them.

AIPS contains tasks which create simulated data. **PATGN** will make images of various patterns. **DTSIM** generates *uv* data following instructions given primarily in a keyin-format text file. You may specify the antennas in detail or as the VLBA or **VLA**. Source parameters and models, frequency values and structure, data calibration errors of numerous types, scan structure and sequence, and more may be specified. **UVCON** also generates *uv* data from “scratch” using text files of antenna data and Tsys and efficiency data as well as images of a source model. The two tasks differ in their approach, making both of considerable interest.

⁶Greisen, E. W. 2015, “Modeling Spectral Cubes in AIPS,” AIPS Memo 118 revised, <http://www.aips.nrao.edu/aipsdoc.html>

⁷Greisen, E. W. 2017, “Modeling Absorption-line Cubes in AIPS,” AIPS Memo 122, <http://www.aips.nrao.edu/aipsdoc.html>

7.7.1 Analysis in the uv plane

In 31DEC17, two new tasks have appeared to perform some analysis on *uv* data. **DFTIM** is similar to the plot task **DFTPL** in that it Fourier transforms visibility data to a particular image pixel (celestial coordinate). **DFTPL** makes a plot versus time for a single group of frequencies. **DFTIM** instead writes out an image at that coordinate with frequency and time as the axes. The frequencies and times in this “waterfall” image may be averaged as the image is created. The other new task, **ELFIT**, fits polynomials to table data as functions of elevation, zenith angle, hour angle, parallactic angle, or azimuth. It has been used, for example, to measure antenna spillover, plotting Psys versus zenith angle.

7.8 Additional recipes

7.8.1 Bananes rôties

1. Preheat oven to 375° F.
2. Place 6 (peeled) **bananas** in a baking dish.
3. Sprinkle bananas with juice of 1/2 **lemon**.
4. Pour 2 tablespoons melted **butter** and 2 tablespoons **dark rum** over the bananas. Sprinkle with 2 tablespoons **brown sugar**.
5. Place in oven for 10 minutes.
6. Pour on 2 more tablespoons **melted butter** and 2 more tablespoons **dark rum** and bake for 5 minutes more.
7. Serve at once, spooning some sauce over each banana.

7.8.2 Almond fudge banana cake

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

7.8.3 Coriander banana nut bread

1. Blend together in a large bowl $1\frac{2}{3}$ cups sifted all-purpose **flour**, $\frac{3}{4}$ cup **sugar**, 1 tablespoon **baking powder**, $\frac{1}{2}$ teaspoon **baking soda**, $\frac{1}{2}$ teaspoon **salt**, 2 teaspoons ground **coriander**.
2. Mix in 1 cup chopped unblanched **almonds** and set aside.
3. Melt $\frac{1}{3}$ cup **shortening** and set aside to cool.
4. Mix until well blended 1 large well-beaten **egg**, $\frac{1}{4}$ cup **buttermilk**, and 1 teaspoon **vanilla extract**.
5. Blend in $1\frac{1}{4}$ cups mashed ripe **bananas** and the shortening.
6. Make a well in center of dry ingredients and add banana mixture all at one time. Stir only enough to moisten dry ingredients.
7. Turn into greased $9 \times 5 \times 3$ -inch loaf pan and spread to corners.
8. Bake at 350° F about 1 hour or until a wooden pick comes out clean when inserted in center of bread. Immediately remove from pan and set on rack to cool.

7.8.4 Easy banana bread

1. Preheat oven to 350° F.
2. In a food processor cream $\frac{1}{2}$ cup soft **tofu**, $\frac{3}{4}$ cup **honey**, $\frac{1}{4}$ cup **sunflower or safflower oil**, 1 teaspoon **vanilla extract**, **egg substitute** for 1 egg, and 1 cup mashed ripe **banana**.
3. In a bowl combine 2 cups **whole wheat pastry flour**, $\frac{1}{2}$ teaspoon **baking powder**, and $\frac{1}{2}$ teaspoon **baking soda**.
4. Add to food processor along with a dash **salt** and process until creamy. Pulse in 1 tablespoon **poppy seeds**.
5. Pour into an oiled $9 \times 5 \times 3$ -inch loaf pan. Bake for 30 to 35 minutes, or until toothpick inserted in center of bread comes out clean. Cool on a wire rack for 30 minutes before removing from pan.

Thanks to Tim D. Culey, Baton Rouge, La. (tsculey@bigfoot.com).

7.8.5 Orange gingered bananas

1. Combine in a small saucepan $\frac{1}{4}$ cup **orange juice** and $\frac{1}{2}$ teaspoon **cornstarch**. Cook and stir over medium heat until boiling.
2. Add $\frac{1}{4}$ cup **orange juice**, $1\frac{1}{2}$ teaspoons **honey**, and $1\frac{1}{2}$ teaspoons chopped **crystallized ginger** and cook, stirring, until thoroughly heated.
3. Place 2 peeled, green-tipped **bananas** in a shallow baking dish and cover with sauce.
4. Bake at 350° deg about 15 minutes or until the bananas are tender (but not soft), basting with the sauce several times.

8 Spectral-Line Software

This chapter deals with the analysis and reduction of spectral-line data after they have had the basic calibrations, described in Chapter 4, applied. Spectral-line software generally involves three-dimensional images, often called “cubes”, in which one of the three axes is frequency or velocity. Special programs are available to build, manipulate, and transpose these cubes and to display them properly. Most of the continuum software will work on data cubes or on appropriate two-dimensional subsets from a data cube. Spectral-line *uv* data can be read into *AIPS* from “*uv*” FITS tapes. Often, however, the data already exist on disk having been calibrated first within *AIPS*.

The data reduction process at this point has a number of stages, namely data preparation and assessment, self-calibration, continuum subtraction, imaging, display and manipulation of data cubes, and analysis. This chapter will address each of these areas in varying, but modest, detail on the assumption that the reader is somewhat familiar with the contents of previous chapters in this *CookBook*. Some aspects of the art of spectral-line imaging are discussed in Chapters 17 and 18 of *Synthesis Imaging in Radio Astronomy*¹. A brief outline of the basic calibration process is given in Appendix A of this *CookBook* and an outline of spectral-line analysis and calibration is given in Appendix B.

8.1 Data preparation and assessment

In the following sections, it is assumed that the *uv* data are on disk and that the basic calibrations have been applied to them. If your data are not currently on disk, you will need to read them in from magnetic tape or FITS disk files. For tape, mount the tape in hardware and software (§ 3.9) and then position the tape to the desired data files:

- | | |
|---|---|
| > NFILES <i>n</i> ; AVFILE <i>C_R</i> | to advance <i>n</i> files. |
| > TPHEAD <i>C_R</i> | to check that the tape is correctly positioned. |

Use **FITLD** to read in the data:

- | | |
|---|---|
| > TASK 'FITLD' ; INP <i>C_R</i> | to review the inputs. |
| > OUTCL ' ' ; OUTN ' ' <i>C_R</i> | to use the file names recorded on tape (the default). |
| > NFILES 0 <i>C_R</i> | to read the current file on tape. |
| > NCOUNT 2 <i>C_R</i> | to load two files (if desired). |
| > GO <i>C_R</i> | to run FITLD . |

For FITS disk files, set **DATAIN** appropriately and then run **FITLD**. Examine the *uv* data set header with **IMHEAD** after **FITLD** finishes. It should show multiple pixels on the **FREQ** axis, like the *uv* header illustrated in § 3.3.4.

If your data are not yet calibrated, consult Chapter 4, especially § 4.7. When the CL, BP, and other calibration tables are complete, apply them to the line data with **SPLIT**:

- | | |
|--|-----------------------------------|
| > TASK 'SPLIT' <i>C_R</i> | |
| > SOURCE 'sou1', 'sou2', ... <i>C_R</i> | to select sources, ' ' means all. |
| > TIMERANG 0 <i>C_R</i> | to keep all times. |
| > BIF 1 ; EIF 0 <i>C_R</i> | to keep all IFs. |
| > BCHAN 1 ; ECHAN 0 <i>C_R</i> | to keep all spectral channels. |
| > FREQID 1 <i>C_R</i> | to set the one FQ value to use. |
| > DOCALIB TRUE <i>C_R</i> | to apply calibration. |

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

> GAINUSE 0 C _R	to use the highest numbered CL table.
> FLAGVER 1 C _R	to apply the flag table.
> DOPOL TRUE C _R	to correct for feed polarization.
> DOBAND 1 C _R	to correct bandpass.
> BPVER 1 C _R	to select BP table to apply.
> STOKES '' C _R	to write the input Stokes type.
> DOUVCOMP TRUE C _R	to write visibilities in compressed format.
> APARM 0 C _R	to clear VLBA options, including the one to calibrate the data weights.
> INP C _R	to review the inputs.
> GO C _R	to run the program when inputs set correctly.

The *uv* data produced by this process should have applied to them the full calibration and editing determined from the calibration sources. Whether these are adequate or will need to be enhanced from the source data themselves is now to be determined.

Your data may need to be edited further at this point. If your calibrator data were almost free of problems, save for simple matters like dead antennas, then your source data are not likely to need much editing either. If your calibrator data required detailed and erratic data flagging, then your source data will also need attention. All of the considerations of § 4.4 apply here too. In either case, the first step is to check for narrow-band interference which is a fairly common problem at some wavelengths. The best task for this is **SPFLG** (§ 10.2.2), a spectral-line version of **TVFLG** (§ 4.4.3). **SPFLG** displays spectral channels for all IFs along the horizontal axis and time along the vertical axis, one baseline at a time. This is tedious to use for editing all of the data from a large interferometer. Instead, select a few of the shorter baselines (with the **ANTENNAS** and **BASELINE** adverbs) and do a quick check for interference. If your data has sufficient signal-to-noise, then this quick examination should let you check which channels have signal and whether the Galaxy, for example, has contributed unwanted signal to your data. If you find significant terrestrial interference, you may flag the bad data with **SPFLG** (consider using the “ALL-BL” option to avoid looking at hundreds of baselines) or with **UVFLG** (entering “by hand” what you find with **SPFLG**). Tasks **UVLIN** and **UVLSF** (below and § 8.3) both offer the option to flag all channels of a particular sample if any one channel in the continuum-fitting portion of the spectrum has excessive flux after the fit. **UVLSF** also offers the option to flag when the rms of the continuum fit is excessive. The task **CLIP** (§ 5.5.1) works in a channel-dependent way on line data. It can be instructed to flag a full spectrum if more than a user-specified number of channels are flagged in that spectrum. **FLGIT** is designed to work on line data sets that have a great deal of interference, generating channel-dependent flagging in the output file or in a flag table. **UVMLN** may also be used to generate flag tables for multi- and single-source files to flag seriously deviant data. **FLAGR** uses multi-channel data to estimate rms and flags any times having excessive rms, amplitude, weight, etc. **RFLAG** will flag data in detailed ways based on excessive variability in time and/or frequency. **REFLG** will compress the large flag table made by **RFLAG**. **UVCOP** will apply even a very large flag table if so instructed. These tasks will eliminate certain classes of problems without bothering you with the details, which may — or may not — be a good thing.

To do serious editing of a non-spectral nature and to do self-calibration, you will need to create a single-channel data set that represents your multi-channel data in some fashion. If your data contains little continuum signal, but does have a strong spectral-line signal, then use **UVCOP** to copy the data for the channel with the strongest signal to a single-channel data set. Alternatively, the task **AVSPC** may be used to average a few adjacent channels together to create the single-channel data. If, on the other hand, your data set contains a useful level of continuum emission — and most do — then your single channel should be some estimate of this continuum. If your line signal is weak and you have done no frequency-dependent editing, then you can use the so-called “channel-0” data set created by **FILM** when you first loaded your **VLA** data to disk. This data set is the vector average of the center 3/4 of the selected spectral channels, but remains useful only if you have applied all of the same calibration, editing and **SPLITTING** to it that you applied to the line data. It may be safest to create a new continuum estimate from the line data in any case.

The classical method for doing this is the task [AVSPC](#):

- > [TASK 'AVSPC'](#) ; [INP C_R](#) to review the inputs needed.
- > [IND m](#) ; [GETN n C_R](#) to specify the input *uv* file.
- > [CLRONAME C_R](#) to use the default output names and disk.
- > [AVOPT '' C_R](#) to average channels only within IFs.
- > [ICHANSEL b₁ , e₁ , i₁ , if₁ , b₂ , e₂ , i₂ , if₂ , ...](#) to average every *i*₁ channel from *b*₁ through *e*₁ in IF *if*₁, plus every *i*₂ channel from *b*₂ through *e*₂ in IF *if*₂, etc.
- > [ICHANSEL 10 , 50 , 1, 0, 81 , 119 , 1 C_R](#) e.g., average channels 10 through 50 and 81 through 119 in each IF, omitting the band edges and the channels with line signal.
- > [INP C_R](#) to review the inputs.
- > [GO C_R](#) to run the program when you're satisfied with inputs.

A non-classical method for estimating the continuum is provided by the task [UVLSF](#). It fits a linear “spectral baseline” in the real and imaginary parts of each visibility spectrum. By adding a slope to the channel-independent bias found in [AVSPC](#), it allows for small changes in the continuum visibility with frequency and spatial resolution across the observed band, as well as small, time-variable changes in the spectral passband not fully removed by [BPASS](#). The option to fit even higher order baselines was added. These are seldom useful and can run into problems since they are not constrained by data in the channel ranges containing the line signals. The principal output of [UVLSF](#) is a multi-channel data set with this baseline subtracted, *i.e.*, a continuum-free line data set. However, the user may also get a single-channel “continuum” data set evaluated at any one channel from the fit baselines. Thus:

- > [TASK 'UVLSF'](#) ; [INP C_R](#) to review the inputs needed.
- > [IND m](#) ; [GETN n C_R](#) to specify the input *uv* file.
- > [CLRONAME C_R](#) to use the default output names and disk.
- > [BCHAN 0](#) ; [ECHAN 0 C_R](#) to include all channels.
- > [ICHANSEL b₁ , e₁ , i₁ , if₁ , b₂ , e₂ , i₂ , if₂ , ...](#) to average every *i*₁ channel from *b*₁ through *e*₁ in IF *if*₁, plus every *i*₂ channel from *b*₂ through *e*₂ in IF *if*₂, etc.
- > [DOOUTPUT TRUE C_R](#) to get the continuum data set.
- > [CHANNEL 0 C_R](#) to select the channel at which the baseline is evaluated — in this case, the default which is the reference channel. Choose one near the center of the band.
- > [INP C_R](#) to review the inputs.
- > [GO C_R](#) to run the program when you're satisfied with inputs.

The continuum data set will have AIPS class [BASFIT](#) and be on the same disk as the output line data set. You may also use [UVLSF](#) to flag data with excessive residual flux and rms in the channels used for the fitting. You may apply the usual data selection and calibration and flag tables to the input data, meaning that [SPLIT](#) need not have been run on the data previously. Note that neither [AVSPC](#) nor [UVLSF](#) should be used if the continuum visibility changes substantially across the passband, due to large bandwidth and complexity in the continuum source. If the phase of the continuum changes substantially across the passband due to the centroid of the continuum being considerably separated from the pointing center, you should instruct [UVLSF](#) to shift the phase reference position before fitting the linear function and then to shift the phases back afterwards.

Your spectral-line data may need further modification and preparation. If you did not observe with “Doppler tracking” (and even the [VLA](#) changes frequency only on a scan-by-scan basis), then your data channels are not at the same velocity through the data set. Task [CVEL](#) may be used to convert the data to full Doppler tracked form, or simply to shift the center velocity of your reference channel to, for example, the velocity used in another data set. [CVEL](#) works on multi-source as well as single-source data sets. It applies any flagging and bandpass calibration to the data before shifting the velocity (which it does by a carefully

correct Fourier transform method). Note, the use of Fourier-transforms means that one *must not* use [CVEL](#) an data with channel separations comparable to the widths of some of the spectral features.

Three other tasks allow you to change the number of spectral channels in your data set. The simplest, [AVSPC](#), allows you to average every n adjacent channels together, producing a data set one n^{th} as large. The wsecond, called [SPECR](#), uses Fourier transform methods to increase or decrease the number of channels with a corresponding change in channel spacing. In 31DEC14, [SPECR](#) can also use simple interpolation methods in case the Fourier methods produce undesirable results (such as ringing when a feature is very narrow in the input data set). The third task, new in 31DEC15 is called [UVFRE](#) and it attempts to convolve the data from one frequency structure onto another frequency structure defined in a second data set (the uv equivalent of [HGEOM](#)). These tasks, along with [CVEL](#) may be required to convert one data set to the same channel velocity and separation as another data set, before the two can be concatenated (task [DBCON](#)) and used together in imaging. If simple frequency smoothing is needed, the calibration routines will apply various convolutions to the spectrum under control of the adverb [SMOOTH](#). [IMAGR](#), allows you to “average” channels together by gridding them into the correct (channel-dependent) locations in the uv plane. This is seldom important to narrow-band observations, but is critical in wide “bandwidth synthesis” observations. Nonetheless, it allows you to keep full spectral resolution in the uv data set, but abandon it for some or all of the imaging.

8.2 Editing and self-calibration

Self-calibration can improve the “dynamic range” of your images significantly but **only** if there is sufficiently good signal-to-noise in the uv database. It works by comparing the input uv data with the predicted visibilities from a model of the source; from this a set of complex gain (amplitude, phase) corrections are generated for each antenna in the array as a function of time. Before engaging in a potentially long and useless exercise, it is wise to look at the continuum or single-channel data set you have created to determine whether there is sufficient signal (with respect to the noise) to enable detailed editing and/or self-calibration. Even if the continuum data would profit from these things, you must also decide whether there is sufficient signal-to-noise in the line data to benefit from the improved editing and calibration. Consult the sections on editing (§ 4.3.1 and § 4.4) and on self-calibration (§ 5.4) before deciding to continue with this section.

In fact, there is very little to be written here. The editing and self-calibration of the continuum or single-channel data set are precisely those described in Chapter 4 and Chapter 5. The goal is to create and fill an FG table (if flagging is needed) and an SN table (if self-calibration is needed) attached to the single-source, single-channel data set. Note that, for single-source data, we use an SN table containing the accumulated calibration while we use, for multi-source files, a number of SN tables with incremental calibrations and a single CL table with the net corrections. When all editing and self-calibration are done, you use [TACOP](#) (or [TAPPE](#) if the flag table must be appended to an existing flag table) to copy the two tables to the multi-channel data set and then use [SPLIT](#) to apply both tables to the multi-channel data (and to the single-channel data set too if desired).

Unfortunately, this nice scheme does not work all the time — mostly due to various programs not being able to apply the tables to the data. Use [SPLIT](#) to get around the problem. The preferred imaging and deconvolution task is [IMAGR](#) which understands and applies such tables.

In general, you may keep the flagging information in a flag table (specified with adverb [FLAGVER](#)) and add to and apply the table whenever it is needed. Be aware that flag tables are applied only by those tasks that have the [FLAGVER](#) adverb, but a number of tasks have been upgraded to understand calibration and flagging tables. In fact, data that are not in time order may have flag tables applied (if they are small enough) and time-independent calibrations (e.g., [DOBAND](#)=1 and [DOPOL](#) > 0) may also be done.

8.3 Continuum subtraction

Most spectral-line observations contain a certain amount of frequency-independent, continuum radiation in addition to the frequency-dependent line signals. In most (all?) cases, it is probably best to separate the two signals at this stage of the data processing. In this way, a *single* continuum image can be constructed to apply to all frequencies. It will probably be necessary to apply Clean or other expensive deconvolution techniques to determine the best estimate of this continuum image. To do this for each channel individually would not only be considerably more time consuming, but would result in different models for the continuum in each channel. Since Clean is a data-adaptive non-linear algorithm, minor disturbances in its progress causes it to converge to surprisingly different solutions. Such “disturbances” could be caused by differing noise and line signals in different channels. (It can be caused simply by differing computational order on identical data using different computers.) Clean also increases the noise in an image in the areas at which there are sources. If the continuum is removed from the line data before Cleaning, then this increase in noise is also removed. To determine the continuum from separately Cleaned channel images is to use noisier individual determinations to get the final estimate. If the imaging and deconvolution were a linear process, then this would not matter. But the deconvolution is non-linear, making it better to start with the best possible estimate of the continuum.

In many cases, the continuum signal is stronger by far than the line signal. In such cases, it is best to use as many channels as possible to estimate the continuum and to remove that estimate as early as possible. The remaining line signal may be of fairly low signal-to-noise ratio and, therefore, not need the same processing that the continuum signal requires. In particular, it may not be necessary to Clean the channel images particularly deeply, if at all.

All of the arguments above suggest that the continuum should be estimated and subtracted in the visibility domain. This has the unfortunate attribute that we can use only those channels which are free of line signal over the entire field of view. (This is because the Fourier transform relation mixes signals from all directions in the field into every visibility sample.) In the image domain, the dirty images also have this unfortunate attribute, in this case because the dirty beam mixes signals from all parts of the field into every pixel. The above arguments suggest that the only time one should determine the continuum in the image domain is when the observations have essentially no channels which are free of line signal over the full field of view. In that case, the continuum at each position will have to be determined from deconvolved images using those channels which are free of line signal at that position.

AIPS provides three separate tasks for fitting and subtracting the continuum in the *uv* plane. All three fit a linear “baseline” to a selected group of channels and subtract that from the data. The reasons for a linear baseline are (1) a continuum source offset from the field center will produce a linear phase slope across the passband, (2) minor other changes in the visibility with baseline may be approximately linear over narrow passbands, and (3) minor passband variations with time appear also to be approximately linear. Higher-order fits can be done by [UVLSF](#), but they do not seem to be justified and are often not well constrained. The first reason suggests that one should do the fit in amplitude and phase, an algorithm implemented by the task [UVBAS](#). This should be used *only* if you have good signal-to-noise in *all* of your visibility samples. The reason for this very stringent requirement is that visibility amplitudes have Ricean rather than Gaussian noise statistics. As a result, very biased estimates are produced in moderate to low signal-to-noise cases. For these, the two tasks [UVLSF](#) and [UWLIN](#) should be used. They do the fits in the real and imaginary parts of the visibilities and, hence, do not produce biased estimates in the absence of signal. Both tasks allow you to shift the visibilities to move a strong source to the center of the field before doing the fits (and then shift the visibilities back to the original phase center). This changes sinusoidal real and imaginary parts to linear for an accurate fit, but only for those cases where a single source dominates the field. Both tasks offer the option to flag discrepant data, but they differ in the details of how they do this. [UVLSF](#) offers the option to write out a “best-fit” single-channel data set as well as subtracting it from the line data; see § 8.1 for an sample inputs to [UVLSF](#).

There are also a number of ways to remove the continuum in the image domain, all of which assume that the channels have been imaged similarly and placed into a three- or more dimensional “cube.” The “classical” method to subtract the continuum, which is the least useful, is to average those image planes which contain no line signals using **SQASH** on each set of line-free planes and **COMB** to average those averages. Finally, **COMB** is used to subtract the resulting plane from each plane of the initial cube. In this method, the cube is in the “natural” transposition with the frequency axis third. The other two methods require you to use the task **TRANS** (see § 8.5.2) to transpose the frequency axis to the first (most-rapidly varying) axis. Then **IMLIN** may be used to fit polynomial baselines (linear is usually all that is justified) to the line-free parts of each spectrum. This task, like all of the other tasks so far mentioned in this section, requires you to use a fixed set of channels for all positions in the field. Many objects (*e.g.*, rotating galaxies) have spatially-dependent spectra in which each pixel has a rather narrow line width compared to the object or objects as a whole. In such cases, the task **XBASL** may be used. This task requires some endurance on your part to complete, but it allows you to specify the baseline region for each pixel interactively using the *AIPS* TV or TEK graphics windows.

8.4 Imaging

There are several tasks which may be used to convolve spectral-line data to a rectangular grid and then Fourier transform and Clean that grid to make an image. The old tasks, which we no longer recommend, are called **UVMAP**, **APCLN**, and **WFCLN** and may be used if you insist. Today, **IMAGR** should be used for all normal imaging and Cleaning. Read § 5.2 for basic information about this task and § 5.3 for information about using it to deconvolve your images. It can read either single- or multi-source data, can apply calibration, and writes all channels into an image cube. **IMAGR** uses much more flexible and correct methods for data weighting (§ 5.2.3) and uses the superior multi-field method of Cleaning in which source components are subtracted from the visibility data and the images recomputed in every “major cycle.” **IMAGR** grids each channel in exactly the proper place in the *uv* plane and can be used to do frequency smoothing (“averaging”) in this most correct of ways. It also offers options to correct your data for various effects of importance over wide fields and bandwidths. It even has experimental variations on Clean (§ 5.3.4) to deal with extended sources and the Clean bias.

To produce a cube of spectral line images with **IMAGR** from channels *n1* through *n2* use:

- | | |
|---|--|
| > TASK 'IMAGR'; INP C_R | to review the inputs needed. |
| > IND <i>m</i> ; GETN <i>n</i> C _R | to specify the input <i>uv</i> file. |
| > CLRONAME C _R | to use the default output names and disk. |
| > BCHAN <i>n1</i> ; ECHAN <i>n2</i> C _R | to include a range of channels. |
| > STOKES 'I' C _R | to make total intensity (unpolarized) images. |
| > NCHAV 1 C _R | to avoid averaging channels. |
| > CHINC 1 C _R | to do every channel, setting the channel increment. |
| > CHANNEL 0 C _R | do not restart. |
| > CELLSIZ Δx , Δy C _R | to set the image cell dimensions in arc seconds. Cells do not have to be square. |
| > IMSIZE <i>N_x</i> , <i>N_y</i> C _R | to set the image size in pixels (must be powers of two). |
| > NITER 0 C _R | to do no Cleaning. |
| > GO C _R | to run IMAGR . |

This makes “dirty” images and beams of the specified channels. If Cleaning is needed, set the Clean adverbs **NITER**, **NBOXES**, **CLBOX**, **GAIN**, **BOXFILE**, etc. Note that you must image one IF at a time in order to avoid combining multiple frequencies. The task **NOIFS** might be useful depending on your data.

Because of the upgrade of the **VLA** in 2010, **IMAGR** is more careful with image units. Unless you force

the Clean beam size with **BMAJ**, **BMIN** and **BPA**, **IMAGR** will image each channel at its “natural” (frequency-dependent) resolution. It now carefully scales each image plane so that it is actually in units of Jy per the Clean beam listed in the header. The Clean beam for each frequency is recorded in a CG table and used by **MCUBE** and **CONVL** among other tasks.

At this point, it is a very good idea to determine the noise in your output images and to compare it to the theoretical noise you expect. If your images are significantly noisier than expected, it is a very good idea to stop processing, to think about what may have gone wrong, and then to check and correct that. The noise may be determined using task **IMEAN**:

- | | |
|--|---|
| > TASK 'IMEAN' ; INP C_R | to review the inputs needed. |
| > IND m ; GETN n C_R | to specify the input image file. |
| > DOHIST TRUE C_R | to plot the histogram. |
| > NBOXES 200 C_R | to use a significant number of boxes in the plot. |
| > PIXRANGE = -x , x C_R | to limit the histogram to the range x , where x should be about 5 times the expected noise. |
| > DOTV TRUE C_R | to put the plot on the TV rather than in a file. |
| > BLC x1, y1, f1 ; TRC x2, y2, f2 C_R | to select a sub-image of the cube that is free of signal. |
| > GO C_R | to run the task, plotting on the TV. |

This will print two rms's on your message screen one computed using all the data and one done by fitting a histogram to the noise portion of the signal. Especially if the latter fails, the plot will allow you to estimate the true rms, ignoring those pixels significantly above and below the Gaussian noise part of the histogram. Using this plot, it may be better to include all of the data (**BLC 0 ; TRC 0 C_R**) rather than to limit the number of pixels contributing to the histogram. The AIPS verb **IMSTAT** does the all-pixel rms computation without the useful plot, while the verb **TVSTAT** allows you to mark one or more non-rectangular regions on an image on the TV over which the rms is computed. Task **RSPEC** will plot robust rmses computed on a per plane basis.

8.5 Display and manipulation of data cubes

8.5.1 Building and dismantling data cubes

Many spectral-line display and analysis functions make sense only on 3-dimensional (or more) images. In particular, image creation functions may create one image per spectral channel while spectral analysis requires the images to be ordered with the spectral axis first (most rapidly varying). To combine a number of n -dimensional images into an n - or $n + 1$ -dimensional image, use the task **MCUBE**. This task requires the images to have some one physical axis whose value varies between the input images in a manner consistent with a regular axis and it is that axis which is extended to make the “cube.” If the output image specified already exists, then the input maps are inserted in the output image in the appropriate places. This allows the n - or $n + 1$ -dimensional output image to be built up a bit at a time and allows replacement of portions of the image with a corrected n -dimensional image. Type **EXPLAIN MCUBE C_R** to receive a variety of hints and suggestions for using this rather general program. As an example, to put 31 frequency channel images into one cube, type:

- | | |
|---|---|
| > TASK 'MCUBE' ; INP C_R | to review the inputs. |
| > INNA 'N315' C_R | to select the source. |
| > INCL 'IIM001' C_R | to select the input image class for <i>all</i> images. |
| > INSEQ 1 ; IN2SE 31 ; IN3SE 1 C_R | to set the first and last image and the step in the task's loop over input sequence number. The task handles any missing images gently. |

> AXREF 1 ; AX2REF 31 C_R	to set the pixel coordinate of the first and last image on the third axis in the cube.
> NPOINTS 31 C_R	to set the total number of points on the third axis.
> OUTN ' ' C_R	to use the default OUTNAME .
> OUTCL 'LMFCUB' C_R	to specify, in OUTCLASS , the order of axes.
> GO C_R	to run MCUBE .

MCUBE will scale each image plane so that it is on the same brightness scale as represented by the Clean beam in the header.

Later on, you might want to replace some of these images by Cleaned images. *E.g.*, assume that you have Cleaned channels 10 through 20, one at a time. These images got the class ICL001. It is a good idea to give them sequence numbers that are the same as the channel numbers, thus 10 to 20. Putting them in the existing cube (make a backup copy with **SUBIM** if you are nervous) can then be done as follows:

> TASK 'MCUBE' ; INP C_R	to review the inputs.
> INNA 'N315' ; INCLA 'ICL001' C_R	to select the clean images.
> INSE 10 ; IN2SE 20 ; IN3SE 1 C_R	to set the first and last image and the step in the loop.
> OUTN 'N315' ; OUTCL 'LMFCUB' C_R	to select the existing cube.
> OUTSE 1 C_R	
> GO C_R	to run MCUBE .

IMAGR with its **DOTV** option allows you to Clean each channel with its own numbers of iterations and its own interactively set Cleaning boxes. There are new options to create boxes automatically on each image plane and an option to control whether the boxes for channel *n* are carried over to channel *n* + 1.

MCUBE can build a cube out of a set of images that do not have a suitable axis for image building. This can be a set of images that have all the same axis parameters (*e.g.*, identical images except for date) or they can be images at, for example, an arbitrary set of frequencies. Set **DOCONCAT** = 2 to force **MCUBE** to make a **SEQ.NUM**. axis or, if there are different frequencies, an **FQID** axis with accompanying **FQ** table. The original coordinate values on an axis that differs from image to image are recorded in the history file and, for frequencies, also in the **FQ** table. Task **FQUBE** does this operation more naturally and can combine two cubes each of which already has an **FQID** axis.

Most programs work on cubes. However, you may find it convenient, on occasion, to work with single image planes. To separate the channels from the cube, use task **SUBIM** typing, for example:

> TASK 'SUBIM' ; INP C_R	to review the inputs.
> INNA 'N315' ; INCL 'LMFCUB' ; INSE 1 C_R	to select the cube.
> OUTN 'N315' ; OUTCL 'IMAGE' C_R	to give it an output name and class.
> BLC 0 ; TRC 0 C_R	to select full planes.
> DOWAIT TRUE C_R	to run in synchronous mode.
> FOR J=10:20 ; BLC (3)=J ; TRC (3)=J ; PRIN J ; OUTSE J ; GO ; END	

This runs the program 11 times, taking planes 10 through 20 and creating separate images for them. To put the images back into the cube after they have been modified, use **MCUBE**.

8.5.2 Transposing the cube

The task [TRANS](#) will transpose the cube, ordering the axes in any way you specify. Typically, one transposes images of spectral channels into spectra at image pixels for both display and analysis purposes. Thus:

- > [TASK 'TRANS'](#) ; [INP C_R](#) to review the inputs.
- > [INCLASS 'LMFCUB'](#) C_R to select the untransposed cube.
- > [TRANSC '-312'](#) C_R to make new axis order -3, 1, 2 in terms of the old axis order (e.g., RA, Dec, Freq becomes Vel (opposite sign of Freq), RA, Dec).
- > [OUTCL 'VLMCUB'](#) C_R to give it an outclass reflecting the axis order.
- > [BLC 0 ; TRC 0](#) C_R to transpose the whole cube.
- > [GO C_R](#) to run the program.

Numerous tasks such as [IMLIN](#), [XSUM](#), [PLCUB](#), and [XMOM](#) act on the first axis and, typically, make sense only when that axis is frequency/velocity. Note that such *x*-axis analysis may also be useful in other cases, including angular and time *x* coordinates.

8.5.3 Modifying the image header

On occasion you may feel the need to modify or add to the information in the image header. For example, to add an alternate velocity description for the frequency axis of a cube, type:

- > [INDISK n ; GETN ctn](#) C_R to select the image.
- > [AXTYPE 'OPTHEL'](#) C_R to specify optical-convention velocities relative to the Sun.
- > [AXREF 16](#) C_R to specify the velocity reference pixel (channel 16 is the center of the band in our example).
- > [AXVAL 5.E6](#) C_R the velocity at the reference pixel in m/s.
- > [RESTF 1420.4E6, 5752](#) C_R to specify the line rest frequency in Hz (1420405752 Hz).
- > [ALTDEF](#) C_R to add the information to the header.
- > [ALTSW](#) C_R to switch between frequency and velocity information in axis labeling, [IMHEADER](#), etc.

The [VLA](#) now provides the velocity information to [FILLM](#), making this operation unnecessary in many cases.

Observers may find the Galactic coordinates of their sources to be of interest. To switch the header between Celestial and Galactic coordinates, type:

- > [CELGAL](#) C_R to go to galactic coordinates.
- > [CELCST](#) C_R to go back to celestial coordinates.

You may change many of the parameters in the image header and may fetch for use by the [POPS](#) language all header parameters. The verb [PUTHEAD](#) changes a header parameter and [GETHEAD](#) gets that parameter. Type [EXPLAIN GETHEAD](#) C_R for the details, including a list of recognized keywords, and [EXPLAIN PUTHEAD](#) C_R for the more restricted list of values which you may change in the header. For example, to move a header parameter from one image to another:

- > [KEYWORD 'keyword'](#) C_R to specify one of the fully allowed keywords.
- > [GETN n ; GETHEAD](#) C_R to get the header value from image *n*.
- > [GETN m ; PUTHEAD](#) C_R to put it in the header of image *m*.

8.5.4 Displaying the cube

The easiest way to look at the cube is by using **TVMOVIE**. This verb loads portions of planes of a cube into the TV memory, and displays them in sequence at a variable frame rate. Type:

- > **INDISK** *n* ; **GETN** *ctn* C_R to select the cube.
- > **INP** **TVMOVIE** C_R to review the inputs; use defaults for a start.
- > **TVMOVIE** C_R to load the images and start the movie.

Now follow the instructions on your screen on how to change the transfer function, change speed, look at single frames, etc. Having terminated the movie (button D), you may restart it with

- > **REMOVIE** C_R to resume the movie.

With the default adverbs, **TVMOVIE** will zoom the display so that a single frame fills the display area. If you are using a workstation, this may seriously limit the maximum possible frame rate. To use a smaller zoom factor and smaller display area, enter:

- > **DOCENTER** **TRUE** C_R to enable the option.
- > **FACTOR** *m* C_R to use a zoom factor of *m*.
- > **REMOVIE** C_R to rerun the movie.

Note that you may set these adverbs before the initial **TVMOVIE** command as well. If your cube has more than 3 axes with multiple pixels, **TVMOVIE** can display all planes if **DOALL** = 1.

TVMOVIE uses an image ordering on the screen designed to improve the performance of movies on “real” TV devices. To use a more pleasing ordering (left to right, top to bottom) that is nearly as efficient on workstations, use the **TVCUBE** verb instead. The verb **REMOVIE** works on these displays as well. One of the reasons to use **TVCUBE** is to capture a full TV plane’s worth of channels and print them on a PostScript printer. The first step is to get **TVCUBE** to put the desired channels in one TV image plane. Set **TBLC(3)** and **TRRC(3)** to select those planes you want and set **TVCHAN** = 2 (or the highest TV plane allowed on your system) to force all the images into one TV image. Stop the move with button D, display the full TV memory (use the F2 button), adjust the enhancement (and color) of the display, and then run task **TVCPS**. If your images are too large for one channel, use **TVCHAN** = 1 and run **TVCPS** once for each channel.

Another way to see multiple image planes on a single page is with the task **KNTR**, which plots a page of contour and/or grey-scale drawings for each plane. This task allows you to include “star” positions and even an outline of the clean beam. A **KNTR** plot of neutral hydrogen emission from NGC6503 is shown in Figure 8.1.

It can be very informative to transpose the cube to look at position-velocity planes instead of position-position planes. A **TVCUBE** display of the NGC6503 data cube transposed to 132 (position-velocity-position) order and then captured with **TVCPS** is shown in Figure 8.2.

Finally, it is useful to look at the cube as a set of spectra. Use **TRANS** to transpose to 312 order (or -312, 321, and the like). The task **XPLOT** displays the spectra on the graphics display selecting only those with a sufficiently strong line signal (set by adverb **FLUX**) and ignoring those with really strong fluxes (controlled with **PIXVAL**). This can get rather tedious and it is easy to get lost in your cube. The task **PLCUB**, illustrated in Figure 8.3, allows you to see numerous spectra at one time plotted in correct orientation with respect to their position coordinate. It is necessary to experiment with the adverbs for **PLCUB**, especially **YINC**, **ZINC**, **PIXRANGE**, and the like, to get a good plot. Use the TV with **DOTV** **TRUE** C_R to save trees.

In general, other display programs work on one plane at a time. Therefore, you must specify which plane in the cube you want to see. For example, if you want to do **TVALL** on channel 16, which is on pixel 16 of the third axis, you type:

- > **TBLC** 0 0 16 ; **TVALL** C_R

Note that adverbs **TBLC** and **TRRC** are used for TV displays while **BLC** and **TRC** are used for image operations.

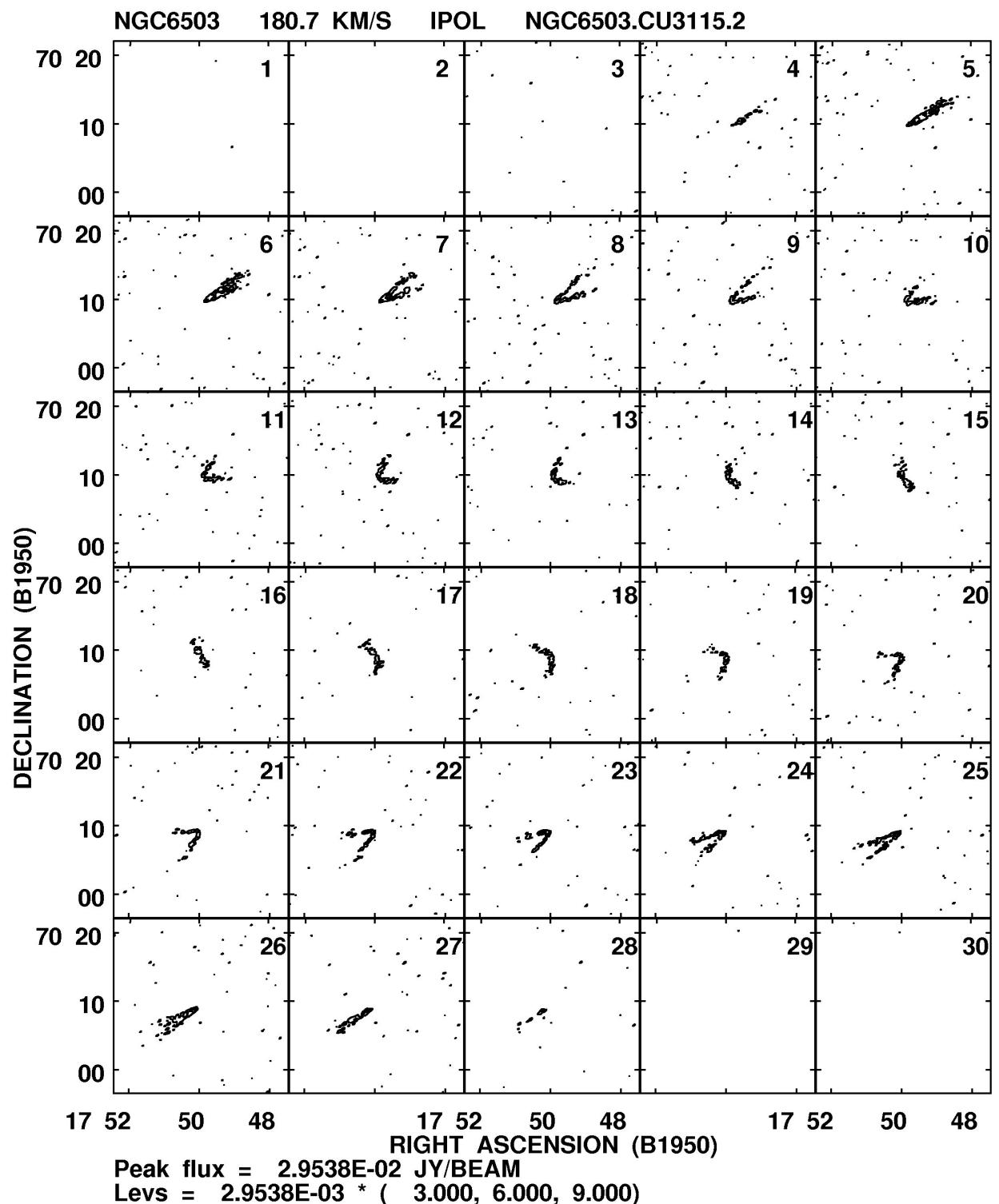


Figure 8.1: Contour images of spectral channels from a cube generated by *AIPS* task [KNTR](#). Data on NGC6503 were provided by Don Wells from observations made with the [VLA](#) on 12 April 1983.

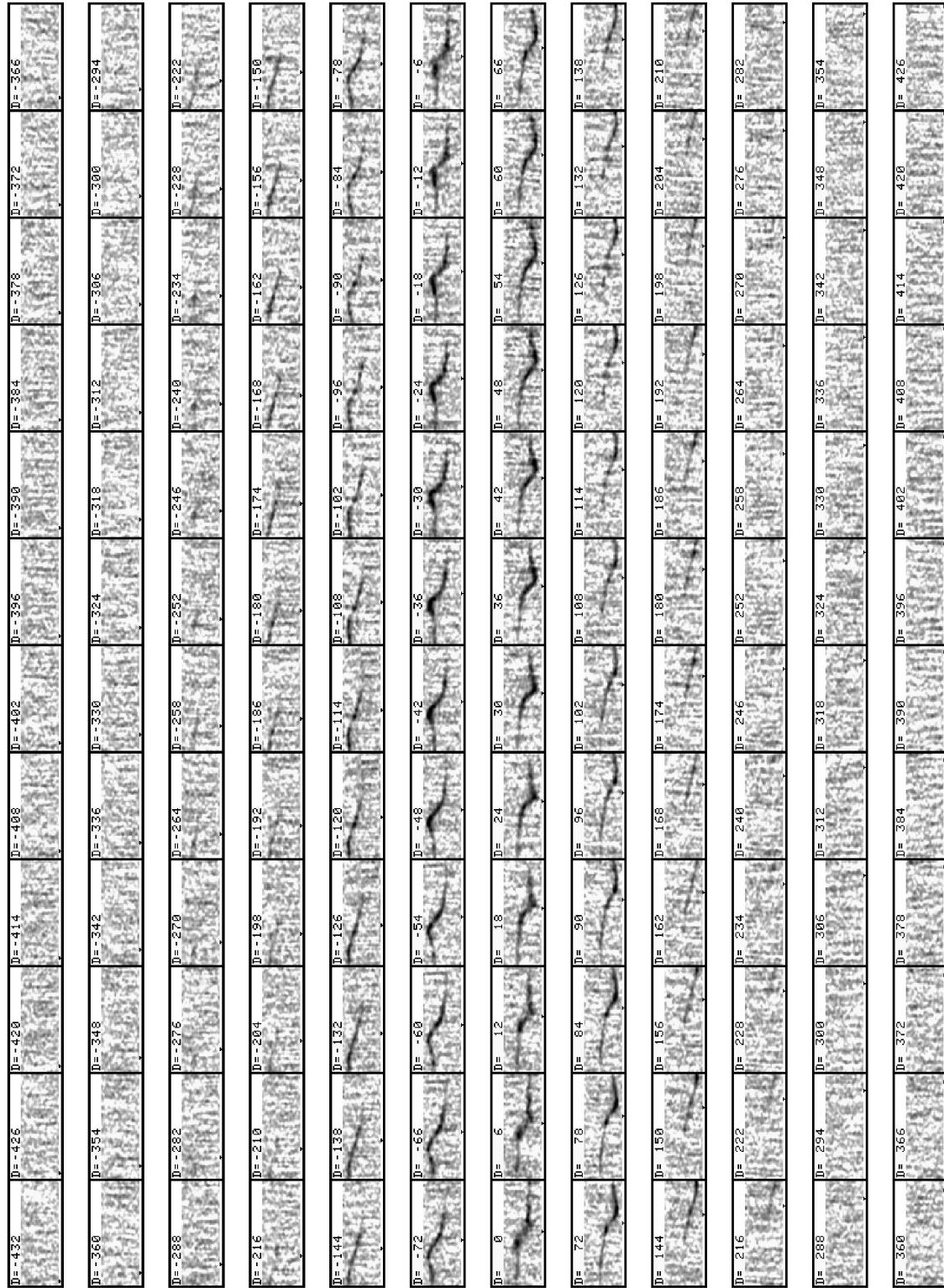


Figure 8.2: Display generated by transposing a data cube into 132 (right ascension, velocity, declination) order and then displaying the central declination position-velocity planes with AIPS verb **TVCUBE**. The TV display was then captured by *AIPS* task **TVCPS** for printing on PostScript printers. Data on NGC6503 were provided by Don Wells from observations made with the **VLA** on 12 April 1983.

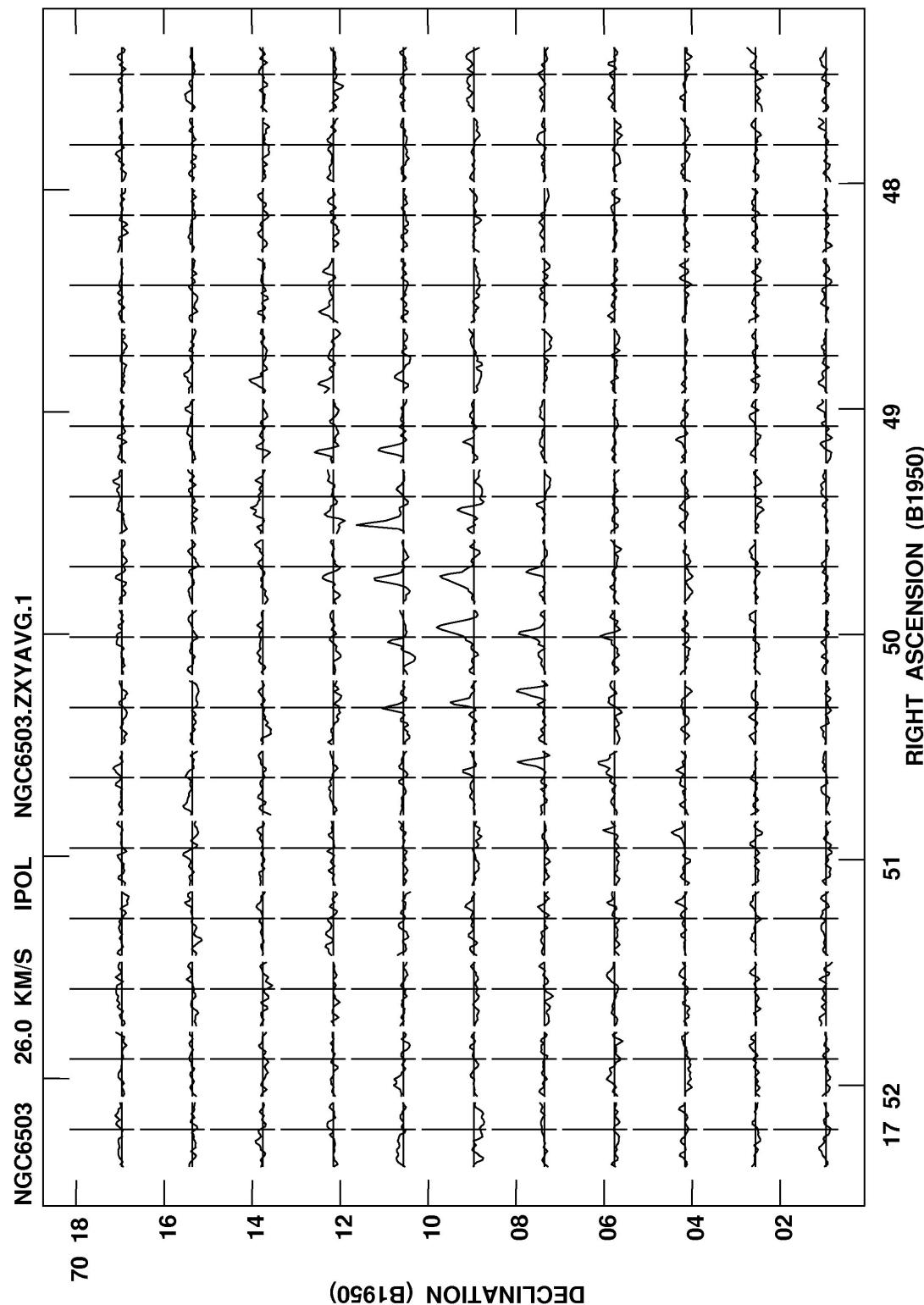


Figure 8.3: Display generated by transposing a data cube into 312 order (velocity, right ascension, declination) and then displaying selected spectra on a regular grid in the angular coordinates with *AIPS* task [PLCUB](#). Data on NGC6503 were provided by Don Wells from observations made by the [VLA](#) on 12 April 1983.

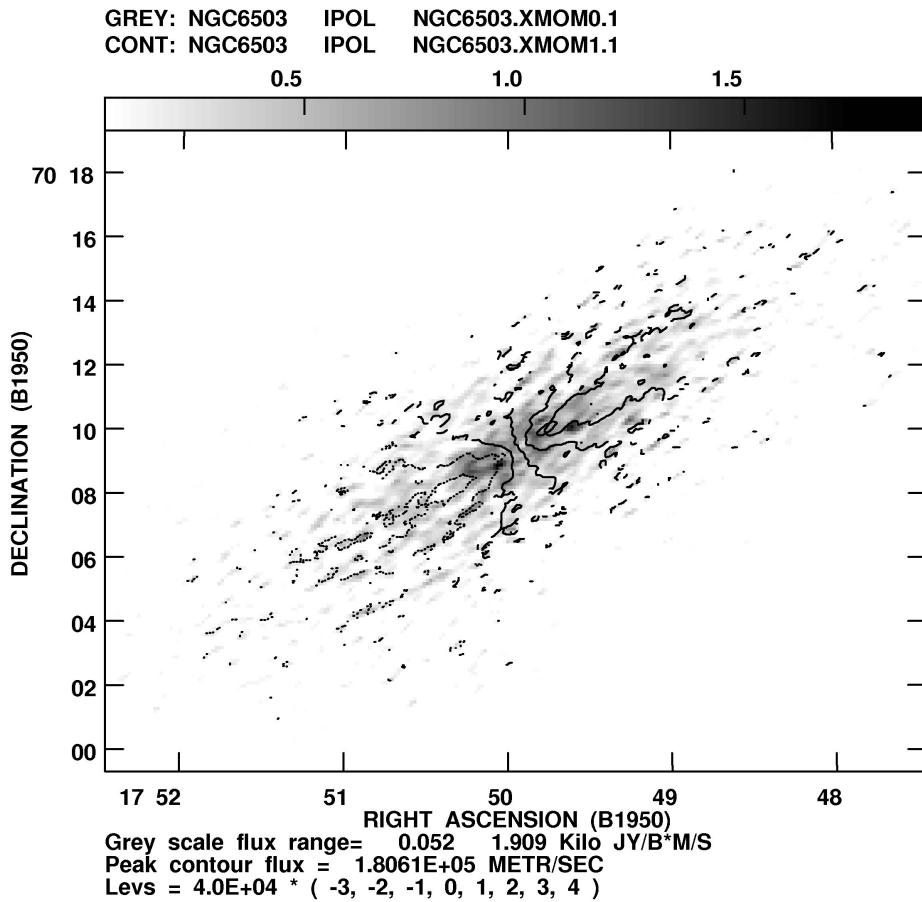


Figure 8.4: Display generated by [GREYS](#) and [LWPLA](#) using the moment 0 image for grey levels and the first moment image for contours. Data on NGC6503 were provided by Don Wells from observations made with the [VLA](#) on 12 April 1983.

8.6 Analysis

A wide variety of programs is available to do further analysis of the data. Exactly which ones you will need depends on the nature of your observations. To see the latest list of symbols related to spectroscopy, enter [> ABOUT SPECTRAL C_R](#) to list them all on your terminal.

or consult Chapter 13 of the *CookBook* for a less current, but paper, copy of the list.

The subject of finding and removing the continuum (aka spectral baseline) was discussed in §8.3 above. In the image domain, task [IMLIN](#) may be used to remove polynomial baselines using the same spectral channels throughout the cube. The task [XBASL](#) can be used to remove baselines in an interactive (hence position-dependent) fashion. Be aware, however, that if you have made an error in the calibration, this has most likely caused slopes in amplitude *and* phase. Therefore, it is generally better to track down the error and correct it than to decide (arbitrarily) to take out slopes in (Cleaned ?) image amplitude. Task [SQASH](#) may also be used to sum or average planes in a cube, which provides, among other things, a simple but crude way to determine a continuum image which can then be subtracted from the cube by [COMB](#).

Smoothing and blanking are important for almost all analysis programs. [CONVL](#) works on cubes and does a spatial smoothing (on position-position-velocity cubes). [CONVL](#) will use the CG table to smooth all image

planes to the same spatial resolution. This is preferable to forcing a resolution with [BMAJ](#) since it smooths the residuals as well as the components. Using all the defaults in [XSMTH](#) performs a Hanning smoothing in velocity (on transposed velocity-position-position cubes) and can be used to do other kinds of smoothing as well. This is not just useful for bringing out weak extended signals. Smoothed images can also assist in determining the boundaries of sources to set windows for subsequent spectral analysis. For example, the smoothed cube could be used to set the [CLIP](#) limits in task [COMB](#) to be applied to the unsmoothed cube.

In fact, finding all regions of significant line signal may be difficult in large image cubes. The task [SERCH](#) offers an algorithm by Juan Uson to find line signals which match specified line widths and exceed a specified signal-to-noise ratio. Histograms may be plotted or printed and a S/N hyper-cube may be written.

The task [BLANK](#) offers a variety of algorithms for “blanking” out regions of bad data or source-free regions in spectral-line cubes. It has an interactive mode, which allows you to indicate with the cursor on the TV what are “good” regions. Set everything but the image name to default values, use [OPCODE 'TVCU'](#) C_R and type [GO BLANK](#) C_R . Then just follow the instructions, pushing button A to lay out the polygon and button D followed by C_R to go to the next image. If marking “bad” regions is easier, set [DOINV TRUE](#) C_R before running [BLANK](#). A less subjective way is to use [BLANK](#) with [OPCODE = IN2C](#). Blanking is done based on the pixel values in a second cube, usually a smoothed version of the original cube. Task [RMSD](#) can be used to blank an image based on its rms in a window surrounding each pixel. It can also be used to compute images of rms using robust, histogram, or median-absolute-deviation (“MAD”) methods.

The blanked cubes can be used to calculate integral profiles with [BLSUM](#) and to calculate moments 0 to 3 of the profiles with [XMOM](#). Thus, the 0-moment image will be the integral under the profile (e.g., total HI), the first moment is the velocity field, etc. [OPTYPE='MAX'](#) in [XMOM](#) will select the peak brightness on each row and make images of it and its coordinate. A comparison of the two types of output can be very illuminating. Task [MOMNT](#) does the smoothing, blanking and calculating of moments all in one run. This is very easy to use, but can be dangerous since you don’t see what is going on. A display of the 0th and 1st moment images computed by [XMOM](#) is shown in Figure 8.4.

Moment images (or any other 2 or 3 images) may be combined in a display in which one image controls intensity, a second image controls hue (color), and a third optional image controls saturation (color richness or purity, done by varying the ratio of white to pure color). In the days of powerful television display devices, this display could be done at truly interactive speeds in the display hardware. Today, most workstations are capable of displaying full color images and the *AIPS* display program supports them. The verb [TVHUEINT](#) will display the two specified TV channels using one as the intensity and the other as hue; see § 6.4.7 *AIPS* task [HUINT](#) may also be used with either full-color TV displays. This task offers a small menu of interactive options to enhance the images and otherwise control the display in manners essentially the same as the verb. It however offers the option to write out an image cube having RGB as its third axis. It can also write an image of the 3-color step wedge. These images may then be processed with the usual display tasks such as [KNTR](#) and rendered in PostScript by [LWPLA](#). *AIPS* task [TVHUI](#) may also be used with either full or pseudo color TV displays. This task also offers a small menu of interactive options to enhance the images and otherwise control the display. It also offers the option to write out an image cube having RGB as its third axis and using much greater mathematical accuracy than is allowed in the TV display.

[RGBMP](#) computes “integral” images another way — as three weighted sums representing the low, center, and high velocity parts of the cube. Like [TVHUI](#), [RGBMP](#) writes its results as a cube with RGB as the third axis. Task [TVRGB](#) can display these outputs (or any other RGB cube or any three image planes), using one image plane to control the red image, one to control green, and one to control blue. It works on real TV displays and full-color workstations and, using an algorithm to minimize the loss of color information, on pseudo-color workstations. Like [TVHUI](#), it offers a small menu of interactive enhancement options. [TVRGB](#) can write 24-bit color PostScript files beginning with the 15JAN96 release. [TVCPS](#) is another way to capture the displays generated by [TVHUI](#) and [TVRGB](#) to send to color PostScript printers.

If you prefer to fit Gaussians instead of calculating moments, the program **XGAUS** can be used. First, it is a good idea to become acquainted with your data cube. In 31DEC16, a highly interactive task called **TVSPC** can let you examine the spectra from one or two transposed cubes at positions selected from an image plane, all on the TV display. The image plane should be some image representing the full field, such as moment-zero or, in polarization, the I polarization image. In 31DEC17, this task can also display another, non-transposed cube with the displayed plane selected from the displayed spectra. In earlier releases, use **XPLOT** first to look at (a sample of) the profiles, before you do any Gaussian fitting. XGAUSS offers a non-interactive mode, but it is frequently unstable and depends on the fit channels and the initial guesses being virtually independent of position. Therefore, in most cases, it is preferable to use the interactive mode, so that you can see what is happening, but be aware that it might be rather time-consuming. In 31DEC13, **XGAUS** prepares a table of solutions which can be edited in various ways after all pixels have been fit once. Images of the fit parameters are examined during the editing process and are written out only after you are happy with the results. The task may be re-started with the same table as often as needed.

In 31DEC13, two similar fitting tasks appeared. **RMFIT** is used to fit rotation measures to Q and U spectral cubes, using the cube of Faraday rotation output by **FARS** to provide initial guesses. It is far more capable than the Faraday rotation synthesis in distinguishing multiple components in the rotation measure. **ZEMAN** fits Zeeman splitting to spectral cubes of I and V polarization. It can fit the line directly or it can fit the individual Gaussians found by **XGAUS** individually. The latter allows multiple magnetic-field values to be found in the same direction. See *AIPS Memo 118*² for a detailed description of these tasks.

In 31DEC17, two more similar tasks appeared to handle the different mathematics required by absorption-line image cubes. When optical depths become significant, the noise in optical depth becomes a strong function of the optical depth, so the fitting is done in the observed absorption spectra rather than in optical depth even though it is the optical depth which is treated as Gaussian. These tasks, **AGAUS** and **ZAMAN** function very much like the tasks for emission-line cubes except that initial guesses are set on a plot of the optical depth spectrum in **AGAUS** and for details of the mathematics. See *AIPS Memo 122*³.

The cube can be rotated with **OGEOM** (if the $\alpha - \delta$ pixels are square), e.g., to align one of the axes with the major axis of a galaxy. If the pixels are not square, use **OHGEO** to regrid the image instead. A single profile can be produced from these images with **SLICE**, then plotted using **TKSLICE**, **TVSLICE**, or **SL2PL** (see § 6.5.2, § 6.4.8, § 6.3.2.2, and Figure 6.5). **PLCUB**, **PLROW** and **XPLOT** are convenient programs for displaying multiple individual spectral profiles after the cubes have been transposed.

To compute and display spectral profiles summed over regions in the two angular coordinates, use **ISPEC** for rectangular regions and **BLSUM** for irregular regions set interactively using the TV display and cursor. Both tasks print their results; both **BLSUM** and **ISPEC** make standard *AIPS* plot files, and **BLSUM** can also make a printer plot. Note that both tasks sum over areas in the first two axes and plot that as a function of position on the third axis, no matter what the three axes actually represent. This suggests a variety of interesting possibilities, such as time functions or line integrals versus position in the source. Both tasks can write the results as slice extension files which may then be printed, plotted and Gaussian fit in a variety of ways. **ISPEC** can plot the spectral derivative of the profile it has computed, which is of interest in Zeeman splitting experiments. Task **RSPEC** is similar to **ISPEC** except that it plots the rms rather than the data. **RSPEC** has options to write out a signal-to-noise image and/or a text file of channel weights as well.

Note that if all you require is a single spectral profile, it may be possible to use **POSSM** which works on the *uv* data directly. If the spectral profile is a function of position within a well-resolved source, then you will have to go to the image plane.

GAL fits models of galaxy rotation to images of the predominant velocity (e.g., the first moment images written by **XMOM**, **XGAUS**, or **MOMNT**). It accepts a second input image to be used as weights for the first input image; it is common practice to use a the zero'th moment image for this. Plots may be produced. The task

²Greisen, E. W. 2015, "Modeling Spectral Cubes in *AIPS*," *AIPS Memo 118* revised, <http://www.aips.nrao.edu/aipsdoc.html>

³Greisen, E. W. 2017, "Modeling Absorption-line Cubes in *AIPS*," *AIPS Memo 122*, <http://www.aips.nrao.edu/aipsdoc.html>

MODVF creates a model velocity field based on a user provided model for the rotation curve, orientation, and warping of the plane.

A much more powerful, but tricky to use, task named **CUBIT** was developed by Judith Irwin and contributed to *AIPS*⁴. This task fits a galaxy rotation and emitter (usually HI) distribution to the full data cube. It uses dynamic memory and so can work on any reasonable size cube. It is much more likely to work if the cube has been very carefully **BLANKed** and if you approach it with patience. Make the best initial guesses for all the parameters that you can. Then fit for one parameter at a time until you begin to get a reasonable result. Then allow the task to fit all of the parameters at once. And finally, check the fits going through the galaxy in halves or quadrants. But first, study **EXPLAIN CUBIT** C_R in detail. The results from this task are worth the trouble to get it to work on your data.

8.7 Additional recipes

8.7.1 Hot banana soufflé

1. Preheat oven to 375° F.
2. Select a 6-cup soufflé dish or other mold and grease it liberally with 1 tablespoon **butter**.
3. Place 6 **eggs**, 1/2 cup **cream**, juice of 1/2 **lemon**, 1 tablespoon **kirsch**, and 1/4 cup **sugar** in blender. Blend until the batter is smooth.
4. Peel 2 large **bananas**, removing any fibers and break into chunks. With blender running, add the chunks one at a time.
5. Break 11 ounces **cream cheese** into chunks and add them to the blender.
6. When all the ingredients are thoroughly mixed, run the blender at high speed for a few seconds.
7. Pour batter into prepared dish and place it in the hot oven. Bake 45–50 minutes until the top is lightly browned and puffy. You may quit when the center is still a bit soft or continue baking until the center is firm.
8. Serve at once. A whipped cream flavored with Grand Marnier makes a nice topping.

8.7.2 Curried bananas

1. Melt 2 tablespoons **butter** in saucepan and cook 2 tablespoons minced **onion** in it for 2–3 minutes.
2. Mix 1 tablespoon **curry powder**, 1 teaspoon **salt**, 1/4 cup **flour**, and a dash of **cayenne pepper** with a little **milk** to make a paste.
3. Add paste to onion, cooking gently for 10 minutes. Add balance of 2 cups **milk** slowly, stirring until it boils.
4. Slice 7 small green **bananas**, and cook gently in the sauce until tender.
5. Serve as a vegetable in a ring of hot cooked rice.

From *Everyday BANANA Recipes*, Banana Distributing Co., New Orleans, published by Bauerlein, Inc. New Orleans, 1927.

⁴Irwin, Judith A. 1994, "Arcs and bridges in the interacting Galaxies NGC 5775/NGC 5774," ApJ, 429, 618-633.

8.7.3 Banana crunch cake

1. Heat oven to 350° F. Grease and flour 10-inch tube (Bundt) pan.
2. In medium bowl, combine 1/2 cup **flour**, 1 cup **coconut**, 1 cup **rolled oats**, 3/4 cup firmly packed **brown sugar**, and 1/2 cup chopped pecans. Mix well.
3. Using fork or pastry blender, cut in 1/2 cup **margarine** until mixture is crumbly. Set aside.
4. In a large bowl, combine 1 1/2 cups sliced very ripe **bananas**, 1/2 cup **sour cream**, and 4 **eggs**; blend until smooth.
5. Add 1 package **yellow cake mix**, Pilsbury Most Supreme is recommended. Beat 2 minutes at high speed.
6. Spread 1/3 of batter in tube pan, sprinkle with 1/3 of coconut mixture. Repeat layers twice more using remaining batter and coconut mixture, ending with coconut mixture.
7. Bake at 350° F for 50 to 60 minutes or until toothpick inserted near center comes out clean. Cool upright in pan 15 minutes; remove from pan. Place on serving plate, coconut side up. Cool completely.
8. HIGH ALTITUDE — above 3500 Feet: Add 3 tablespoons flour to dry cake mix. Bake at 375° F for 45 to 55 minutes.

8.7.4 Panecillos de Plátano

1. Sift together 2 cups **flour**, 1 teaspoon **salt**, and 3 teaspoon **baking powder**.
2. Add 4 tablespoons softened **butter**, mix well, add 3/4 cup **milk**, and stir only until dampened.
3. Roll to 1/2 inch thickness, cut into cookies about 2 inches in diameter, and place on greased cookie sheet.
4. Slice 2 **bananas** in 1/2 inch thicknesses and dip pieces in 2 tablespoons **lemon juice** and then in 2 tablespoons **sugar**. Place a slice on each cookie, pressing it down.
5. Bake in a 425° F oven for 12 minutes or until golden brown.

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

8.7.5 Banana-Rhubarb Crisp

1. Slice 2 large bananas into 1/4-inch rounds. Combine with 2½ cups diced rhubarb, 2 tablespoon sugar, 1/4 teaspoon cinnamon, and a generous dash nutmeg. Spoon the mixture into a well-greased 9-inch pie plate or shallow baking dish (preferably glass or ceramic).
2. In a medium bowl, combine 1/2 cup white or whole-wheat pastry flour, 1/2 cup graham cracker crumbs, 1½ teaspoons baking powder. With a pastry blender or two knives worked in a crisscross fashion, cut in 1/4 cup butter until the mixture is crumbly.
3. Combine 1 egg lightly beaten with 1/4 cup **milk** and stir into the flour mixture. Spoon the batter as evenly as possible over the fruit mixture. Sprinkle with 2 tablespoons sugar.
4. Bake in a pre-heated 400° F oven for 25–30 minutes.

Thanks to Jane Brody's *Good Food Book*.

9 Reducing VLBI Data in *AIPS*

This chapter describes the reduction of VLBI data in *AIPS*. A Step-by-step recipe, covering both simple and more difficult situations, is presented. See Appendix C for simpler and shorter recipes suitable for straightforward observations. Procedures to simplify some of the VLBI reduction steps are mentioned and become available to you after you enter the command `RUN VLBAUTIL CR`. There is also a VLBA pipeline, `VLBARUN`, which is useful for simple datasets, see § 9.2. We also include here some background information concerning the structure of VLBI data sets, the data reduction philosophy and a description of some of the effects for which corrections must be determined and applied. It is important to understand these aspects if you wish to reduce your data reliably. For more background information on VLBI data reduction consult *VLBI and the VLBA*, Astronomical Society of the Pacific (ASP) Conference Series No. 82, 1995.

Programs of particular interest for VLBI may be found in Chapter 13 or displayed from inside *AIPS* by typing `ABOUT VLBI CR` or `APROPOS VLBI CR`. Remember, the best and most complete information available on all *AIPS* verbs and tasks may be found in their `EXPLAIN` files. A 15APR97 or later version of *AIPS* is required to support full Space VLBI data reduction.

Most types of VLBI data, once read into *AIPS*, appear very similar in structure as far as the user is concerned. We shall concentrate on describing the reduction path for data produced by the VLBA correlator, but most operations also apply to MkIII and MkII data. This chapter contains no specific discussion of data from a MkIV correlator since we have little experience as yet with such data. Where appropriate, we shall draw the reader's attention to any differences. In particular, § 9.3.2 deals with reading data from a MkIII correlator into *AIPS* and the steps necessary to prepare such data for calibration. The few extra steps necessary for calibrating phase-referencing observations are described in § 9.5.7.4. Note that successful phase-referencing observations require careful planning **before** the observations. See VLBA Scientific Memo No. 24 by J. Wrobel, C. Walker, J. Benson, and A. Beasley.

Some of the VLBI-related tasks require the ability to read files resident outside *AIPS*. To communicate to *AIPS* the directory in which these files exist it is necessary to define a logical pointer or environment variable. Please refer to § 3.10 to see how this is done.

While the majority of VLBI observations are continuum observations, more sophisticated data reduction techniques are increasingly common. Continuum VLBI observers sometimes also apply spectral-line VLBI techniques to improve the dynamic range of their data sets. For these reasons, this chapter is organized to make the discussion of data reduction techniques more uniform. The overview in this section of the steps involved for several type of VLBI data reduction is meant to guide the user through the rest of the chapter. It is strongly recommended that you read the overview carefully before proceeding.

The expected size of the output *uv* data file can be an important consideration in VLBI data reduction. The disk space required by *AIPS* for a compressed dataset is given by the relation:

$$\text{Disk Space} = 4 \times 10^{-6} N_{\text{Stokes}} N_{\text{chan}} N_{\text{IF}} \frac{N_{\text{ant}}}{2} (N_{\text{ant}} + 1) \frac{T_{\text{expt}}}{\Delta T} \text{ MBytes}$$

where T_{expt} is the total observing time, ΔT the correlator integration time, N_{Stokes} the number of polarization correlation pairs (*RR*, *LL*, *RL*, *LR*), N_{chan} the number of spectral channels per IF, N_{IF} the number of IF's, and N_{ant} the number of antennas in the network. Space VLBI (SVLBI) data can have different integration times for the ground and space baselines of ΔT_g and ΔT_s , respectively, and therefore the total disk space requirement is larger. If N_{ant} is the number of ground telescopes,

$$\text{Disk Space} = 4 \times 10^{-6} N_{\text{Stokes}} N_{\text{chan}} N_{\text{if}} \left[\frac{1}{2} N_{\text{ant}} (N_{\text{ant}} - 1) \frac{T_{\text{expt}}}{\Delta T_g} + N_{\text{ant}} \frac{T_{\text{expt}}}{\Delta T_s} \right] \text{ MBytes}$$

In uncompressed format the same data set will require two-three times the disk space. Be forewarned that some tasks attempt to create uncompressed scratch files which may not fit into the available disk space. The

amount of available free disk space can be determined using the *AIPS* command [FREE](#). The blocks referred to in the [FREE](#) output are equal to 1024 bytes.

Note that certain operating systems are still subject to a 2-Gigabyte limit for any individual file, as a result of their 32-bit file systems. Larger *AIPS* files are supported on DEC Alpha, SGI (running XFS), HP, Solaris (revision \geq 2.6), and Linux (kernel \geq 2.4.2). The last three require all *AIPS*' C code to be compiled with an additional option. The size of the output file can be reduced by IF selection or limited concatenation in [FITLD](#) or by time- or spectral-averaging later using [UVAVG](#), [SPLAT](#) or [AVSPC](#).

It is possible to construct data sets on disk that cannot be written to a single tape using [FITTP](#) because [FITTP](#) uncompresses the data when writing to tape. The task [FITAB](#) is designed to address this problem. [FITAB](#) writes data in compressed form to tape and can write data in pieces to multiple tapes. Note that [FITAB](#) is only available in 15APR99 and later releases and that versions of [FITLD](#) from earlier releases cannot read such data. Packages other than *AIPS* may also be unable to understand these files.

One large point of divergence in the reduction of continuum polarization VLBI data is the question of whether or not to determine separate LL and RR phase solutions. The polarization-specific portions of the recipe given below are based upon the premise that L and R phase solutions should always be determined separately on the grounds that it is safer and should work with data from a wide variety of antennas. If the L-R phase offsets for antennas in your data set are small and constant in time, you may consider modifying the recipe in § 9.1 by determining averaged LL,RR phase solutions everywhere except in step 7.

9.1 VLBI data calibration recipe

See Appendix C for simpler and shorter recipes suitable for straightforward observations.

1. LOAD THE DATA For data from the VLBA correlator, run [FITLD](#) (§ 9.3.1.1); if needed, follow up with [MSORT](#), [USUBA](#), [INDXR](#), [VBGLU](#), [VBMRG](#), and [MERGECAL](#) (§ 9.3.1.4–§ 9.3.1.7). For data from a MkIII correlator, run [MK3TX](#), [MK3IN](#), [MSORT](#), [DBCON](#), [UVAVG](#), [TAMRG](#), [SBCOR](#), and [INDXR](#) as needed (§ 9.3.2.1–§ 9.3.2.6). Data from the Penticton correlator should be loaded using [FITLD](#), sorted ([MSORT](#), § 9.3.1.4), and indexed ([INDXR](#), § 9.3.1.6).

POLARIZATION: The combination of the VLBA correlator and [FITLD](#) incorrectly labels polarizations for dual parallel-hand correlation (RR and LL only), even if RR and LL are in different frequency bands (e.g., LL at 5 GHz and RR at 8.4 GHz). For these types of data, you must run [FXPOL](#) (§ 9.3.1.8).

2. EXAMINE THE DATA It is important to familiarize yourself with the data set before proceeding further, especially if you have little experience with VLBI data. There are many *AIPS* tasks for the examination of your data (see § 9.4 for a fuller discussion). Minimally, you should at first run [LISTR](#), [IMHEAD](#), [EDITR](#), [POSSM](#), [VPLOT](#), and [PRTAN](#). At later stages you will probably find [SNPLT](#), [PRTAB](#), [DTSUM](#), and [SHOUV](#) useful for examining data and calibration tables.

SVLBI: Task [OBPLT](#) allows you to examine different aspects of the spacecraft orbit.

3. PROCESS THE CALIBRATION FILES You will have either received calibration files, or instructions on where to obtain them. Some calibration files can be automatically processed into a form suitable for use within *AIPS* using [VLOG](#) (§ 9.5.2). [ANTAB](#) is now the primary *AIPS* task for loading calibration information from log files.

VLBA CORRELATOR: The VLBA correlator will usually attach calibration information directly to your data for all VLBA and some other antennas. This obviates the need to run [VLOG](#), [ANTAB](#), [PCLOD](#), and [UVFLG](#) to process your *a priori* calibration information for VLBA antennas. Some information for non-VLBA antennas must usually still be loaded from text files.

POLARIZATION: Be careful to make sure that the polarization labeling of the IFs in the calibration text files is the same as the labeling in the data.

4. **CORRECT FOR THE IONOSPHERE** For low frequency experiments [TECOR](#) should be run to remove at least part of the ionospheric contribution to the phase offsets. This should also be considered for higher frequencies (*e.g.*, 8 GHz) depending on the amount of phase wrapping caused by the ionosphere.
5. **CORRECT FOR THE EARTH ORIENTATION PARAMETERS** For phase referencing experiments correlated at the VLBA correlator, particularly between 5-May-2003 and 2-August-2005, [CLCOR](#) ([OPCODE](#)='EOPS') should be run. This will correct the possibly inaccurate Earth Orientation Parameters used by the VLBA correlator. This is particularly important for astrometry experiments but can effect any phase referencing experiment including those correlated outside the above range of dates.
6. **EDIT THE DATA** Identifying and editing bad data now can save you time later. Data should first be edited using [UVFLG](#) to apply editing information supplied with your calibration files (§ 9.5.3). Some useful tasks for examining and editing data are [EDITR](#), [UVFLG](#), [TVFLG](#), [SPFLG](#), [EDITA](#), [BPEDT](#), [FLAGR](#), [FINDR](#), [VPLOT](#), and [QUACK](#).

POLARIZATION: You may want to edit the data consistently in all polarizations (select [STOKES](#) = 'IQUV' within [EDITR](#), [TVFLG](#) or [SPFLG](#)) — this can greatly simplify the imaging stage (see step 15).

7. **POLARIZATION: ADD PARALLACTIC ANGLE CORRECTION** For alt-az mounted antennas, a parallactic angle correction for the rotating orientation of the antenna feeds with respect to the observed source must be performed as the first step in the phase calibration using [CLCOR](#) (§ 9.5.4.1). This step should be performed no later than immediately after the Tsys calibration.

PHASE REFERENCING: You *will* want to perform the parallactic angle correction described above for phase referencing observations even if you only correlated the parallel hands (RR, LL).

8. **APPLY SAMPLER CORRECTION** We now advocate a new amplitude calibration strategy based on VLBA Scientific Memo #37 (Walker 2015). This strategy interleaves the classic *a priori* calibration with instrumental delay and bandpass calibration to improve the calibration of data from the new Roach Digital Backend (RDBE) on the VLBA (see VLBA Observational Status Summary for a description of the RDBE and VLBA Scientific Memo #37 for a discussion of amplitude problems when using the RDBE). With data from before the RDBE you can use either the old or new strategy. First corrections for sampler biases should be applied using [ACCOR](#) (§ 9.5.4.2) for data from the VLBA and some other correlators. Some correlators apply this correction to the data before writing them out — notably the EVN JIVE correlator and correlators used for the Australian LBA. The VLBA hardware and software (DiFX) correlators do *not* apply this correction to the data. Therefore, [ACCOR](#) is required for the VLBA correlators and any others that do not apply the correction. [ACCOR](#) should be benign (do nothing) for those correlators that do apply the correction prior to reading the data into AIPS. Note that you can always run [ACCOR](#) and look at the SN table produced with [PRTAB](#) or [SNPLT](#) to see if it was benign or not.

9. **CALIBRATE THE INSTRUMENTAL DELAYS** *Phase-cals*, or measured single-band and multi-band instrumental phase errors, should be applied using [PCCOR](#) (§ 9.5.4.3). You can manually perform a phase-cal by running [FRING](#) on a limited subset of your data to account for missing phase-cal information or to refine the reported phase-cal measurements (§ 9.5.4.4).

SPECTRAL-LINE: Delay calibration should be carried out only on the continuum sources at this stage. Since there should be no pulse-cals, the “manual” phase-cal method should be used.

POLARIZATION: If running [FRING](#) to determine the instrumental delays, be certain to solve for independent left- and right-polarization delay solutions [APARM](#)(3) = 0. Run [RLDLY](#) after calibrating the instrumental delays, to determine a single delay offset between left and right polarization (§ 9.5.7.11).

10. CALIBRATE THE COMPLEX BANDPASS RESPONSE FUNCTION Run [BPASS](#) or [CPASS](#) to determine the bandpass response function using the *cross-power* spectra (§ 9.5.4.5). The normalization should be over the full bandwidth, be careful because the default channel selection is the inner 75% of the band. If you use the inner 75%, it can lead to amplitude errors of up to 15%.

SPECTRAL-LINE: The bandpass response function should be determined using only the continuum calibrator sources.

11. APPLY ONE MORE AUTOCORRELATION CORRECTION AND [THEN FINISH A PRIORI](#) CALIBRATION After determining the bandpass calibration the autocorrelations are probably different from unity by a few percent. To correct this run the task [ACSCL](#), applying the previous calibration and bandpass correction. Then finally use [APCAL](#) to complete the *a priori* amplitude calibration (§ 9.5.4.6) — this is called the Tsys method of amplitude calibration. [APCAL](#) can also be used to perform opacity corrections.

SPECTRAL-LINE: Unless the line emission is very weak, you may wish to defer amplitude calibration of your *line sources only* until step 16 below. The template method described there is much more accurate than the Tsys method.

12. FRINGE FIT THE [DATA](#) Estimate and remove residual delays, rates and phases using [FRING](#) or [BLING](#) and [CLCAL](#) (§ 9.5.7.7–§ 9.5.7.8).

SPECTRAL-LINE: Only fringe-fit the calibrator source at this stage. Check the coherence of the target source using the resulting solutions to decide whether or not to zero the rate solutions using the 'ZRAT' option in [SNCOR](#) (§ 9.5.7.10).

PHASE REFERENCING: You should **not** fringe-fit on the *target*, or phase-referenced source. Rather, you should fringe-fit on the *cal*, or phase-reference calibrator. When you apply the solution, be sure to set the [CALSOUR](#) and [SOURCES](#) adverbs in [CLCAL](#) appropriately to interpolate the solutions for the cal source onto the target source (see § 9.5.1.2). If you are not interested in astrometric calibration and your target source is strong enough, you may wish to consider fringe-fitting on it to further refine the phase calibration (§ 9.5.7.4).

13. POLARIZATION: ESTIMATE THE INSTRUMENTAL POLARIZATION . Correct for the instrumental polarization terms, commonly known as 'D-terms' using [PCAL](#), [LPCAL](#), or [SPCAL](#) on the polarization calibrator (§ 9.5.7.13). This polarization calibrator should first be fully calibrated and imaged before this step can be performed.

14. POLARIZATION: CALIBRATE THE POLARIZATION POSITION ANGLE If a calibration source with known polarization orientation is available, use [CLCOR](#) to make a final correction to adjust the polarization angles of the target source data (§ 9.5.7.11).

SPECTRAL-LINE or POLARIZATION: The bandpass response function should be determined using only the calibrator source. Unlike step 7, this step cannot be skipped.

15. APPLY THE DOPPLER CORRECTION (spectral-line data only). Run [CVEL](#) to compensate for the changing Doppler shifts of the antennas with respect to the source during the observation and between the different observations (§ 9.5.5).

16. **SPECTRAL-LINE:** REFINE THE AMPLITUDE CALIBRATION Run [ACFIT](#) to amplitude calibrate the program source using the template spectra method (§ 9.5.6). Note that the traditional Tsys method (§ 9.5.4.6) can also be used if the line emission is too weak for the template method to work successfully.

17. **SPECTRAL-LINE:**DETERMINE RESIDUAL RATES Now estimate the residual rates ONLY by running [FRING](#) or [BLING](#) on one or a few spectral points on the target source (§ 9.5.7.10).

18. APPLY CALIBRATION, AVERAGE, AND INSPECT THE FINAL DATA Run [SPLIT](#) or [SPLAT](#) to apply the calibration solutions and to average the data in frequency if appropriate (§ 9.6.1), and [UVAVG](#) to average the data in time (§ 9.6.2). You can also run [SPLAT](#) to combine these three operations into a single step. It is recommended that you take the time to inspect the calibrated data to see if more editing is needed, and to check that no gross calibration errors remain in the data (§ 9.6.3).
19. SELF-CALIBRATE/IMAGE OR SELF-CALIBRATE/MODEL-FIT THE DATA The final complex gain corrections are determined by iterating self-calibration with imaging of the resultant data set. This is called hybrid-mapping. Alternatively, self-calibration can be iterated while fitting models directly to the data — the goal is to self-calibrate using the best model possible. The options are outlined in § 9.7.

SPECTRAL-LINE: One final distinction remains between continuum and spectral-line data. Only one or a few spectral points are used to determine final complex gain corrections which are then applied to all spectral points in the line data. After applying these gains, the line source data can be imaged to form an image cube.

POLARIZATION: While the Stokes I and Stokes V images formed using the RR and LL visibilities will be real-valued, the Stokes Q and Stokes U images formed using LR and RL visibilities can, in principle, be complex-valued. You must use a fully complex imaging and deconvolution technique (see the [HELP](#) files for [CXPOL](#) and [CXCLN](#)) or you can simply edit the LR and RL visibilities to enforce the condition that the whenever you have a RL visibility on a baseline, you also have the LR visibility on the same baseline; this ensures that the Stokes Q and U images are real-valued and allows you to use the standard imaging tasks.

9.2 Pipeline for the VLBA

[VLBARUN](#) is a procedure which uses the VLBA calibration procedures (from [VLBAUTIL](#)) and some logic to calibrate and image VLBA data. [VLBARUN](#) attempts to make intelligent decisions on defaults, so it can be run fairly automatically, if the names of the sources are known. If desired, [VLBARUN](#) will produce diagnostic plot files and write them to disk creating an HTML file to ease examination of these files. Images will be produced, but no self-cal is done, so the images should be considered diagnostic in nature.

Sample inputs for procedure [VLBARUN](#) are:

- | | |
|--|---|
| > RUN VLBAUTIL | to acquire the procedures used by VLBARUN . |
| > RUN VLBARUN | to acquire VLBARUN . |
| > DATAIN /dirname/data.fits | to set data file to load from disk. |
| > OUTDISK n | to set disk for output. |
| > OPTYPE 'CONT' | to say this is a continuum dataset. |
| > CLINT 0 | to use default. |
| > CHREFANT 'FD' | to set reference antenna to Fort Davis. |
| > TIMERANGE 0 | to have VLBARUN determine a good instrumental delay calibration scan. |
| > CALSOUR 'bandpass', 'phasescal"cal1' | to list calibrators, bandpass calibrator <i>must</i> be first. |
| > SOURCES 'phasescal', 'target' | to list phase referencing and target pairs. |
| > INFIL " | do not apply DELZN file. |
| > SOLINT 0 | to use default. |
| > IMSIZE 512 | to make images and specify size of target images. |
| > FACTOR 0 | to make calibrator images 128x128. |

- > **DOPLOT** 1 to make some diagnostic plots.
- > **OUTFILE** 'it /directoryname to specify directory for output plots. If this is set then plots are written out from *AIPS* and organized in an HTML file for easy viewing. Preferably, this directory should be empty at the beginning.
- > **OUTTXT** 'it email@somewhere.edu to specify an e-mail address if the users wants to be notified when the pipeline is done
- .
- > **BADDISK** 0 C_R to specify which disks to avoid for scratch.
- > **VLBARUN** C_R to run the procedure.

VLBARUN will then run and produce the requested number of diagnostic plots. For details on the plots produced for each level of **DOPLOT**, see [EXPLAIN VLBARUN](#). If an e-mail address is specified then a **VLBARUN DONE** or **VLBARUN FAILED** message will be sent. However, the **VLBARUN FAILED** message will only be sent if **VLBARUN** failed because of problems with the inputs, if **VLBARUN** failed because a task it was running failed this message is not sent. It is highly recommended that the user read the explain file for **VLBARUN** before use.

9.3 Loading, fixing and inspecting data

In theory, *AIPS* can process data from multiple frequency bands (FQ numbers in *AIPS* parlance) coexisting within the same data set. However, it is recommended that the data be separated into different frequency bands as soon as possible after loading the data and process each FQ number separately. If you wish to do this, you should do it immediately after performing the relevant steps in § 9.3. The **VLBAUTIL** procedures **VLBAFQS** and **VLBAFIX** do this automatically (**VLBAFIX** is recommended and does other “fixing” tasks like fixing subarrays etc.). If you want to do this by hand use the task [UVCOP](#).

9.3.1 Loading data from the VLBA correlator

9.3.1.1 Running **FITLD**

The information below applies to data from the VLBA Correlator in what the VLBA archive calls “raw” format, more formally known as FITS-IDI format. The archive now also contains data called “calibrated” which have been run through a few of the simplest VLBA procedures but which are nowhere near calibrated. These files are recommended under normal circumstances. They may also be loaded with **FITLD**, but the **DOCONCAT** option does not work with the FITS table format used in the archive.

Data generated by the VLBA correlator are loaded from DAT (or Exabyte) tape (or from disk files) into *AIPS* using **FITLD**. First, physically load your tape and **MOUNT** it (§ 3.9), then run **FITLD**. Often the data on your tape will be divided into a number of separate files (corresponding to separate “correlator jobs”). In this case, run **FITLD** with **NCOUNT** set equal to the number of files on the tape (or a suitably large number), as listed on the paper index which comes with the tape. The adverb **ANTNAME** allows the user to control the antenna numbering if desired. Also set **DOCONCAT** = 1 C_R to ensure that all tape files with the same structure are concatenated into a single *AIPS* file. Note that standard tape handling tasks (e.g., **PRTTP** and **TPHEAD**) can be used to inspect the tape contents.

Note that antennas, sources, frequency IDs, and other things may be numbered differently in different correlator jobs. **FITLD** fixes all this for you, but only if you set **DOCONCAT** = 1 and, better still, load as many files as possible in each execution of **FITLD**. **FITLD** can load VLBA correlator data from multiple disk files so

long as they have the same name plus a consecutive post-pended number beginning with 1. If you forget to put all the related data together with **FITLD** you can use **MATCH** to align the antenna numbers followed by **DBCON** later.

Typical inputs to **FITLD** would be:

> TASK 'FITLD' ; INP C _R	to review the inputs.
> INTAPE <i>n</i> C _R	to specify the input tape number.
> NFILES 0 C _R	to skip no files on tape.
> DATAIN '' C _R	to load from tape, not from disk.
> OUTNAME 'TEST' ; OUTCL 'FITLD' C _R	to specify the name of the output file.
> OUTSEQ 0; OUTDI 1 C _R	to specify the sequence number and disk of the output.
> OPTY '' C _R	to load any type of file found.
> NCOUNT 20 C _R	to load 20 tape files.
> DOUVCOMP 1 C _R	to save disk space by writing compressed data.
> DOCONCAT 1 C _R	to concatenate files with same data structure into one disk file.
> CLINT Δt C _R	set CL table interval to Δt minutes (see discussion below).
> DIGICOR 1 C _R	to request digital corrections (usually VLBA correlator only).
> DELCORR 1 C _R	to request delay decorrelation corrections (VLBA correlator only).
> WTTHRESH 0.65 C _R	flag incoming visibilities with correlator weights less than 0.65.
> SOURCES ''; QUAL 0 C _R	to accept all sources found.
> TIMERANG 0 C _R	to accept data from all times.
> BCHAN 0; ECHAN 0; BIF 1; EIF 0 C _R	to accept all channels in all IFs.
> SELBAND 0 C _R	bandwidth to select (kHz).
> SELFREQ 0; FQTOL 0 C _R	frequency to select with tolerance of 10 kHz.
> OPCODE '' C _R	to not copy the tape statistics table ('VT' table).
> GO C _R	to run the program.

This may seem a bit formidable. For straightforward VLBI observations, there is a collection of procedures to simplify matters including the loading of data. Enter

> RUN VLBAUTIL C _R	to acquire the procedures; this need be done only once since they will be remembered.
> INTAPE <i>n</i> C _R	to specify the input tape number.
> NCOUNT 20 C _R	to load 20 tape files.
> OUTNAME 'TEST' ; OUTDI 1 C _R	to specify the name and disk of the output file.
> DOUVCOMP 1 C _R	to save disk space by writing compressed data.
> CLINT Δt C _R	to set the CL table interval to Δt minutes (see discussion below).
> INP VLBALOAD C _R	to review the inputs.
> VLBALOAD C _R	to run the procedure.

Because the data files tend to be very large, you will usually write compressed data (**DOUVCOMP**=1). These files take about 1/3 of the space of 'uncompressed' data sets, but cause information about the weights of individual polarizations, spectral channels, and IFs to be lost. There is some loss in dynamic range and sensitivity when the weight information is (partially) compromised. (See Appendix F for an expanded discussion of when to and when not to write 'compressed' data sets.) If your observation has more than one DAT or Exabyte tape, simply run **FITLD** for each tape. Setting **DOCONCAT** 1 and setting the output file name completely will ensure that the data from separate tapes with compatible observing band/data structure

will be appended to existing *AIPS* files. Generally, after loading all of your data, you will have one file for each such observing band and/or observing mode. However, observations which require multiple passes through the correlator (including MKIII Modes A, B, and C observations) will have one file per observing mode *per correlation pass*. Data from separate correlator passes can be concatenated using task **VBGLU** and/or merged with task **VBMRG**.

Adverb **CLINT**, which specifies the CL table time sampling interval, must be short compared to the anticipated coherence time. **CLINT** should be set such that the shortest anticipated fringe-fit interval is spanned by a few CL entries. Time sampling in the CL table that is too coarse can lead to calibration interpolation errors when applying the fringe-fit solutions at later stages of the data reduction. If the interval is made unnecessarily short the CL table may become unmanageably large.

It is recommended that corrections for digital representation of the correlated signals be performed in **FITLD** under control of adverb **DIGICOR**, but only for data from the VLBA correlator. **DIGICOR** should be set to one for all continuum and nearly all spectral line experiments. Set **DIGICOR** to 3 or 4 if the digital corrections are desired for a non-VLBA correlator, *e.g.*, some versions of the DiFX correlator. In the special case of spectra with very strong narrow features, the absence of correlator zero-padding may limit the accuracy of the quantization corrections. See the **FITLD** help file for further information. The details of digital correction for FX correlators can be found in *Radio Science* 33, 5, 1289–1296, “Correction functions for digital correlators with two and four quantization levels”, by L. Kogan.

Adverb **DELCORR** enables amplitude corrections for known delay decorrelation losses in the VLBA correlator, as described in *AIPS Memo 90* (1995, “Delay decorrelation corrections for VLBA data within *AIPS*” by A. J. Kemball). Setting **DELCORR=1** will create a correlator parameter frequency (CQ) table for each file written by **FITLD**. Do this for the VLBA correlator only. The presence of this table enables the delay decorrelation correction once the residual delays have been determined in fringe-fitting. These corrections will not be applied if the data were not correlated at the VLBA correlator or if the CQ table is missing. For older **FITLD** files the CQ table can be generated using task **FXVLB** and this must be done before any changes in the frequency structure of the file are made. The CQ table is used for rate and delay amplitude decorrelation corrections after residual delay and rate errors have been determined by fringe-fitting, and are being applied to the data. The CQ table has no immediate effect on the data written by **FITLD** but is essential for later processing.

The **WTTHRESH** adverb can be applied to drop incoming data with playback weights less than the specified limit. Note that data flagged in this way are *unrecoverable* except by re-running **FITLD**. The data weights are normalized to unity so good data usually have weights close to 1.0. You should examine your data carefully if you use **WTTHRESH** to make sure that you have not discarded too much data at this stage. Typically 0.8 or higher is good for the VLBA, but for non-VLBA stations a lower value such as 0.6 or 0.7 may be appropriate.

Calibration data have been transferred from the correlator with your data if your data include VLBA antennas and were correlated after 1 April 1999 and before late 2009, when the DiFX correlator came on line and your **IMHEADER** listing shows the presence of GC, TY, WX, PC and FG tables, as in the example below. If you loaded more than one tape file, you must merge the calibration tables. **VLBALOAD** does the merging for you. See § 9.3.1.2 for additional details. Note that, as this example shows, it is possible your data have calibration transfer tables even though they were correlated before 1 April 1999. If your **IMHEADER** does not show GC and TY tables, you do not have calibration transfer and must manually load calibration information from text files. Also, even if you have calibration transfer, you may still have to manually load calibration information for some non-VLBA antennas (see <http://www.vlba.nrao.edu/astro/obscor/cal-transfer/> for some information in this regard).

The output files produced by **FITLD** are in standard multi-source format (as described in § 4.1) and contain data from all the target and calibrator observations in your observation. **FITLD** also writes a large number of extension tables including an index (**NX**) table, and many tables containing calibration information. A description of the VLBA correlator table types is given in § 9.8. If you are missing the CORR-ID random

axis, your *AIPS* release is stale (pre-15APR97) and you are strongly encouraged to upgrade to the latest release; much of the information presented in this chapter will not be usable with pre15APR97 releases of *AIPS*. Your catalog header should be similar to the one, obtained using verb [IMHEADER](#), given below. If you have GC, TY, FG, WX, and PC tables as in this example data header, your data were processed with calibration transfer - see § 9.3.1.2 for more details.

```

Image=MULTI      (UV)          Filename=329           .OVLB   .   1
Telescope=VLBA               Receiver=VLBA
Observer=TM008                User #= 44
Observ. date=23-SEP-1998      Map date=06-JAN-1999
# visibilities    6567      Sort order  **
Rand axes: UU-L  VV-L  WW-L  TIME1  BASELINE  SOURCE  FREQSEL
              INTTIM  CORR-ID  WEIGHT  SCALE
-----
Type    Pixels   Coord value      at Pixel     Coord incr   Rotat
COMPLEX    1    1.0000000E+00      1.00  1.0000000E+00   .00
STOKES     1   -2.0000000E+00      1.00 -1.0000000E+00   .00
FREQ       16   4.9714900E+09      .53  5.0000000E+05   .00
IF         8    1.0000000E+00      1.00  1.0000000E+00   .00
RA         1    00 00 0 .000      1.00      .000000   .00
DEC        1    00 00 0 .000      1.00      .000000   .00
-----
Coordinate equinox 2000.00
Maximum version number of extension files of type HI is   1
Maximum version number of extension files of type CQ is   1
Maximum version number of extension files of type AT is   1
Maximum version number of extension files of type IM is   1
Maximum version number of extension files of type CT is   1
Maximum version number of extension files of type GC is   1
Maximum version number of extension files of type TY is   1
Maximum version number of extension files of type FG is   1
Maximum version number of extension files of type PC is   1
Maximum version number of extension files of type MC is   1
Maximum version number of extension files of type OB is   1
Maximum version number of extension files of type AN is   1
Maximum version number of extension files of type WX is   1
Maximum version number of extension files of type FQ is   1
Maximum version number of extension files of type SU is   1
Keyword = 'OLDRFQ '  value = 4.97149000D+09

```

Note that the sort order of the output data set is listed as ** rather than TB and that there are no attached CL and NX tables. This happens when [FITLD](#) detects what might be a sub-array condition (two frequency IDs or two sources observed at the same time) on reading the data. In clear cases, the actual simultaneous frequency IDs and sources will be reported. In this case, [FITLD](#) detected the use of multiple integration times on different baselines in the data set; this is common for SVLBI data. The message reported by [FITLD](#) in this case takes the form:

```
*****
FITLD5: Subarray or multiple dump-rate condition found.
FITLD5: NX/CL tables deleted.
FITLD5: Use USUBA to set up subarrays.
FITLD5: Rerun INDXR using CPARM(3) and (4)
FITLD5: *****
```

Unless any of the following criteria are met, the data written by [FITLD](#) are immediately ready for further processing.

- If the sort code has been blanked as in this example, you must sort the data (use **UVSRT** or **MSORT**).
- If the source subarray condition is encountered, you may need to run **USUBA** *after* doing any **ANTAB** that may be needed..
- If the frequency ID subarray condition is encountered, you must separate the frequency IDs into separate data sets; procedure **VLBAFQS** will do this for you.
- If **FITLD** does not leave behind CL and **NX** tables, you must run **INDXR** to create them. Procedure **VLBAFIX** will do all of the above for you.
- If you wish to join together data processed in multiple correlator passes, you must run **VBGLU** and/or **VBMRG**.

FITLD can also be used to load archived *AIPS* data previously written to tape using either **FITTP** or **FITAB**, as described in § 5.1.2. In this case the VLBA correlator-specific adverbs, such as those enabling digital and delay corrections, are not active.

9.3.1.2 Calibration transfer

Beginning on 1 April 1999, the VLBA correlator attaches calibration information for VLBA and some non-VLBA antennas directly to the output FITS files. If your **IMHEADER** listing shows GC, TY, WX, FG, and PC tables, then the correlator has provided calibration information; this service is called *calibration transfer*. Note that projects correlated at slightly earlier dates may also have calibration transfer information. You must have 15APR99 or later version of *AIPS* to take advantage of calibration transfer. Not all antennas provide all the information needed for calibration transfer to the VLBA correlator, see

http://library.nrao.edu/public/memos/vlba/ops/VLBA0_34.pdf

for the latest information on this subject. For those antennas for which calibration information was not transferred by the VLBA correlator, you must process the log files in the traditional way as outlined in § 9.5.2. Calibration for the **VLA** and the GBT began to be transferred with the FITS files in November 2003.

Between April 1999 and late 2009 the information processed by the correlator was somewhat redundant so that the calibration tables, the GC table in particular, must be merged using **TAMRG**, a very general and hence complicated task. There are a couple procedures to do this for you in the **VLBAUTIL** package, **VLBAFIX** and **VLBAMCAL**, if you have previously run **VLBAFIX** your tables have been merged:

- | | |
|--|---|
| > RUN VLBAUTIL C_R | to acquire the procedures; this should be done only once since they will be remembered. |
| > INDISK n ; GETN ctn C_R | to specify the input file. |
| > INP VLBAFIX C_R | to review the inputs. |
| > VLBAFIX C_R | to run the procedure. |

X

You should use **VLBAFIX** after you have finished loading the data from tape, but before you either change the polarization structure of the data with **FXPOL**, load any calibration data for non-VLBA telescopes, or apply the calibration data.

At this point it is a good idea to save the tables that were loaded with your data with **TASAV**. This protects you from having to reload the uv data from scratch if one of the original tables is damaged in some way. If you need to copy a table from the **TASAV**'ed file use **TACOP**.

9.3.1.3 Repairing VLBA data after FITLD

As listed above, there are a variety of reasons why VLBA data may need some repair after **FITLD** has been run. They may need to be sorted into strict time order, to have the subarray nomenclature corrected, to be split into different frequencies, to have the polarization structure fixed, and/or to have the original index (**NX**) table and calibration ((CL) recreated. These repairs can all be done by the procedure **VLBAFIX**, which will examine the data and perform any of the necessary fixes. If the data contain subarrays then the procedure must be told to split the data into multiple subarrays (**SUBARRAY**=2), otherwise it will assume no subarrays and force all the data into one subarray. **VLBAFIX** is intended to replace **VLBASUBS**, **VLAQFS**, **VLBAMCAL** and **VLAQFOL**, all of which can be run individually instead. Also we have recommended in the last few sections that **VLBAFIX** be run, if you have already run it, it does not need to be run again.

- > **RUN VLBAUTIL** c_R to acquire the procedures; this should be done only once since they will be remembered.
- > **INDISK** n ; **GETN** ctn c_R to specify the input file.
- > **CLINT** Δt c_R to set the CL table interval to Δt minutes (see discussion above in § 9.3.1.1).
- > **OUTDISK** m c_R to specify the output disk when needed.
- > **INP VLBAFIX** c_R to review the inputs.
- > **VLBAFIX** c_R to run the procedure.

Remember that all of the **VLBAUTIL** procedures have **HELP** files with good discussions about when to use the simple procedures and when to use the tasks directly.

9.3.1.4 Sorting and indexing VLBA correlator data

If multiple integration times are used on different baselines, the VLBA correlator will write data that are not in strict time-baseline (TB) sort order. **VLBAFIX** (§ 9.3.1.3) will sort your data if needed, if you want to do the sorting by hand do the following. In general, task **UVSRT** can be used to sort randomly ordered *uv* data files in *AIPS*, but has significant disk space requirements through the use of intermediate scratch files. A special task, **MSORT**, has been written which uses a direct memory sort with sufficiently large buffers to accommodate the scale over which the data deviate from true time-baseline sort order. No intermediate scratch files are used and it can be significantly faster than **UVSRT** for this special case. **MSORT** competes with **UVSRT** in performance even in other cases, particularly when the individual visibility records are large due to many spectral channels and/or IFs. The inputs to **MSORT** are similar to those required by **UVSRT** and take the form:

- > **TASK 'MSORT'; INP** c_R to review the inputs.
- > **INDISK** n ; **GETN** ctn c_R to specify the input file.
- > **OUTDISK** n ; **OUTNAM** ‘’, **OUTCLA** ‘’ to specify the output file.
- > **SORT** ‘’ c_R to select default sort order ('TB' or time-baseline).
- > **GO** c_R to run the program.

Note that if the input and output file names are identical, the input file is sorted in place. In-place sorting is dangerous, but may be necessary if there is insufficient disk space for a second copy of the data set or for the intermediate scratch files required by **UVSRT**. *Never abort an in-place sort in progress because you will destroy the integrity of your data set.*

9.3.1.5 Subarrying VLBA correlator data

If the project was observed without using subarrays (defined as times at which separate antennas are simultaneously observing different sources or at different frequencies), *this step involving USUBA is not*

necessary and should be skipped.

If the observations have been scheduled in separate subarrays, defined either by source or frequency selection, the subarrays should be labeled in *AIPS* before proceeding any further. The VLBA correlator does not conserve subarray information, which in any event often has no unique characterization. This is specified in *AIPS* using task **USUBA** which allows subarrays to be defined through either the input adverbs, an external KEYIN text file, or through the use of an automatic algorithm to identify and label subarrays found in the data. The automatic algorithm is recommended, *but its results should be checked closely*. Note that **ANTAB** tables do not know about subarrays that will be assigned by **USUBA**, so you must run all **ANTABS** before running **USUBA**.

If you have subarrays, they need to be sorted, have the subarray nomenclature corrected, and/or have the index (**NX**) table and calibration (**CL**) version 1 table rebuilt. In this case, there is a simplified procedure to combine the three repair operation, **VLBASUBS**. Only use this procedure if you know you have subarrays.

- | | |
|--|---|
| > RUN VLBAUTIL <i>C_R</i> | to acquire the procedures; this should be done only once since they will be remembered. |
| > INDISK <i>n</i> ; GETN <i>ctn</i> <i>C_R</i> | to specify the input file. |
| > CLINT <i>Δt</i> <i>C_R</i> | to set the CL table interval to <i>Δt</i> minutes. |
| > INP VLBASUBS <i>C_R</i> | to review the inputs. |
| > VLBASUBS <i>C_R</i> | to run the procedure. |

The only user-controllable input is the CL table interval; see discussion above. **VLBAFIX** will perform this operation if requested (§ 9.3.1.3).

For automatic subarray labeling by **USUBA**, representative input parameters would be:

- | | |
|--|---|
| > TASK 'USUBA' ; INP <i>C_R</i> | to review the inputs. |
| > INDISK <i>n</i> ; GETN <i>ctn</i> <i>C_R</i> | to specify the input file. |
| > OPCODE 'AUTO' <i>C_R</i> | to identify subarrays automatically. |
| > TIMERANG 0 <i>C_R</i> | to include all times. |
| > ANTENNAS 0 ; SOURCES '' <i>C_R</i> | to include all antennas and sources. |
| > FREQID -1 ; SUBA 0 <i>C_R</i> | to include all frequency IDs and subarrays. |
| > INFIL '' <i>C_R</i> | to use no external file for subarray identifications. |
| > GO <i>C_R</i> | to run the program. |

Sometimes **FITLD** erroneously identifies a subarray condition, usually because of spurious total-power data points. In such cases, you can set **OPCODE** = '' ; **SUBARRAY** = 1 to force all data into the first subarray.

In circumstances requiring **USUBA**, one often wants the calibration in one subarray to apply to other subarrays. **USUBA** will make the subarray column have value 0 which means all. Other tasks may not be so obliging so you may need to use **TABED** or verb **TABPUT** to change tables from subarray-specific to subarray-general.

9.3.1.6 Indexing VLBA correlator data

If **FITLD** had not written **NX** or **CL** tables or it was necessary to sort the data as described in § 9.3.1.4, you must run task **INDXR**. If **VLBAFIX** (§ 9.3.1.3) was run this is done automatically. **INDXR** will generate an **NX** table and, if need be, a **CL** table. Typical parameters for **INDXR** are:

- | | |
|--|--|
| > TASK 'INDXR' ; INP <i>C_R</i> | to review the inputs. |
| > INDISK <i>n</i> ; GETN <i>ctn</i> <i>C_R</i> | to specify the input file. |
| > PRTLEV 0 <i>C_R</i> | to print minimal details of progress. |
| > CPARM 0, 0, <i>Δt</i> , 1 <i>C_R</i> | to set the CL interval to <i>Δt</i> and recalculate the model. |

> `GO CR` to run the program.

Note that `CPARM(4)` can be set to zero unless the correlator model is required in later reduction (e.g., in astrometry or geodesy observations) The CL table sampling interval Δt should be chosen subject to the same considerations given regarding adverb `CLINT` in the discussion of `FITLD` in § 9.3.1.1. `VLBAFIX` will perform this operation if needed (§ 9.3.1.3).

9.3.1.7 Concatenating VLBA correlator data

Sometimes an observation is correlated using multiple passes through the VLBA correlator. In this context, multiple pass means different IFs/pass; this is due to data rate limitations in the correlator. Be careful to have `FITLD` load each pass into a separate disk file; otherwise a very confused data set will be produced. If it is desired to join together the IFs correlated on each pass, the task `VBGLU` should be used. `VBGLU` can only join data sets which are identical except in the frequencies covered. Task `MATCH` may be used to make the antenna, source, and frequency ID numbers in one data set the same as those in another data set so that they may be used as inputs to `VBGLU`.

The inputs to `VBGLU` are rather simple. Each of the input files to be glued together is specified via `INNAME-IN4NAME`, and an output file is specified via `OUTNAME`. The choice of input file 1 is no longer important. No data are lost in the revised version of this task.

> <code>TASK 'VBGLU' ; INP C_R</code>	to review the inputs.
> <code>INDISK n ; GETN ctn C_R</code>	to specify the input file.
> <code>IN2DISK n ; GET2N ctn C_R</code>	to specify the 2 nd input file.
> <code>IN3DISK n ; GET3N ctn C_R</code>	to specify the 3 rd input file.
> <code>IN4DISK n ; GET4N ctn C_R</code>	to specify the 4 th input file.
> <code>OUTDISK n C_R</code>	to specify the output disk.
> <code>GO C_R</code>	to run the program.

With the changes in recording technology, it is also possible that a correlator will not have enough playback units for all antennas in an experiment. In this case, multiple correlations will also have to be done in order to correlate every possible baseline. But, inevitably, certain baselines will appear more than once in these correlations. `FITLD` will load all passes into a single data set (if `DOCONCAT=1`) or separate disk files which may be concatenated, after `MATCH`, with `DBCON`. Sort the data set into 'BT' order with `UVSRT`. Then task `VBMRG` may be used to discard any duplicate data. In 31DEC14, task `DBAPP` may be used to avoid the 2ⁿ proliferation of files, but only if the files are fairly similar in antennas, subarrays, and frequency IDs.

9.3.1.8 Labeling VLBA correlator polarization data

The VLBA correlator does not preserve polarization information unless it is operating in full polarization mode. This results in polarizations not being labeled correctly when both RR and LL polarizations are observed without RL and LR. Each VLBA correlator band is loaded into `AIPS` as a separate IF and is assigned the same polarization. `XPOL` takes a data set from the VLBA correlator and produces a new data set that has the correct IF and polarization assignments. Unfortunately, there is no reliable way to determine the polarization of each IF from the input data set and you must specify the polarization assignments using the `BANDPOL` adverb.

Most VLBA setups assign odd-numbered bands to RCP and even-numbered bands to LCP. In this case `BANDPOL` should be set to '`*(RL)`' (the default) and `XPOL` will generate a new data set that is of equal size to the input data set, but has two polarizations and half the number of IFs. This case normally applies if

LISTR shows pairs of IFs with the same frequency and **QHEADER** shows one pixel on the **STOKES** axis with coordinate value RR, but there may be exceptions to this rule when non-VLBA antennas are used.

Most MkIII and MkIV VLBI setups reverse the polarizations and assign odd-numbered bands to LCP and even-numbered bands to RCP. In this case **BANDPOL** should be set to '*(LR)' and the output data set will again be of equal size to the input data with two polarizations and half the number of IFs. This case normally applies if **LISTR** shows pairs of IFs with the same frequency and **QHEADER** shows one pixel on the **STOKES** axis with coordinate value LL, but there may be exceptions to this rule when non-VLBA antennas are used.

There is a procedure for use with VLBA-only data that attempts to determine which of the above cases applies and then runs **FXPOL** for you, if you ran **VLBAFIX** (§ 9.3.1.3) this has already been done:

- > **RUN VLBAUTIL C_R** to acquire the procedures; this should be done only once since they will be remembered.
- > **INDISK n ; GETN ctn C_R** to specify the input file.
- > **INP VLBAFPOL C_R** to review the inputs.
- > **VLBAFPOL C_R** to run the procedure.

Use **VLBAFPOL** to check whether you need to relabel the polarizations in your data after loading the data, looking for subarrays, and merging redundant calibration data, but before reading any calibration data from non-VLBA stations. **VLBAFPOL** assumes that all of your **FREQIDS** have similar polarization setups. For this reason, you should normally run **VLBAFPOL** after copying each frequency ID to a separate file using **VLBAFQS** (§ 9.5). This strategy also reduces the amount of disk space needed for **VLBAFPOL**.

To use **FXPOL** directly, typical inputs are:

- > **TASK 'FXPOL' ; INP C_R** to review the inputs.
- > **INDISK n ; GETN ctn C_R** to specify the input file.
- > **BANDPOL '*(RL)' C_R** to specify the normal VLBA polarization structure.
- > **GO C_R** to run the program.

Consult **HELP FXPOL** for further information about more complicated cases. Note that **FXPOL** has to write a new output file since the structure of the data is being changed. All standard extension files are also converted, but it is still a good idea to run **FXPOL** before running the calibration tasks.

In single-polarization observations, LL data may simply be mis-labeled as RR or vice-versa. This does not need to be corrected within *AIPS* but the user needs to take this into account when selecting or calibrating the data, particularly in specifying the polarization in the amplitude calibration text file (§ 9.5.2). The Stokes axis can however be modified. Before running **PUTHEAD**, you should run **IMHEAD** to check which axis is the Stokes axis in the catalog header.

- > **INP PUTHEAD C_R** to review the inputs.
- > **INDISK n ; GETN ctn C_R** to specify the input file.
- > **KEYWORD 'CRVAL_m' C_R** to select the Stokes or m^{th} axis in the header.
- > **KEYVALUE = -2 C_R** to set the Stokes value to 'LL' (or -1 for 'RR').
- > **PUTHEAD C_R** to set the coordinate value.

9.3.1.9 Ionospheric corrections

At low frequencies (2 GHz and lower) the ionosphere can cause large unmodeled dispersive delays, seen as rapid phase wrapping. This can be of particular importance in phase referencing observations, where phases must be interpolated over weak sources. Even at high frequencies (e.g., 8 GHz) the ionosphere can be important, depending on the experiment and the condition of the atmosphere during the observation. One

way to remove at least some of the ionospheric phase offsets is by applying a global ionospheric model derived from GPS measurements. The *AIPS* task **TECOR** processes such ionospheric models that are in standard format known as the IONEX format. These models are available from the Crustal Dynamics Data Information System (CDDIS) archive. There is a procedure which is part of **VLBAUTIL**, called **VLBATECR** that automatically downloads the needed IONEX files from CDDIS and runs **TECOR**. It will examine the header and the **NX** table and figure out which dates need to be downloaded, so the observation date in the header must be correct and an **NX** table must exist. See **EXPLAIN VLBATECR** for other requirements.

- > **RUN VLBAUTIL C_R** to acquire the procedures; this should be done only once since they will be remembered.
- > **INDISK n ; GETN ctn C_R** to specify the input file.
- > **INP VLBATECR C_R** to review the inputs.
- > **VLBATECR C_R** to run the procedure.

You can also download the files manually from the CDDIS archive through anonymous ftp and run **TECOR**, see **EXPLAIN TECOR** for detailed instructions on how to retrieve the models. **TECOR** interpolates between the maps of electron content in the ionosphere; therefore IONEX files must be retrieved to cover the entire experiment. Presently, each IONEX file contains maps every 2 hours from hours 00:00 to 24:00. Before November 2002, they contained maps every 2 hours from hours 1:00 through 23:00. Therefore, for example, if an experiment prior to November 2002 started at 0:00 then files must be retrieved for the day of the experiment and the previous day so the times between 0:00 and 0:59 can be interpolated. More recent experiments require two or more files only if they occurred in two or more days.

Typical inputs to **TECOR** are:

- > **TASK 'TECOR' ; INP C_R** to review the inputs.
- > **INDISK n ; GETN ctn C_R** to specify the input file.
- > **INFIL 'FITS:JPLG1230.01'** C_R to set the name of the IONEX file. If there is more than one file, this name *must* be a standard format and be the first file. See **EXPLAIN TECOR** for more details.
- > **NFILES n C_R** to set number of IONEX files to be read.
- > **SUBARRAY 0 C_R** to process all subarrays. This option allows you to process subarrays used on different dates.
- > **ANTENNAS 1 2 3 4 6 7 8 9 10 C_R** to find corrections for all antennas except antenna 5 in a ten antenna experiment. This is important because if the IONEX models do not cover an antenna and it is not excluded here then all the solutions for that antenna will be undefined and the data flagged when the CL table is applied.
- > **GAINVER 1 C_R** to apply corrections to the first CL table.
- > **GAINUSE 2 C_R** to create CL table 2 with the corrections.
- > **APARM 1 0 C_R** to correct for dispersive delay; otherwise only the ionospheric Faraday rotation will be corrected.
- > **GO C_R** to run **TECOR**, correcting for the ionosphere in a new CL table.

The dispersive delays should be checked using **SNPLT** (options **INEXT 'CL'**; **INVERS 2**; **OPTY 'DDLY'**) and **VPLOT** (options **BPARM 0**; **APARM 0**; **DOCAL 1**; **GAINUSE 2**). **TECOR** is only as good as the models, which at this time are quite rough. Therefore, it is a very good idea to compare the corrected and uncorrected phases using **VPLOT**.

CLCOR has a **OPCODE = 'IONO'** to make delay corrections for the ionosphere, similar to the '**ATMO**' operation which is for the atmosphere (§ 9.5.7.5).

9.3.1.10 Corrections to the Earth Orientation Parameters

This correction is only useful for experiments correlated at the VLBA correlator. VLBI correlators must use measurements of the Earth Orientation Parameters (EOPs) to take them out of the observations. These change slowly with time and therefore the EOPs used by the correlator must be continually updated. From 5-May-2003 to 9-Aug-2005 the VLBA correlator used old predicted EOPs which could be significantly wrong and will effect all phase referencing experiments. Incorrect EOPs can both move the position and possibly smear the target of a phase referencing experiment. Self-calibration can improve the smearing. Even outside the above quoted period of particularly bad EOPs the EOPs can be off so it is recommended that all phase-referencing experiments, particularly astrometry experiments should have their EOPs corrected. **CLCOR** (**OPCODE**=’EOPS’) can do this correction. It uses the CT table which is only produced by the VLBA correlator, so at the moment **CLCOR** can only correct experiments processed at the VLBA correlator. **CLCOR** also uses a file of measured EOPs, which can be downloaded from NASA (see **EXPLAIN CLCOR** for details). There is a procedure which is part of **VLBAUTIL**, called **VLBAEOPS**, which downloads the file automatically and runs **CLCOR**. To run the procedure:

- | | |
|--|---|
| > RUN VLBAUTIL C _R | to acquire the procedures; this should be done only once since they will be remembered. |
| > INDISK n ; GETN ctn C _R | to specify the input file. |
| > INFILE ‘ ’ C _R | to automatically download file. |
| > INP VLBAEOPS C _R | to review the inputs. |
| > VLBAEOPS C _R | to run the procedure. |

The procedure will correct the highest CL version while copying it to a version one higher.

To run **CLCOR** manually, download the file using the instructions in **CLCOR**’s explain file. Sample inputs are as follows:

- | | |
|--|--|
| > TASK ‘ CLCOR ’ ; INP C _R | to review the inputs. |
| > INDISK n1 ; GETN ctn C _R | to specify the input file. |
| > OPCODE ‘EOPS’ C _R | to select correction of the EOPs. |
| > GAINVER clin C _R | CL table to read, new default is the current highest version. |
| > GAINUSE 0 C _R | CL table to write; version highest+1 is written unless GAINUSE = GAINVER . |
| > INFILE ‘FITS:usno_finals.erp C _R | to specify file with correct EOPs — note missing close quote to retain lower case letters. |
| > GO C _R | to run the program. |

9.3.1.11 Preparing the OB table for SVLBI data

The spacecraft orbit table (OB) as produced by **FITLD** contains the spacecraft position (x, y, z) and velocity (v_x, v_y, v_z) as calculated to high accuracy from the JPL reconstructed orbit using the SPICE package (developed at JPL). These quantities are calculated by the correlator on-line software and are passed directly through to *AIPS* via **FITLD** by the VLBA correlator. The orbit table is indexed on time and can include information such as the angle between the spacecraft pointing direction and the Sun, the time since the start and end of the last eclipse, and the spacecraft parallactic angle. The latter quantities are not available to the correlator on-line software and, if desired, need to be computed separately for later use in *AIPS* by task **OBTAB**. Additionally **OBTAB** stores orbital elements in the AN table; these are essential for later use in plotting or inspecting spacecraft orbit information. Sample inputs for **OBTAB**, are as follows:

- | | |
|--|----------------------------|
| > TASK ‘ OBTAB ’ ; INP C _R | to review the inputs. |
| > INDISK n ; GETN ctn C _R | to specify the input file. |
| > INVERS 1 C _R | to process OB table 1. |

```
> SUBARRAY 1 CR                                to select subarray number.  

> APARM 1, 0 CR                                to update orbital elements in AN table  

> GO CR                                     to run the program.
```

Note that **APARM**(8) can be used to directly specify which antenna is the orbiting antenna (this task assumes there is only one orbiting antenna).

Task **OBTAB** determines mean orbital elements from the OB table using the spacecraft positions and velocities and updates the AN table under control of **APARM**(1). The orbital elements can be examined by using **PRTAB** to review the updated AN table. The mean elements are used to compute *uv* coordinates for the spacecraft so that model amplitudes AND closure phases and amplitudes can be plotted (by tasks **VPLOT**, **CLPLT**, and **CAPLT**). The orbit table can be plotted using task **OBPLT**. Alternatively, the task **TAPLT** can be used to display individual columns. Use **PRTAB** to determine the names of the columns you wish plotted.

9.3.1.12 Loading the time corrections file for SVLBI data

The round-trip residual delay measurements determined by the tracking stations are supplied to the correlator in FITS format. These so-called delta-T tables are not passed to *AIPS* by the correlator but can be loaded indirectly. This table might be used, for example, to plot the time correction as a function of time. Such information could be useful if a user suspects a loss of coherence due to a poor predicted orbit or a clock jump at the tracking station. The table contains no internal time stamps, so the row number must be used to determine the approximate time of a given entry (there are typically 10 rows per second).

The delta-T tables can be loaded at present using **FITLD** and attached to a null *uv* data file, using input parameters as follows:

```
> TASK 'FITLD' ; INP CR                      to review the inputs.  

> OUTDISK n CR                                to specify a separate output file.  

> OUTNAM 'DUMMY', OUTCLA 'DT'  

> DATAIN 'FITS:3551708.kct.a' CR          to specify the external FITS file.  

> GO CR                                     to run the program.
```

The other **FITLD** adverbs are not relevant in this instance. Note that lower case letters can be used in the **DATAIN** adverb if the trailing quotation mark is omitted. **FITLD** will load the external FITS file successfully but will print an error message complaining that no array geometry table was found. This message can be ignored in this case. The delta-T table will appear as an unknown table of type UK, and can be plotted using task **TAPLT**

9.3.2 Loading data from a MkIII/MkIV correlator

9.3.2.1 Running MK3IN

Data from a MkIII correlator, such as that in Bonn, Germany or Haystack, Massachusetts, can also be read into *AIPS*. To do this you need to be supplied with the so called "A" tape output, also known as "type 52's." These data tapes can be read and translated by the task **MK3IN**. The process of reading MkIII correlator data into *AIPS* and preparing it for further processing is more cumbersome than the equivalent process for VLBA correlator data. This simply reflects the manner in which data are generated on a baseline-based correlator with a limited number of playback drives. MkIII data may also appear in the form of a Unix tar file. For such data, use **M3STAR** and **TFILE** rather than **MK3IN** and **AFILE**, respectively.

Before running **MK3IN**, run the task **MK3TX** to extract the text files from the MkIII archive tape. These text files contain information about the correlated scans in the data set. **MK3TX** will first provide an index of all the

text files and then ask you to select files for loading onto disk. It then asks you interactively for the desired destination of the text files. It is important to load and concatenate all the “A” files, *i.e.*, those files having names like `Atttt`. The meaning of the other text files is described in the [MK3TX](#) Explain file. Sometimes the text files are not on the tapes, which means that you cannot select sub-sets of the data using the A-files, but is not otherwise catastrophic.

If the A-files are present and have been loaded onto the disk, use [AFILE](#) to sort and edit these files to produce a list of scans to be loaded by [MK3IN](#). Use [APARM](#) settings in [AFILE](#) to establish criteria for selecting between any duplicate scans which may appear on the archive. If the data set contains data at multiple frequencies, you should edit the resulting output text file so that there is a version for each frequency, containing only those scans at that frequency.

The final step before running [MK3IN](#) is to create another text file which provides the commands for the task. This step is necessary since some information that is needed by [AIPS](#) is not present on the tape. Ideally, in this text file (as shown below), the parameter STATIONS should be a list of all the stations correlated, with the exact name used at correlation. If you do not have such a list, you can instead specify a list containing STATIONS ‘ANY’, ‘ANY’ ... Note that there must be at least as many ‘ANY’ entries as there are stations in the data set or some of the stations will not be loaded. The parameters in this text file are:

`STOKES='RR','LL'`

the Stokes range of the output file. The standard abbreviations are used to select the polarization range. The largest consistent range is used. For example: `STOKES='RR','LL'` will cause only RR and LL to be written. `STOKES='LL'` will cause just LL to be written. `STOKES='RR','LR'` will cause all four circular polarization combinations to be in the output file, since RR and LR span the range of allowed [AIPS](#) Stokes values.

`FREQCODE='R','L','r','l'`

the polarization codes used by MkIII correlators are anything but standard and they need to be supplied to [MK3IN](#) using the parameter FREQCODE. The one character polarization identifiers are expected in the order RR, LL, RL, and LR. The usual correlator convention is ‘R’=RR, ‘L’=LL, ‘r’=RL, ‘l’=LR and this is the default assumed by [MK3IN](#). However, other codes are possible. For example `FREQCODE = 'A', 'B', 'C', 'D'` will interpret ‘A’ as RR, ‘B’ as LL and so forth, while `FREQCODE = 'R', 'C', 'r', 'l'` will use the default abbreviations except that ‘C’=LL. If [MK3IN](#) encounters an unidentified polarization code the task will report: AT20XX: Unidentified Stokes parameter: ‘X’. In this case, modify the FREQCODE parameter to include this polarization identifier. This will ensure that polarizations are not misidentified inadvertently.

`NO_POL=2`

the number of polarization correlations (*e.g.*, RR, LL, RL and LR), the default is 1.

`STATIONS='NRAO','VLA','OVRO','FDVS','MPI'`

/

station names.

keyin style delimiter.

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

- > `TASK 'MK3IN' ; INP CR`
- > `INFILE 'MYVLB:PARAM.LIS' CR`
- > `IN2FILE 'MYVLB:AFILE.LIS' CR`
- > `INTAPE 4 CR`

to review the inputs.

to define the text control file.

to point to a file containing a list of scans to be loaded as produced by [AFILE](#)

to specify the tape drive number.

> NFILES 0 C _R	to skip no files on tape.
> OUTNA 'EXP 86-34' C _R	to select the output file name.
> OUTCL 'MK3IN' C _R	to select the default output class name.
> REFDATE '12/11/89' C _R	to tell MK3IN the start date of the observations — get this right or you may get negative times.
> SOURCES " C _R	to accept all sources found.
> TIMERANG 0 C _R	to accept data from all times found.
> DOUVCOMP 1 C _R	to write data on disk in compressed format.
> APARM 1, 0 C _R	to set the time increment in the CL table entries in minutes.
> APARM(7) 1 C _R	to separate sidebands into separate AIPS IFs; the default is to store both USB and LSB in the same IF.
> GO C _R	to run the program.

If the data are contained on more than one Exabyte or DAT tape, load the second tape and re-run MK3IN, setting DOCONCAT = 1 C_R so that the data are appended to the previous output file. Before running MK3IN a second time, it is important to set the list of STATIONS in the control file to exactly those found when loading the first tape; use PRTAN on the output file to obtain this list. Also leave additional 'ANY' entries after the list for any stations that are on the second tape but which were not on the first tape. The use of DOUVCOMP = 1 is recommended for most data sets, see Appendix F.

9.3.2.2 Sorting MkIII/IV data

The AIPS data files created by MK3IN will be in an arbitrary sort order. Use UVSRT or MSORT to sort them into time-baseline order:

> TASK 'UVSRT' ; INP C _R	to review the inputs.
> INDISK <i>n</i> ; GETN <i>ctn</i> C _R	to select the input file.
> OUTNA INNA ; OUTCL 'TBSRT' C _R	to specify the output file.
> SORT 'TB' C _R	to sort to time-baseline order.
> GO C _R	to make the sorted <i>uv</i> file.

9.3.2.3 Concatenating MkIII/IV data

If you did not set DOCONCAT=1 when running MK3IN and as a result several files were loaded from tape for one observation, use DBCON to concatenate them together. In order to have the concatenated data all appear in a single subarray, both input files for DBCON must have the same reference day number and identical antenna numbers. That is, the antennas extension (AN) files with each input *uv* data file must be the same. MATCH may be used to repair discrepancies.

You may list the contents of AN files using PRTAN. To run DBCON:

> TASK 'DBCON' ; INP C _R	to review the inputs.
> INDISK <i>n1</i> ; GETN <i>ctn1</i> C _R	to select the 1 st input file.
> IN2DISK <i>n2</i> ; GET2N <i>ctn2</i> C _R	to select the 2 nd input file.
> OUTNA INNA ; OUTCL 'DBCON' C _R	to specify the output file.
> DOARRAY 1 C _R	to force DBCON to mark the output data records as being in the same sub-array. For this to work properly, both of the input files must have the same reference day and have identical antennas files.

> **GO** C_R to concatenate the two files.
 In 31DEC14, task **DBAPP** may be used to avoid the 2ⁿ proliferation of files, but only if the files are fairly similar in antennas, subarrays, and frequency IDs.

9.3.2.4 Merging MkIII/IV data

MkIII VLBI correlators usually produce redundantly correlated data. You must merge the data using **UVAVG**:

> TASK 'UVAVG' ; INP C _R	to review the inputs.
> INDISK <i>n</i> ; GETN <i>ctn</i> C _R	to specify the input file.
> OUTNA INNA ; OUTCL 'UVMRG' C _R	to specify the output file.
> YINC 4.0 C _R	to set the averaging interval of the input data records (in seconds).
> OPCODE 'MERG' C _R	to direct the task to perform the merge operation.
> GO C _R	to run the program.

The CL table should only contain one entry for each antenna at each time stamp. But, due to the merging process described above and the fact that redundant correlations may have been performed, there is one step to follow before you have consolidated your database fully. You must run **TAMRG** to remove the redundant CL entries:

> TASK 'TAMRG' ; INP C _R	to review the inputs.
> INDISK <i>n</i> ; GETN <i>ctn</i> C _R	to specify the input file.
> INEXT 'CL' C _R	to specify the table type to merge.
> INVER 1; OUTVER INVER C _R	to process the input table in place.
> APARM 4, 1, 4, 0, 1, 1, 1, 0 C _R	to control the merging: don't ask why, just do it!
> BPARM 1, 4 C _R	to set compared columns — again, don't ask.
> CPARM 1.157e-5, 0.2 C _R	to set degree of equality — ditto.
> GO C _R	to run the program.

9.3.2.5 Correcting MkIII/IV sideband phase offsets

If your observation contains a mixture of VLBA and non-VLBA antennas and you have not stored the sidebands as separate IFs, there will be a phase offset of about 130° between the upper and lower sidebands on baselines from VLBA to non-VLBA antennas. A correction for this offset is achieved using the task **SBCOR**:

> TASK 'SBCOR' ; INP C _R	to review the inputs.
> INDISK <i>n1</i> ; GETN <i>ctn</i> C _R	to specify the input file.
> OUTNA INNA ; OUTCL 'SBCOR' C _R	to specify the output file.
> BCHAN 1 C _R	to specify the lowest channel of lower sideband.
> ECHAN 4 C _R	to specify the highest channel of lower sideband.
> APARM (1) 0 C _R	to apply the default phase offset (<i>i.e.</i> , -130 ^{deg} .)
> ANTENNAS = VLBA ; INP C _R	to specify the VLBA antenna numbers; the = sign is required here. The verb VLBA reads the antenna file to find VLBA antennas.
> GO C _R	to run the program.

If you have loaded the **VLBAUTIL** procedures, then you may use a procedure called **ANTNUM** to translate a station name into a station number. Thus **ANTENNAS** = **ANTNUM**('BR'), **ANTNUM**('FD'), The verb VLBA in is easier.

9.3.2.6 Indexing MkIII/IV data

Next, you must index your data. The [NX](#) table is useful as a summary of the file for you, and is also used by the calibration programs to provide quick access for reading data. Create this file with [INDXR](#):

- > [TASK 'INDXR'](#) ; [INP C_R](#) to review the inputs.
- > [INDISK n](#) ; [GETN ctn C_R](#) to specify the input file.
- > [CPARM 0, 30, -1 C_R](#) to allow \leq 10-minute time gaps within scans, to limit scans to \leq 30 minutes, and to not create a new CL table.
- > [GO C_R](#) to run the program.

Other than these initial loading and merging steps, the reduction of MkIII and MkIV correlator data is identical to that of VLBA correlator data.

9.4 Tools for data examination

Before proceeding further it is important to examine the data, to make sure they are all loaded, and (especially if this is the first time you have reduced VLBI data) to familiarize yourself with the data structure. As processing continues it is also important to inspect the data periodically to check on the progress of the calibration. Use the verb [IMHEAD](#) regularly to check the *uv*-data header, particularly the list of tables (as seen in § 9.3.1.1).

Some tasks that can be used to examine the data and the associated tables are [LISTR](#), [DTSUM](#), [POSSM](#), [VPLOT](#), [CLPLT](#), [CAPLT](#), [EDITR](#), [TVFLG](#), [SPFLG](#), [SNEDT](#), [SNPLT](#), [PRTAB](#), [FRPLT](#), [PRTAN](#), [COHER](#), [OBPLT](#), and [SHOUV](#). Some of these tasks are described in the next few pages.

9.4.1 Textual displays

As a first step, use the procedure [VLBASUMM](#) to print out the essential contents of your data set:

- > [RUN VLBAUTIL C_R](#) to acquire the procedures; this should be done only once since they will be remembered.
- > [INDISK n](#) ; [GETN ctn C_R](#) to specify the input file.
- > [DOCRT -1 C_R](#) to direct the output to the line printer.
- > [INP VLBASUMM C_R](#) to review the inputs.
- > [VLBASUMM C_R](#) to run the procedure.

This will make a listing of the scans, sources, frequency structure, and antennas found in your data set. You should run this procedure after “fixing” the data with [VLBAMCAL](#), [VLBAFQS](#), [VLBASUBS](#), and [VLBAFPOL](#), but you may also find it useful on the initial dataset.

[VLBASUMM](#) runs the task [LISTR](#) to give a listing of the scans, with source names, time ranges, frequency ID's and total number of visibilities per scan for each of your output files. It is often useful to print out a paper copy of this to facilitate later data plotting/editing. If you did not do [VLBASUMM](#), use:

- > [TASK 'LISTR' ; INP C_R](#) to review the inputs.
- > [INDISK n](#) ; [GETN ctn C_R](#) to specify the input file.
- > [OPTYP 'SCAN' C_R](#) to request printing of scan summaries.
- > [DOCRT -1 C_R](#) to direct the output to a printer.
- > [OUTPRINT '' ; C_R](#) to have the output printed immediately.

> **GO** C_R to run the program.

Note that at the end of the above **LISTR** output is useful information about the frequency structure of your data set. **LISTR** with **OPTYP = 'LIST'** and **DPARM(1) = 1** is also a good way to look for phase coherence. If **LISTR** fails at this point, you may have forgotten to run **INDXR** and/or **MSORT** (see § 9.3.1.6 and § 9.3.1.4).

Other verbs/tasks for inspecting your data include;

1. **IMHEAD** lists the file header including information on the number of frequency channels and Stokes parameters in the data and gives a list of all the extension tables.
2. **PRTAB** can be used to print the contents of any of these extension tables, for instance the SU or Source table contains information about each of the sources observed. Some tables have very many columns. You can use input parameter **BOX** to select the list of the columns you want to print.
3. **PRTAN**, run by **VLBASUMM**, provides a listing of the antenna names and their associated antenna numbers. This is useful because antennas are generally specified by their antenna numbers. (Procedure **ANTNUM** in the **VLBAUTIL** set allows you to translate station names into numbers.)
4. **DTSUM** produces a matrix showing the number of visibilities on each baseline for each scan allowing a check to be made that all baselines have been loaded. It also tells you the data integration time. **DTSUM** also has a mode (triggered by setting **APARM(1) = 1**) that will produce a useful matrix summary of your whole data set. **DTSUM** does not properly report the integration times when there are multiple integration times in the data set.

9.4.2 Spectral displays: POSSM

Your data file will probably contain a number of IFs, observed at different frequencies, corresponding to the separate “IF channels” used during the observations. Use **IMHEAD** to find the number of IF channels and the number of spectral channels per IF channel, or examine the **LISTR** output. The reason that the data must be stored in narrow spectral channels, even for continuum applications, is that, in VLBI, the geometrical and propagation errors affecting the data can be large enough to cause significant phase changes across an IF channel bandwidth, preventing a coherent integration over the full bandwidth.

The frequency structure of the data can be inspected using **POSSM**, which provides a plot of visibility data as a function of frequency as integrated over a specified time interval. Optionally, data from up to nine baselines can be plotted on a single plot page. Initially it may be interesting to view the frequency structure of data on a bright calibrator source, as in the example below. Because, prior to calibration, the phases in each IF channel are likely to vary rapidly with time, it is important to average data coherently only over a short time interval. In general, you will see phase slopes and offsets affecting the data; these phase errors must be determined and removed before the data can be averaged in frequency and/or time. See § 9.5.7 for more information and a sample plot. There is a procedure simplifying the use of **POSSM**:

> RUN VLBAUTIL C _R	to acquire the procedures; this should be done only once since they will be remembered.
> INDISK <i>n</i> ; GETN <i>ctn</i> C _R	to specify the input file.
> SOURCES ' ' C _R	to plot all sources.
> TIMERANGE 0 C _R	to plot all times.
> SUBARRAY 0 C _R	to plot all subarrays.
> REFANT <i>n</i> C _R	to plot the cross-power spectrum for baselines with antenna <i>n</i> .
> STOKES 'I' C _R	to plot Stokes I.
> GAINUSE <i>CLin</i> C _R	to apply CL table <i>CLin</i> to the data before plotting.

> **DOTV** 1 C_R to plot the data on the TV; -1 to make a plot file.
 > **VLBACRPL** C_R to plot the data.

To use **POSSM** directly to display the visibility spectrum of a source on the TV, use:

> TASK 'POSSM'; INP C _R	to review the inputs.
> INDISK <i>n</i> ; GETN <i>ctn</i> C _R	to specify the input file.
> SOURCE 'OQ208', '' C _R	to specify a single source name.
> TIMER 1 2 15 0 1 2 15 30 C _R	to define a time range.
> ANTENNAS 8 ; BASELINE 0 C _R	to plot all baselines to antenna 8.
> DOCAL -1 C _R	to plot the data without calibration.
> BIF 0 ; EIF 0 ; BCHAN 0 ; ECHAN 0 C _R	to include all IFs and spectral channels.
> STOKES 'HALF' C _R	To plot RR and LL separately.
> CODETYPE '' ; POLPLOT '' C _R	to plot amplitude and phase.
> SOLINT 0 C _R	to average over the full time range.
> APARM 1, 1, 0, 0, -180, 180, 0, 0, 3, 0 C _R	to control the plot: APARM(1)=1 to use vector averaging, APARM(2)=1 to use fixed scale plots, APARM(5) and APARM(6) to set phase range, APARM(9)=3 to plot all IFs and polarizations together in one diagram.
> NPLOTS 9 ; BPARM 0 ; OUTTEXT '' C _R	to have 9 plots per page without division by "channel 0" and without writing the spectrum to a file.
> DOTV 1 C _R	to plot on the TV, else create plot extension.
> BADDISK 0 C _R	to use all disks for scratch.
> GO C _R	to run the program.

Note that the amplitudes are totally uncalibrated at this stage and are in units of "correlation coefficients"; these will generally appear on plots mislabeled as mJy (representing multiples of 10^{-3} in correlation coefficient). **POSSM** can produce text output into the file given by **OUTTEXT**.

Sample **POSSM** displays are given in Figure 9.1, Figure 9.2, and Figure 10.2.

Task **SHOUV** with **OPTYPE** 'SPEC' will display the data from a number of channels on the printer with optional time averaging.

9.4.3 Time displays: VPLT, CLPLT, and CAPLT

The task **VPLT** can be used to view the visibility data as a function of time (or other variables). Again, data from several baselines can appear on one plot page. Plots of amplitudes and phases and several other quantities can be made, although, to view closure phase and amplitude, you must use tasks **CLPLT** and **CAPLT**. Note that **VPLT** can average spectral channels or plot them individually under control of **AVGCHAN** and can plot spectral channels and IFs in separate panels or all together under control of **CROWDED**. Data points are plotted with a user-selected **SYMBOL** and may be connected by lines under control of **FACTOR**. Calibration can be applied to the data before they are plotted. The data can be averaged in time and the max/min within the interval plotted along with the average. Also, if desired, a model can be plotted against the data. The model can either be displayed at the times of the data samples or, with somewhat less accuracy, continuously, even at times for which there are no associated data or recorded *uv* coordinate values.

The following parameters will display uncalibrated amplitudes and phases from a single spectral channel of a single IF channel for a short scan on a bright calibrator:

> TASK 'VPLOT' ; INP C_R	to review the inputs.
> INDISK n ; GETN ctn C_R	to specify the input file.
> CLR2NAME C_R	to ensure no model is plotted.
> SOURCE 'OQ208' C_R	to specify the source name.
> BIF 4 C_R	to give first included IF channel.
> EIF 4 C_R	to give last included IF channel; if EIF > BIF then IFs are averaged.
> BCHAN 8 C_R	to set the lowest spectral channel to include in average prior to plotting.
> ECHAN 8 C_R	to set the highest spectral channel to include in average prior to plotting; no averaging in this case.
> TIMER 1 2 15 0 1 2 25 00 C_R	to define a 10-minute time range.
> DOCAL -1 ; DOBAND -1 C_R	to apply no calibration tables or bandpass tables to data.
> OPTYP ''	to display cross-correlations; 'AUTO' to get auto-correlations.
> SOLINT 0	to do no time averaging of the data before plotting.
> XINC 1 C_R	to plot every record.
> BPARM 0, -1 C_R	to set <i>x</i> -axis type (BPARM(1) = 0 plots time (Hrs, min, sec)), <i>y</i> -axis type (BPARM(2)= -1 plots both amplitude and phase), and to use self-scaling. See EXPLAIN VPLOT C_R for other options.
> NPLOT 4 C_R	to plot 4 baselines/page.
> GO C_R	to run the program.

An example **VPLOT** output appears as Figure 9.1. It may be useful to make a plot of data “weight” versus time on the autocorrelation data from each antenna (set **BPARM(2)=16** and **OPTYP = 'AUTO'**). The weight depends on the number of valid bits correlated and is a good indication of tape playback quality.

9.4.4 EDITR

Another task which can display data as a function of time is **EDITR**. This program is used primarily to edit data interactively (see § 5.5.2), but its interactive aspects (*i.e.*, allowing the user to “zoom” in on certain time periods) make it useful for pure data inspection. **EDITR** has been much improved in recent releases to offer options reminiscent of difmap, the VLBI data reduction package from CalTech. In particular, the **CROWDED** adverb allows displays of all IFs and/or all polarizations at the same time (and color may be used to separate them visually) and the **FLAG** **QUICKLY** run-time option allows fast sample deletion with only quick mouse clicks. It is well worth exploring the abilities of this powerful program. Tasks **TVFLG** (§ 4.4.3), **SPFLG** (§ 10.2.2), and **WIPER** are also useful in this way.

9.4.5 SNPLT

Supplied with your VLBI data will be a number of important tables used for calibration, and many more are generated as calibration proceeds. § 9.8 summarizes the contents of each of these tables. Two of the most important tables are the calibration or CL tables and the solution or SN tables.

The task **SNPLT** should be used periodically to inspect the contents of the latest CL and SN tables. There is a simplified procedure for making these plots:

> RUN VLBAUTIL C_R	to acquire the procedures; this should be done only once since they will be remembered.
-------------------------------------	---

> INDISK <i>n</i> ; GETN <i>ctn</i> C_R	to specify the input file.
> INEXT 'CL' C_R	to plot a CL table.
> INVERS 0 C_R	to plot the highest version.
> SOURCES ' ' C_R	to plot all sources.
> TIMERANGE 0 C_R	to plot all times.
> STOKES ' ' C_R	to plot both R and L solutions.
> SUBARRAY 0 C_R	to plot all subarrays.
> OPTYPE 'AMP' C_R	to look at amplitudes; 'PHAS', 'DELA', and 'RATE' are other useful choices.
> DOTV 1 C_R	to plot the data on the TV; -1 to make a plot file.
> VLBASNPL C_R	to plot the data.

Using **SNPLT** directly, the example below plots the antenna-based amplitude corrections stored in CL table *m*.

> TASK 'SNPLT' ; INP C_R	to review the inputs.
> INDISK <i>n</i> ; GETN <i>ctn</i> C_R	to specify the input file.
> OPTYPE 'AMP' C_R	to plot amplitudes.
> OPCODE 'ALST' C_R	to plot both polarizations in each plot; selected IFs are plotted separately.
> INEXT 'CL' ; INVER <i>m</i> C_R	to choose CL table version.
> DOBLANK 1 ; DOTV 1 C_R	to plot on the TV screen including failed solutions; otherwise, create plot extension files.
> NPLOTS 5 C_R	to plot 5 antennas/IFs per page.
> DO3COL 2 C_R	to separate the calibration sources by color.
> GO C_R	to run the program.

SNPLT can also be used to plot quantities from other tables generated by the calibration process including the contents of TY ("system temperature"), and PC ("phase-cal") tables. **OPTYPE** = 'MULT' allows you to compare values of two or more parameters, plotting them at the same time in separate panels. Task **SNIFS** is similar and may be used to compare values across IFs.

9.4.6 COHER

The task **COHER** can be used to determine the coherence time in a *uv* data set broken down both in time and by antenna and baseline. The coherence time is estimated by comparing vector and scalar averaged amplitudes over increasing time averaging intervals. Averaging is not performed over source scan boundaries. The coherence time is defined as the averaging interval over which the ratio of vector and scalar amplitudes falls below a pre-assigned level. This can be set under user control. In addition, data that fall below a specified signal-to-noise ratio can be excluded from the coherence time estimates. Provision is made for selection by source name, time range, IF channel, antenna and frequency ID, and the ability to average over a subset of individual frequency channels. The input parameters to task **COHER** take the form:

> TASK 'COHER' ; INP C_R	to review the inputs.
> INDISK <i>n</i> ; GETN <i>ctn</i> C_R	to specify the <i>uv</i> input file.
> APARM 0 C_R	SNR cutoff 5; vector to scalar cutoff 0.8.
> TIMERANG 0, 10, 5, 0, 0, 10, 15, 0 C_R	time range selection.
> BIF 5 ; BCHAN 1 ; ECHAN 8 C_R	IF and channel selection.
> SOURCES 'DA193' , ' ' C_R	source selection.
> FREQID 1 C_R	Frequency ID selection.

> **GO** C_R to run the program.

Be warned that **COHER** can take quite a long time to run. A simpler, though less rigorously correct method for determining coherence intervals is to examine the data on different baselines using **EDITR**.

9.4.7 FRPLT

A preliminary examination of the coherence of individual scans or time segments in the data can also be performed using task **FRPLT**, which allows time series or fringe-rate spectra to be plotted for one or more baselines and time intervals. This task allows data to be selected by the usual criteria, including time range, source name, and frequency parameters, amongst others, and will then plot the individual time series in amplitude and phase or the associated fringe-rate spectrum. Provision is made for averaging over frequency channels within each IF, for varying degrees of padding in the **FFT**, and for division by the pseudo-continuum average “channel zero” before plotting. Typical input parameters to **FRPLT** are:

> TASK 'FRPLT' ; INP C _R	to review the inputs.
> INDISK <i>n</i> ; GETN <i>ctn</i> C _R	to specify the input file.
> SOURCES 'DA193', ' ' C _R	to select a source.
> TIMERANG 0, 10, 5, 0, 0, 10, 15, 0 C _R	to select a time range, strongly recommended.
> SOLINT 2 C _R	to plot fringe-rate spectra for every 2-minute interval in TIMERANGE.
> NPLOTS 6 C _R	to do 6 plots per page.
> STOKES 'LL' C _R	to select a single Stokes.
> BIF 1 ; EIF 0 C _R	to select the IFs included.
> BCHAN 3 ; ECHAN 12 C _R	to select the range of frequency channels averaged within each IF.
> ANTENNAS 3, 4 C _R	to select baseline 3-4, 3-5, and 4-5.
> BASELINE 4, 5 C _R	to select baseline(s) plotted in the familiar way.
> DOCAL -1 ; DOPOL -1 C _R	to apply no continuum calibration.
> DOBAND -1 C _R	to do no bandpass calibration.
> APARM(1) 1 ; APARM(2) 0 C _R	integration time 1 sec; self-scale the plots.
> APARM(7) 0 C _R	plot fringe-rate spectra with no padding.
> BPARM 0 C _R	to do no division by “channel zero.”
> DOTV 1 C _R	to plot directly on the TV device.
> GO C _R	to run the program.

The underlying time series in amplitude and phase can be plotted by setting **APARM(7)=1**; otherwise the fringe-rate spectrum is plotted. Note that the baseline(s) shown are selected with **ANTENNAS** and **BASELINE** in the usual way and a time range must be selected with **TIMERANG**. **APARM(1)** sets the integration time to be used before doing the **FFT** over the selected time range. **SOLINT** may be used to break the time range into intervals. Separate plots are produced for each IF, baseline, and time interval.

9.5 Calibration strategy

If you have multiple frequency IDs in your data, you may want to separate the data for different **FREQID** before performing any calibration. Use **UVCOP** to do this and take advantage of the opportunity to delete data flagged by the correlator with **FLAGVER=1**. You no longer need to re-run **INDXR** on the output files. Again there is a procedure to do this for you:

- > **RUN VLBAUTIL C_R** to acquire the procedures; this should be done only once since they will be remembered.
- > **INDISK n ; GETN ctn C_R** to specify the input file.
- > **CLINT Δt C_R** to set the CL table interval to Δt minutes (see discussion above in § 9.3.1.1).
- > **INP VLBAFIX C_R** to review the inputs.
- > **VLBAFIX C_R** to run the procedure.

VLBAFIX will normally be run after loading the data.

We can now begin the process of calibrating VLBI data. As the calibration process proceeds, both amplitude and phase corrections are incorporated into the CL tables. VLBI correlator output is in terms of dimensionless “correlation coefficients.” To convert to Janskys, large amplitude correction factors have to be entered into the CL tables. In addition, phase correction factors must be entered into the CL table to correct for phase offsets and ramps as functions of frequency and time. These corrections must be made so that the data can be averaged over frequency and time without loss of coherence. The process of determining the phase corrections is known as “fringe-fitting” in VLBI; see § 9.5.7.

In order to calibrate the absolute phase of their data, VLBI users use either “phase-referencing” or “self-calibration”. If the self-calibration route is chosen, then the phase derivatives with respect to time and frequency are calibrated and the absolute phases are normally left uncalibrated. Then self-calibration methods are then used to generate images (see § 9.7). For phase-referencing, an absolute-phase calibration is done using an external calibrator. Phase-referencing in VLBI is similar to, but slightly more complicated than, phase calibration on the **VLA**. In general terms, phase-referencing VLBI data is accomplished by similar methods as used for **VLA** data in **AIPS**; be sure to read § 9.5.1.2 and § 9.5.7.4 for details on how to calibrate phase-referenced observations.

For astrometric data reduction methods in **AIPS** the reader is referred to the guide to **AIPS** astrometric data reduction available from within **AIPS** by typing **HELP ASTROMET**.

Optimum fringe-fitting results are obtained if amplitudes are calibrated first, since, in this case, the data will be weighted appropriately (the **AIPS** task **FIXWT** may be used to adjust the weights in the data to reflect the scatter of the actual data). We therefore describe the process of amplitude calibration first in § 9.5.4.6. Then, in § 9.5.7, we describe the calibration of residual phase using “fringe-fitting” techniques. Note however, that for observations of strong sources or observations using only VLBA antennas (where, particularly at centimeter wavelengths, the sensitivities on all baselines are roughly the same), the order of the two calibration steps may be reversed.

9.5.1 Incremental calibration philosophy

The general strategy adopted by **AIPS** for calibration is, starting with the lowest version of the CL table, to incorporate step-by-step amplitude and phase corrections for a number of different effects. At each stage either an existing CL table is modified or a new version is created from a lower version by a task which applies a certain type of calibration. Note that the actual visibilities are not changed until you are satisfied that you have the best possible calibration file; at this point the task **SPLIT** can be used to apply the calibration information of the best CL table to the data. However at each point along the way the effect of a particular CL table on the data can be viewed using **POSSM** or **VPLOT** by setting **DOCAL**=1 and **GAINUSE** equal to the chosen CL table. Since many CL tables may be produced in the course of calibrating a VLBI data set it is important to keep a note of which effects are included in each one. Ideally one should delete CL tables which are judged incorrect and ensure that the accumulated corrections lie in the highest numbered CL table. It is suggested that version 1 of the CL table, as produced by **FITLD**, be copied to CL version 2 using **TACOP** before any calibration is begun, and that CL version 2 be used as the starting point in the calibration sequence.

An effort is made within *AIPS* to insure that CL version 1 is not deleted inadvertently. If this does occur however it can be re-generated using task **INDXR** (described in § 9.3.1.6). (Note that **INDXR** cannot re-generate some types of information, *e.g.*, the phase-cals inserted by the MKIII correlator so that it is important to try to preserve the first CL table.) The task **TASAV** can be used to back up your tables by copying all of them to a dummy file containing no data. This can be used to save a “snapshot” of the tables at various points in your data processing for insurance purposes. The tables can be copied back into your data file if necessary using **TACOP**.

You can also use the verb **HINOTE** to add comments into the history file. This can be very useful for check-pointing progress during calibration.

When you are ready to apply the calibrations, you run either **SPLIT** or **SPLAT**. Both tasks can average data in the spectral domain if appropriate. **SPLAT** can also time-average the data and produces multi-source data sets on output if requested.

9.5.1.1 Smoothing and applying corrections in SN and CL tables

The various stages of calibration described below produce SN tables which are then used to create CL tables using **CLCAL**. The ancillary tasks **SNCOR**, **SNSMO**, **SNEDT**, and **CLCOR** can be used to modify the SN and CL tables directly. It is important to choose the proper methods of interpolation in these tasks.

SN tables can be smoothed using tasks **SNEDT** and **SNSMO** before being used to update CL tables using **CLCAL**. **SNSMO** uses superior smoothing methods to those available in **CLCAL** and should always be used to do any smoothing of VLBI data, *i.e.*, data with non-zero delays and rates. The adverb **DOBLANK** now controls which data are actually altered by the smoothing; use it carefully.

Typical inputs for **SNSMO** would be:

> TASK 'SNSMO' ; INP C_R	to review the inputs.
> INDISK n ; GETN ctn C_R	to specify the input file.
> SOURCES '' C_R	to modify the solutions for all sources.
> SELBAND -1; SELFREQ -1; FREQID 0 C_R	to do all frequency IDs.
> BIF 0; EIF 0 C_R	to include all IFs.
> TIMERANG 0; ANTENNAS 0 C_R	to include all times and antennas.
> SUBARRAY 0 C_R	to select the first subarray; NB, SNSMO works only on one subarray at a time.
> SAMPTYPE 'MWF' C_R	to use the median window filter method.
> SMOTYPE 'AMPL' C_R	to smooth amplitudes only.
> BPARM 0.5,0 C_R	to use a 30-minute filter time for amplitude.
> DOBLANK 1 C_R	to replace blanked values with interpolated values <i>only</i> .
> CPARM 0.5, 0, 0, 0, 0, 0.02, 0 C_R	to set ranges of allowed values.
> INVER snin ; OUTVER snout C_R	to read in the SN table version <i>snin</i> and to write SN table version <i>snout</i> (which should be a new table).
> REFANT 0 ; BADDISK 0 C_R	to keep the current reference antenna and to allow all disks to be used for scratch files.
> INP C_R	to check the inputs.
> GO C_R	to run the task.

Typical inputs for **CLCAL** would be:

> TASK 'CLCAL' ; INP C_R	to review the inputs.
---	-----------------------

9.5. Calibration strategy

9. Reducing VLBI Data in AIPS

> INDISK <i>n</i> ; GETN <i>ctn</i> C_R	to specify the input file.
> CALSOUP '' C_R	to use all corrections in the SN table.
> SOURCE '' C_R	to apply corrections to all sources.
> OPCODE 'CALP' C_R	to apply SN tables to a CL table.
> INTERPOL '' C_R	to use linear vector interpolation ('2PT').
> SAMPTYPE '' C_R	to do no further smoothing of the merged SN tables..
> SNVER <i>snin</i> ; INVERS 0 C_R	to select the one SN table containing solutions to be interpolated.
> GAINVER <i>clin</i> C_R	to select the CL version to which solutions are to be applied.
> GAINUSE <i>clout</i> C_R	to select the output CL version, containing updated calibration information.
> REFANT 0 ; BADDISK 0 C_R	to try to use a single reference antenna if possible during all steps of calibration.
> INP C_R	to check the inputs.
> GO C_R	to run the task.

Note well that **SNVER**=0 means here to combine the solutions from all SN tables, **GAINVER**=0 means to apply the solutions to the highest numbered CL table and **GAINUSE**=0 means to write a new CL table. If two SN tables contain two similar attempts at finding corrections and **SNVER**=0 then, effectively, **CLCAL** will be inconsistent in applying the solutions from these two tables. Basically, **CLCAL** simply concatenates all SN tables and then merges apparently identical records (same time and antenna) to eliminate blanked solutions and to complain about otherwise non-identical solutions.

The parameters **INTERPOL** and **SAMPTYPE** allow the user to choose between several different methods of smoothing the SN files followed by interpolation to the times in the CL table. Use **EXPLAIN CLCAL** C_R to view all the options. The default interpolation option is **INTERPOL** = '2PT', in which the SN table is linearly interpolated between the measurements in the SN table. Using **SAMPTYPE** = 'BOX' causes the SN table to be smoothed with a boxcar function before being interpolated onto the CL table. The smoothing times for delay, rate etc. are specified in parameter **BPARM. DOBLANK** controls how both failed and good solutions are handled when smoothing. **DOBLANK** ≥ 0 replaces failed solutions with smoothed ones, while **DOBLANK** ≤ 0 replaces good solutions with smoothed ones. However it is recommended that SN smoothing be done prior to **CLCAL** using task **SNSMO**. When smoothing delays, use the new **SMOTYPE** = 'VLDE' method whenever the IFs have been grouped to find single solutions (**APARM(5)>0** in **FRING**) across multiple IFs. Be sure to set **NPIECE** to indicate how many groups of IFs were used in **FRING**, e.g., 1 for a multi-band delay, 2 for a delay for IFs 1 — $N_{if}/2$ and a delay for IFs $N_{if}/2 + 1$ — N_{if} .

With good quality data, the **INTERPOL** = 'AMBG' option should work well. Note, however, that this option uses the SN solutions immediately before and after a CL entry to make the interpolation and it uses any SN solution found for any source specified in **CALSOUP**. Therefore, if **CALSOUP** is left blank (allowing all sources) and delay and rate solutions were significantly different for different sources, then inappropriate solutions may be applied for a few minutes before or after a source change.

One way of avoiding this problem is to run **CLCAL** with **INTERPOL** = 'AMBG' several times, once for each source, setting both **SOURCE** and **CALSOUP** to the name of the desired source with all other inputs remaining unchanged. Another way of avoiding the problem is to use **INTERPOL** = 'SELF'. In this option, only solutions found on a given source are used to calibrate that source and the SN table entries closest in time for that source are used with interpolation. This is not as good as doing multiple runs with the **INTERPOL** = 'AMBG' option because there can be jumps in phase at points equidistant from two SN table entries.

If there are bad SN solutions, **INTERPOL** = 'POLY' is used to fit a polynomial to the rate solutions and then integrate this polynomial to determine the phase corrections to be entered into the CL table.

A final note on **CLCAL**. It is sometimes the case that *a priori* information is not available for all antennas in

a single format. For example, you may have system temperature information for VLBA antennas in your SN version 2 table and for non-VLBA antennas in SN version 3. You can merge this information by running **CLCAL** twice with the same **GAINVER** and **GAINUSE**; each time you should explicitly set the SN version number and list the antennas to be processed using the **ANTENNAS** adverb. If you leave the **ANTENNAS** adverb blank, the final CL table will contain information only for antennas present in the last SN table processed. Note that you must use **OPCODE** = '**CALI**' when building up a CL table in pieces.

9.5.1.2 Running CLCAL for phase referencing observations

In particular, this example illustrates how to set the inputs for **CLCAL** for the specific case when phase corrections determined for the cal source 'J1636-16' are to be transferred to the target source 'P1643-12' in a phase referencing experiment:

> TASK 'CLCAL' ; INP C _R	to review the inputs.
> INDISK <i>n</i> ; GETTN <i>ctn</i> C _R	to specify the input file.
> CALSOUR 'J1636-16', '' C _R	to use the corrections determined for the cal source.
> SOURCE 'J1636-16', 'P1643-12', '' C _R	to apply corrections to both the cal source and the target source.
> OPCODE 'CALP' C _R	to apply SN tables to a CL table passing all data.
> INTERPOL 'AMBG' C _R	to use linear vector interpolation with no SN table smoothing and simple phase ambiguity removal. See above for more discussion of INTERPOL .
> SAMPTYPE ''; BPARM 0 C _R	to clear smoothing parameters.
> SNVER <i>snn</i> C _R	to select the SN table containing solutions to be interpolated.
> GAINVER <i>cln</i> C _R	to select the CL version to which solutions are to be applied.
> GAINUSE <i>clout</i> C _R	to select the output CL version, containing updated calibration information.
> REFANT 5 ; BADDISK 0 C _R	Use a single reference antenna if at all possible during all steps of calibration.
> INP C _R	to check the inputs.
> GO C _R	to run the task.

It's a good idea to always apply the calibration information to both the cal and target sources when running **CLCAL** for phase-referencing observations. This allows you to monitor the cal source data to check the progress of the phase calibration procedure.

9.5.2 Processing observing log and calibration information

As of 1 April 1999, the VLBA correlator provides calibration transfer information, as described in § 9.3.1.2 for VLBA antennas. Experiments correlated after November 2003 also have full calibration transfer for the VLA, the GBT, Arecibo and the Bonn 100m. Consequently, **you can skip § 9.5.2 entirely** unless you have data from non-VLBA telescopes (*e.g.*, the VLA before November 2003, Space, other European telescope) or other correlators or you wish to process the log files manually for other reasons.

This section describes the processing of external calibration information, as supplied in ASCII log files. The information that may be used by *AIPS* includes Tsys or related total power measurements, edit flags as written by the tracking stations or on-line monitor control system, weather information, and pulse-calibration data. These external data can be read into *AIPS* by tasks **ANTAB**, **UVFLG**, **APCAL**, and **PCLOD** respectively, as described in § 9.5.2.3, § 9.5.3, § 9.5.4.6 and § 9.5.4.3.

You should have received information about where to obtain your calibration data. VLBA calibration log files may be obtained by ftp as described below. Similar calibration files for other participating antennas and VSOP should be obtained from the appropriate sites.

9.5.2.1 Automatic formatting of VLBA and VLBA-like log files

For VLBA antennas, the external calibration file for a given experiment can be downloaded from

<http://www.vlba.nrao.edu/astro/VOBS/astronomy/mmmyy/xxxxxcal.vlba.gz>

where *mmmyy* is the month and year (e.g., aug03) and *xxxxx* is the project code (e.g., bz199). The calibration file is named as above and is in GNU-zipped format. Gain curve information can be obtained from the same web site in the astronomy directory (*i.e.*, two directories up from the project directory). The gain curve should be concatenated to the external calibration file.

The external calibration file, if suitably close to the standard VLBA format, concatenated with the gain curve file, can be automatically subdivided and re-formatted to comply with AIPS requirements using task **VLOG**. Typical inputs would be:

- | | |
|--|---|
| > TASK 'VLOG' ; INP C _R | to review the inputs. |
| > INDISK <i>n</i> ; GETN <i>ctn</i> C _R | to specify the FITLD output file. |
| > SUBARRAY 1 C _R | to select the required subarray. |
| > CALIN 'FITS:bz199cal.vlba' C _R | to specify the input external calibration file. |
| > OUTFILE 'FITS:BZ199' C _R | to define the directory and prefix for the output files. |
| > FQTOL 1000 C _R | to set the tolerance for frequency match in kHz; one channel width is recommended. |
| > PRTLEV 0 C _R | to limit output; in particular to avoid echoing the calibration file to the screen. |
| > GO C _R | to run the program. |

A sequence of output text files will be created in the specified (\$FITS here) directory named BZ199.* with suffixes:

1. **.TSYS**: Tsys calibration data, including gain curves, suitable for direct use by **ANTAB**. An INDEX record is constructed for each frequency ID if possible. If no match can be made to the FQ data, a warning message is printed and the INDEX keyword is omitted; the INDEX keyword must then be inserted by hand (see the **HELP** file for **ANTAB**). Gain curve entries for the frequency and time range in the *uv* data are copied to this output file. It is recommended to insert Tsys values at the end of scans immediately before source changes to avoid interpolation problems for sources of greatly differing flux density or use **INTERPOL** = '**SELF**' in **CLCAL**. Occasionally spurious data from previous observing runs or system startup files will end up in the **.TSYS** file and must be edited out by hand.
2. **.FLAG**: Flag data, suitable for direct use by **UVFLG**. In the older format "antennas=vlba_xx" the appropriate antenna and day numbers are inserted.
3. **.WX**: Weather data, as used by **APCAL** if performing an opacity solution. This file is altered to conform with **APCAL** requirements (e.g., WEATHER keyword) and lines with bad entries (*) are commented out.
4. **.PCAL**: Pulse-calibration data, for input to **PCLOD**. No editing is performed.

Files with suffixes **.SCAN** and **.MKIII** contain scan summaries and MkIII information and are for information purposes only.

9.5.2.2 Manual formatting of log files

Partitioning of the calibration file can and must be done by hand if the calibration file format is sufficiently distinct from the standard VLBA format or, possibly, if it contains multiple frequency bands. If your observation used non-VLBA antennas, you will need to edit the calibration text file manually to add any log information supplied for these antennas. The necessary steps are as follows:

Extract the flagging and Tsys information from the calibration text file. You will also need to append gain curves to the Tsys file. Try [EXPLAIN ANTAB](#) to see an example file in the proper format. The parameter TIMEOFF should be set to zero for each station since both flag information and data are stored with UTC times. The keyword DTIMRANG is also supported which pads each flagged time interval to insure that even very short flag intervals are applied.

While older VLBA format calibration files were supplied with the [ANTENNA](#) keyword, newer VLBA format calibration files are supplied with the [ANT_NAME](#) keyword. In the former case, the file should be edited to insert the antenna numbers as listed in the AN table (use [PRTAN](#) on your *AIPS* file to find these) and the absolute day numbers must be replaced by relative day numbers with respect to the *AIPS* reference date. In the latter case, no adjustments to either day numbers or antenna numbers are necessary.

9.5.2.3 Loading calibration log information

The calibration information in the external text files such as Tsys and gain curve measurements are read into TY and GC tables using [ANTAB](#). These tables are then used by [APCAL](#) to generate an amplitude solution (SN) table, allowing an optional solution for atmospheric opacity. The user is advised to read the [ANTAB](#) help file closely and check the syntax of the text file carefully.

The INDEX keyword is used to assign the tabulated Tsys data to individual *AIPS* IF channels and polarizations. Up-to-date information on the usage of the INDEX keyword may be found by typing [EXPLAIN ANTAB](#). Be careful to match up the proper polarization labels for the tabulated Tsys information. The frequency and polarization association for each IF channel in the *AIPS* file can be compared (use [LISTR](#) with [OPTYPE = 'SCAN'](#)) with that at the head of the calibration text file.

The CONTROL group at the head of the calibration file is used only to specify a default index mapping. If the IF channel orders in the calibration file and the *uv* file are identical it is not required.

Source flux densities are not specified in the [ANTAB](#) input file. If source flux densities are required by [APCAL](#), the source (SU) table will be searched. Use [SETJY](#) to insert flux densities if necessary.

The parameter TIMEOFF in the input file adds a time offset to the all entries. Non-VLBA stations sometimes measure the system temperature between, rather than during, scans causing [ANTAB](#) to be unable to match the measurements with the source and frequency ID. The [ANTAB](#) input parameter [OFFSET](#) serves the same purpose, but is more successful since the scan times are expanded at both ends.

[ANTAB](#) permits specification of IF-dependent and tabulated gains; the format description may be found by typing [EXPLAIN ANTAB](#).

[ANTAB](#) can be run multiple times to append to the same TY and GC tables. Also, calibration files from separate antennas (*e.g.*, [VLA](#)) which have Tsys data tabulated in a different format can be concatenated and processed in one run. In this case the INDEX keyword must be specified for each antenna to fix the data format.

Note that [ANTAB](#) will (usually) ignore calibration data for which there are no corresponding *uv* data (see the help file for [ANTAB](#)). There is one exception however: calibration data for an antenna that does not appear

in the AN table will cause **ANTAB** to fail. If **ANTAB** quits under such circumstances, you have two choices. You can edit the calibration text file, removing all reference to the missing antennas; or you can use the input adverb **SPARM** to specify explicitly the names of antennas for which there are calibration data, but which do not appear in the AN table.

*** *The use of **SPARM** is no substitute for careful inspection of the calibration text files.* ***

Having created the input text file, typical inputs for **ANTAB** would be:

- > **TASK 'ANTAB' ; INP C_R** to review the inputs.
- > **INDISK n ; GETN ctn C_R** to specify the input file.
- > **CALIN 'MYVLB:BC25CAL.VLBA' C_R** to specify the text file.
- > **SUBARRAY 1 C_R** to select subarray one.
- > **TYVER 0 ; GCVER 0 C_R** to create new TY and GC tables.
- > **BLVER 0 C_R** to create new BL table for any specified baseline factors.
- > **PRTLEV 1 C_R** to select print level.
- > **GO C_R** to run the program.

PRTLEV = 2 will echo the calibration file as it is processed, which can be useful in locating format errors.

9.5.2.4 Generating VLA amplitude calibration with VLAMP

Before February 2013, the **VLA**'s *a priori* calibration was done with calibration files, named *xxxxxcal.y.gz* (stored in gzipped format). These were obtained from the same server and disk directory as for VLBA files. This **VLA** file started with an explanatory preamble, including minor editing instructions. See § C.11 for more detailed instructions.

After February 2013 and the advent of the phased-**EVLA**, either an **ANTAB** style file was provided to the observer or the calibration information is in the data file. This calibration information is generated using the **EVLA** switched power (SY table, see § E.6), the known gain curve (GC table), and the first CL table using the task **VLAMP**. Sometimes there are problems with the values in the SY, GC, and/or CL tables and the observer might want to recreate the amplitude calibration of the phased-**EVLA** after editing one or more of them. To do this the phased-**EVLA** only data (not the VLBI data) must be loaded into **AIPS** with **BDF2AIPS**, see § E.1. Then the SY and/or GC tables can be edited. For instructions on how to edit or smooth the SY table see § E.6. Then run **VLAMP**, which will create an output text file that can be loaded into the VLBI data with **ANTAB**. Typical inputs for **VLAMP** would be:

- > **TASK 'VLAMP' ; INP C_R** to review the inputs.
- > **INDISK n ; GETN ctn C_R** to specify the input file.
- > **FLAGVER 1 C_R** to apply flags to the input tables.
- > **INVER 1 C_R** to specify the SY table.
- > **GAINVER 1 C_R** to specify the CL table.
- > **IN2VERS 1 C_R** to specify the GC table.
- > **OUTTEXT 'filename' C_R** to specify the name of the output text *filename*.
- > **GO C_R** to run the program.

The **RUN** file and procedure named **DOVLAMP** can be used to run the full process, reading the data into **AIPS** with **BDF2AIPS**, smoothing and applying the SY tables with **TYSMO** and **TYAPL**, correcting standard calibration source amplitudes with special usage of **CALIB** and **CLCAL**, and finally running **VLAMP** to produce the needed file for **ANTAB**.

9.5.3 Data editing

Before proceeding to calibrate, you should first flag any obviously bad data. In summary, initial editing is based on the flagging information supplied by the on-line antenna monitor systems, which is applied using **UVFLG**. This information may be extracted to a **.FLAG** file as outlined in the previous section or subsequent editing based on the station report logs, or elevation limits can also be performed using **UVFLG**. Finally, graphical editing tasks such as **EDITR** and **IBLED** may be used for interactive baseline-based editing. Until the data are converted into single-source data format, flagging information is stored in the FG table instead of being used to discard data directly. Flag tables may also be used with single-source files (at least with all tasks offering the **FLAGVER** adverb). One may also undo a flag operation using **OPCODEs** '**UFLG**', '**REAS**', and '**WILD**' in **UVFLG**. Note that this operation works *only* when the operation being undone is in the current flag table. (Note that many tasks delete fully flagged data when copying a data set.)

To edit *uv* data by reading a text file listing periods of known errors (*e.g.*, the **.FLAG** text file created by **VLOG**) run **UVFLG** with the following inputs:

> TASK 'UVFLG' ; INP C _R	to review the inputs.
> INDISK <i>n</i> ; GETTN <i>ctn</i> C _R	to specify the input file.
> SOURCES " C _R	to flag all sources, which is usually desired.
> SUBARRAY 0 C _R	to select the required subarray, all in this case.
> FREQID -1 C _R	to flag all frequency IDs.
> OUTFGVER 2 C _R	to specify the output flag table; use 2 <i>only</i> if you copied FLAGVER 1 to 2 as suggested in § 9.3.1.2.
> INTEXT 'MYVLB:BC25CAL.FLAG' C _R	to specify the input text file.
> GO C _R	to run the program.

This will generate a FG table with entries read from \$MYVLB:BC25CAL.FLAG.

To edit *uv* data based on elevation limits, **UVFLG** can be used with input parameters:

> TASK 'UVFLG' ; INP C _R	to review the inputs.
> INDISK <i>n</i> ; GETTN <i>ctn</i> C _R	to specify the input file.
> SOURCES 'DA913', '' C _R	to select one source.
> SUBARRAY 1 C _R	to select subarray one.
> ANTENNAS 3, 4, 5, 8 C _R	to flag only certain antennas.
> APARM 0, 0, 0, 0, 10 C _R	to flag data between elevations 0 and 10 degrees (APARM (4) to APARM (5)), with no flagging on amplitude or weight.
> OUTFGVER 2 C _R	to specify the output flag table.
> GO C _R	to run the program.

This will generate FG table flagging time ranges that fall between the specified elevation limits.

Note that in both of these examples, the flagging information is incorporated into a FG table instead of “irrevocably” deleting your data. In order to apply these flags, you must set **FLAGVER** to the appropriate FG table version.

Another editing task which may be useful is **QUACK**, which can edit the selected portion of the scan at its beginning and/or end. This might be needed because (at non-VLBA antennas) the telescopes were still slewing or system temperature measurements were being made. The first 20 seconds or so of a scan, for baselines to the **VLA**, are often unusable while the **VLA** correlator phases up at the new source position. The first second or so after a scan may also need to be flagged (**OPCODE** 'TAIL') since some antennas may not leave the previous source promptly, leaving bad data marked good. **QUACK** can be used to flag specific antennas for these and other reasons.

Tasks `EDITA` and `EDITR` can be used to inspect and edit the data interactively. `IBLED` is no longer of much use except for its ability to display the degree of coherence in the data. These tasks are similar in many respects to `TVFLG`, but are more suited to interferometers with small numbers of baselines. It is also possible to use `TVFLG` to perform your editing although the TV display is somewhat confusing on sparse arrays of data, especially if there is significant source structure. Read § 4.4 through § 4.4.2 and § 5.5 for more details on the editing of data. Also `VPLOT` may be used to edit data which deviates excessively from the mean amplitude over a specified averaging interval; see `HELP VPLOT` C_R for details. Task `WIPER` is a dangerous but powerful editing tool as well.

The task `DEFLG` generates a flag table to delete data having coherence less than a user-specified limit. It must be run only on sources that should be strongly coherent although it may be used to flag data from other sources in between the strong-source scans. The task `SNFLG` is also used to flag data whenever the phase jumps in an SN or CL table are excessive on a baseline-by-baseline basis.

The VLBA correlator may produce a lot of samples which it knows to be bad and which are flagged by the transferred flag table. For this reason alone, or if you also flag a noticeable fraction of the data set, you may wish to run `UVCOP` to discard the flagged data to conserve disk space and processing time.

9.5.4 Amplitude and instrumental delay calibration

We now advocate a new amplitude calibration strategy base on VLBA Scientific memo #37 (Walker 2015). This strategy interleaves classic *a priori* calibration with instrumental delay and bandpass calibration to improve the amplitude calibration of data from the new Roach Digital Backend (RDBE) on the VLBA (see the VLBA Observations Status Summary for a description of the RDBE and VLBA Scientific Memo #37 for a discussion of amplitude problems with the RDBE). With data from before the RDBE you can use either the old or new strategy.

9.5.4.1 Parallactic angle correction

The RCP and LCP feeds on each antenna will rotate in position angle with respect to the source during the course of the observation for alt-az antennas (which probably constitute a majority of the antennas in your observation). Since this rotation is a simple geometric effect, it can be corrected by adjusting the phases without looking at the data. This correction must be performed before any phase calibration which actually examines the data is executed. This correction is important for polarization and phase-referencing observations, so it should probably be applied to all cases.. Task `CLCOR` is used for this purpose; see § 9.5.4.1.

There is a procedure which assists you in running `CLCOR` to correct phases for parallactic angle:

- > `RUN VLBAUTIL` C_R to acquire the procedures; this should be done only once since they will be remembered.
- > `INDISK n ; GETN ctn` C_R to specify the input file.
- > `SUBAR ss` C_R to select the subarray number — only one per execution.
- > `INP VLBAPANG` C_R to review the inputs.
- > `VLBAPANG` C_R to run the procedure.

`VLBAPANG` should be run before applying any other phase corrections.

9.5.4.2 Digital sampler bias corrections for VLBA correlator data

The voltage threshold levels in the digital samplers at the antennas may differ from their optimum theoretical values and this may vary from antenna to antenna and from polarization to polarization. This sampler bias, which is usually significant only in two-bit quantization, introduces an antenna/polarization-based amplitude offset. In full polarization observations this appears as an amplitude offset between RR and LL. The cross-correlation amplitudes may be corrected if the auto-correlation spectra have been measured. See VLBA Scientific Memo No. 9 (1995, “Effect of digitizers errors on the cross and auto correlation response of an FX correlator”, by L. Kogan). Task **ACCOR** can be used to remove these digital sampler biases from VLBA correlator data, if **FITLD** was run with **DIGICOR** = 0, 1, or 2. **ACCOR** corrects these offsets by examining the autocorrelation spectra. Since **ACCOR** computes the necessary correction by examining the total-power spectra, it must be run immediately after **FITLD** in the sense that nothing else should be run that actually modifies the total-power spectra directly. Although **ACCOR** ignores any SN or CL tables that are present, it is essential to correct for the sampler biases before performing any *a posteriori* calibration, e.g., fringe-fitting or self-calibration. If the **ACCOR** gain factors deviate from unity, there may be overall scaling or *b*-factor errors. For 2-bit quantization, however, the amplitude offsets for the VLBA correlator are typically of order 5 – 10%, but values as high as 20% have been observed.

For one-bit quantization, no significant sampler bias correction is expected. Nonetheless, it is recommended that **ACCOR** be run on one-bit data as a consistency check. **VLBACCOR** is a procedure which runs **ACCOR**, smooths the results with **SNSMO**, and applies the solutions using **CLCAL**. Typical inputs would be:

- > **RUN VLBAUTIL C_R** to acquire the procedures; this should be done only once since they will be remembered.
- > **INDISK n ; GETN ctn C_R** to specify the input file.
- > **FREQID ff ; SUBAR ss C_R** to select the frequency ID and subarray numbers — only one of each per execution.
- > **INP VLBACCOR C_R** to review the inputs.
- > **VLBACCOR C_R** to run the procedure.

VLBACCOR will normally be run after “fixing” the the data with **VLBAFIX** (if needed). Use this procedure only on data from the VLBA correlator.

If you do not want to use **VLBACCOR**, typical inputs to **ACCOR** for this correction would be:

- > **TASK 'ACCOR' ; INP C_R** to review the inputs.
- > **INDISK n; GETN ctn C_R** to specify the input file.
- > **TIMERANG 0 C_R** to select data from all times.
- > **SOLINT 2 C_R** to set the solution interval for sampler corrections (min).
- > **GO C_R** to run the program.

The correction factors were expected to be fairly stable over time, but they have been found to vary over times less than an hour. With a solution interval of a few minutes, such as the two minutes indicated here, it is well to examine the solution (SN) table generated by **ACCOR** using **SNPLT** for any bad points or inconsistent values. One approach is to inspect the SN table and then run **SNSMO** with clipping to get rid of discrepant points. Alternatively, the interactive table editing task **SNEDT** can be used.

9.5.4.3 Instrumental phase corrections

If you run **POSSM** on a short (\approx 1 minute) section of data on a strong calibrator using the inputs described in § 9.4.2 and set **DOCAL** = -1 **C_R**, you will see that each individual IF channel has its own independent phase offset and its own phase gradient against frequency. These phase offsets and instrumental “single-band delays” are caused by passage of the signal through the electronics of the VLBA baseband converters (or

MkIII/MkIV video converter units). The VLBA and MkIII systems can inject narrow band signals (“phase-cals”) into the data recorded at each antenna from which the IF channel phase offsets, and the instrumental single-band delays, can be determined.

For those antennas for which phase-cal measurements are available, task [PCCOR](#) can be used to incorporate the phase-cals into an SN table. (see § 9.5.4.3). If you have other antennas for which phase-cal measurements are not available, you can run [CLCAL](#) using the [OPCODE='CALP'](#) option to incorporate the incomplete SN table information loaded by [PCCOR](#) without throwing away data for the antennas with missing phase-cal information.

For MkIII data from the Bonn correlator, phase-cal measurements are incorporated directly into the first CL table produced by [MK3IN](#) — this is another strong reason to protect the first CL table.

If external phase-calibration data (pulse cals) are not available then directly fringe-fitting a short scan of data to measure the phase and single-band delay offsets may be applicable under limited circumstances (see § 9.5.4.4). Even when phase-calibration information is available, performing a manual phase-cal can be a good idea to confirm that the IF-dependent delays and phases have been successfully estimated and removed. One good check is to inspect the data using [POSSM](#) at times different from the time used to determine the manual phase calibration. Note that time-dependent delays may still be seen because of low elevation and ionospheric effects.

As of 1 April 1999, the VLBA correlator is capable of transferring phase-cal information directly to a PC table for some antennas. For other antennas, the phase-cal information may be read into the PC table using the task [PCLOD](#). Type [EXPLAIN PCLOD](#) for further information. If you are unsure whether your VLBI data has phase-cal information, use [IMHEAD](#) to list the extension tables and look for a PC extension.

Phase-calibration data in a PC table can be used by task [PCCOR](#) to generate an SN table which corrects for the single-band instrumental phase and delay offsets (note that [PCCOR](#) uses two phase-cals in each IF). A short calibrator scan must be specified to be used by [PCCOR](#) to resolve any 2π phase ambiguities in the phase-calibration data. The specified time range must include at least one PC table entry for each antenna appearing in the PC table. Having resolved the 2π phase ambiguities, [PCCOR](#) uses the whole PC table to calculate entries in an SN table for all times (not just the [TIMERANG](#) used to resolve the ambiguities). The procedure [VLBAPCOR](#) runs [PCCOR](#), [FRING](#) (if needed because of missing pulse-cals for some antennas), and [CLCAL](#) for you. Inputs are:

- | | |
|--|--|
| > RUN VLBAUTIL C _R | to acquire the procedures; this should be done only once since they will be remembered. |
| > INDISK n ; GETN ctn C _R | to specify the input file. |
| > TIMERANGE d1 h1 m1 s1 d2 h2 m2 s2 C _R | to specify a short scan on a calibrator. There is no default. |
| > REFANT m C _R | to select a particular reference antenna. |
| > SUBARRAY 0 | to do all subarrays. |
| > CALSOUR 'cal1', '' | to specify the calibrator source name. |
| > GAINUSE CLin C _R | to indicate the CL table with all calibration up to this point. |
| > OPCODE 'CALP' C _R | to indicate that there are antennas with no usable pulse cals; use OPCODE ' ' if all antennas have pulse cals. |
| > ANTENNAS a1 a2 a3 C _R | to solve for antennas a1, a2, a3 “manually” (using FRING). |
| > VLBAPCOR C _R | to run the procedure. |

This should be done after the sampler correction ([VLBACCOR](#)) and before the complex bandpass. The CALP option requires the data in the specified [TIMERANG](#) to include strong fringes for those antennas lacking phase-cal data.

Running the tasks individually, typical inputs to [PCCOR](#) are:

- | | |
|--|-----------------------|
| > TASK 'PCCOR'; INP C _R | to review the inputs. |
|--|-----------------------|

> INDISK <i>n</i> ; GETN <i>ctn</i> C_R	to specify the input file.
> TIMERANG 1 2 15 0 1 2 20 0 C_R	to isolate a short scan on a calibrator; <i>no default</i> .
> SNVER 0 C_R	to create a new SN table.
> INVER 1 C_R	to specify the input PC table version.
> REFANT 5 C_R	to specify reference antenna.
> SUBARRAY 1 ; FREQID 1 C_R	to set subarray and freqid.
> CALSOUR '3C345' ; C_R	to set calibrator source name.
> GO C_R	to run the program.

The resulting solution table is applied using **CLCAL**. If you are missing phase-cal information for some antennas, you must use the 'CALP' mode of **CLCAL**; this mode allows calibration information for some antennas to be incorporated into the CL table while passing other antennas through without modification. Examine the corrected data using **POSSM** to determine if the instrumental phase and delay offsets between the IF channels have been removed correctly.

To check that the applied phase-cals are valid, run **POSSM** on a short section of data containing a strong source setting **DOTAL** = 1, **GAINUSE** to the version number of the CL table containing the phase-cal calibration, and **APARM(9)** = 1 to place all IFs on the same plot. The phase as a function of frequency on each baseline should be smoothly varying, with no sharp jumps between different IF channels. There may be an overall linear gradient with frequency due to residual delay errors. Unless these conditions hold for all baselines, you should proceed to §9.5.4.4.

In 31DEC16, the DiFX correlator was given the capability of writing a large number (128 typically) of pulse cals into each IF. *AIPS* can handle this with new tasks **PCFLG** (flags PC data with a **SPFLG**-like TV display), **PCEDT** (flags PC data with a **BPEDT**-like TV graphical display), **PCPLT** (plots PC table data all antennas with one time per page or all times with one antenna per page), **PCASS** (finds AMPLITUDE bandpass), and **PCFIT** (fits delays and phases to PC data, writing an SN table of the changes in these as a function of time). **PCLUD** can now read **PCAL** files written by DiFX. **POSSM** can also now plot PC table data (**APARM(8)** = 9). This package of routines remains highly experimental and we have to demonstrate that they are scientifically useful.

9.5.4.4 “Manual” instrumental phase corrections

If your file does not have phase-cal information, or if these phase-cals do not successfully remove frequency phase offsets, you can use observations of a bright calibrator source and the task **FRING** to correct for these effects. If you attempted phase-cal calibration, it is best to avoid possible confusion by first deleting any partial or erroneous phase-cal information that already exists. Using **CLCOR**:

> TASK 'CLCOR'; INP C_R	to review the inputs.
> INDISK <i>n</i> ; GETN <i>ctn</i> C_R	to specify the input file.
> GAINVER <i>cln</i> ; GAINUSE 0 C_R	to specify which CL table to modify after copying it to a new table.
> OPCODE 'PCAL' ; CLCORPRM 0 C_R	to set phase-cals to zero.
> INP C_R	to check the inputs.
> GO C_R	to run the task.

If you have large known delay offsets you may also wish to run **CLCOR** using **OPCODE**='SBDL' to shift the center of the fringe search window.

If you have a known clock offset, as may be common for an SVLBI data set, you may wish to run **CLCOR** with **OPCODE** = 'CLOC' and **CLCORPRM** = 0, *offset*, 0, 0, 0, 0, 1, 0.

Now you can determine the manual phase correction for one or two strong calibrator scans for which all the antennas are present using **VLBAMPCL**. Task **BSCAN** may be used to select the scan(s) to be used at this stage. **VLBAMPCL** runs **FRING** and **CLCAL** once or twice, depending on whether one or two scans are used. Choose a scan with strong fringes to all antennas; if none exists, find a second scan that has strong fringes to the antennas missing from the first. Note that if you use 2 scans the **REFANT** must have good fringes in both scans. Typical inputs are:

> RUN VLBAUTIL C _R	to acquire the procedures; this should be done only once since they will be remembered.
> INDISK n ; GETN ctn C _R	to specify the input file.
> TIMERANGE d1 h1 m1 s1 d2 h2 m2 s2 C _R	to specify a short scan on a calibrator. There is no default.
> REFANT m C _R	to select a particular reference antenna.
> SUBARRAY 0	to do all subarrays.
> CALSOEUR 'cal1', ''	to specify the calibrator source name.
> GAINUSE CLin C _R	to indicate the CL table with all calibration up to this point.
> OPCODE 'CALP' C _R	to indicate that there are antennas with no fringes for the scan in TIMERANGE; use OPCODE ' ' if all antennas will be corrected by the first scan.
> TIME2 d1 h1 m1 s1 d2 h2 m2 s2 C _R	to specify a short second scan on a calibrator.
> CALSOEUR 'cal2', ''	to specify the calibrator source name for the second scan.
> ANTENNAS a1 a2 a3 C _R	to solve for antennas <i>a1, a2, a3</i> .
> VLBAMPCL C _R	to run the procedure.

If you wish to run **FRING** separately, you can determine the phase offsets/single band delays by running **FRING** on a short section of calibrator data where all or most of the antennas are present. Suitable inputs for **FRING** for this purpose are shown below; for more details of some of the **FRING** input parameters see § 9.5.7.7. Note that it is simplest to choose a single short section of data within a single scan using **TIMERANG** and to set **SOLINT** equal to the scan length so that a single solution is achieved. The interval chosen must be less than than an atmospheric coherence time, but long enough that high signal-to-noise is achieved. At centimeter wavelengths with Jansky-level calibrators, solution intervals of a few minutes will work well. For example:

> TASK 'FRING' ; INP C _R	to review the inputs.
> INDISK n ; GETN ctn C _R	to specify the input file.
> CALSOEUR '0954+658', '' C _R	to select a strong calibrator source.
> TIMERANG 0, 16, 0, 0, 0, 16, 2, 0 C _R	to select a single short scan.
> DOCALIB 1 ; GAINUSE 2 C _R	to apply the amplitude calibration from CL table 2 to the weights as well as the visibilities.
> FLAGVER 0 C _R	to apply the most recent flag table.
> SMODEL 0 C _R	to use the null (point-source at the origin) source model.
> REFANT 5 C _R	to specify a reference antenna that will give fringes to most other antennas.
> SOLINT 0 C _R	to set the solution interval in minutes; do not exceed the atmospheric coherence time.
> APARM 0; DPARM 0 C _R	to initialize FRING options to defaults.
> APARM(1)=2 C _R	to require at least 2 antennas.
> APARM(6)=1 C _R	to solve for the rate, single-band delay and phase of each IF separately.
> DPARM(1)=1 C _R	to use only 1 baseline in the initial (coarse) fringe search.
> DPARM(9)=1 C _R	to suppress fitting of rates; rates will be 0 in the SN table.
> SNVER 0 C _R	to create a new SN table.

- > **ANTWT** 0 C_R to apply no additional weights to the antennas before doing the solutions.
- > **INP** C_R to check the inputs.
- > **GO** C_R to do the fit.

If there was no single scan where all the antennas were present, you can run **FRING** again for another scan setting **REFANT** to be one of the antennas found in the first run (the same **REFANT** would be best) and **ANTENNAS** to this antenna plus all of the antennas *not* found in the first run. **FRING** will generate a new SN table each time; be careful to keep track of which SN tables you wish to use.

The phase solutions in the SN table(s) are interpolated onto a calibration or CL table using task **CLCAL** as described in § 9.5.1.1. To apply the calibrator solutions to the other sources in the data file, set **CALSOUR** to the calibrator source used when running **FRING** (in the example above, **CALSOUR** '0954+658') and set **SOURCE** = ' ' for all sources. Also, set **REFANT** to whatever was used when running **FRING**. If multiple runs of **FRING** were required, you can set **SNVER**=0 so that all SN tables are combined before being applied ; if you do this, you must first be careful to delete all SN tables except those generated by **FRING**. Or you can run **CLCAL** multiple times specifying each SN table in turn, with the specific antenna numbers, while keeping the same CL table versions.

To check the output CL table, run **POSSM** for the scan used for the **FRING** solution with **DOCAL** = 1 and **GAINUSE** = 3 (*i.e.*, the output CL table from the **CLCAL** above). The phase should be flat as a function of frequency on all baselines, although it may not be centered on zero. Run **POSSM** on another scan containing a strong calibrator to check that the assumption of constant IF phase offset holds.

9.5.4.5 Bandpass calibration

Full bandpass response calibration should be performed for all observations. It is suggested that integrating over a variable bandpass function is one of the most significant sources of non-closing errors in continuum VLBI data. By calibrating the bandpass before averaging over frequency, these effects can be avoided. In VLBA test observations, a dynamic range of 28,000:1 was achieved on the source DA193 (Briggs *et al.* 1994, VLBA Memo No 697A, "High Dynamic Range Imaging with the VLBA") after applying bandpass calibration.

Bandpass calibration is carried out using the task **BPASS** using either the auto- or cross-correlation data. The output is a BP or bandpass table. For this amplitude calibration scheme the cross-correlation (complex) bandpass must be used. The derived bandpass solutions can be plotted using **POSSM** by setting **APARM(8)=2**. The effect of applying these bandpass solutions to your data can also be viewed using **POSSM** by setting **DOBAND=1** and **BPVER**. An example of the inputs to produce bandpass spectra from the cross-correlation data would be:

- > **TASK** 'BPASS' ; **INP** C_R to review the inputs.
- > **INDISK** $n1$; **GETN** $ctn1$ C_R to select the multi-source visibility data as the input file.
- > **CALSOUR** 'BLLAC', 'DA193' C_R to specify the continuum source(s) which were observed for the purpose of bandpass calibration.
- > **DOCALIB** 1 C_R to apply calibration.
- > **SOLINT** 0 C_R to average data over whole scans before determining the bandpass.
- > **BPASSPRM** 0 C_R to use the cross-spectra.
- > **BPASSPRM(5)** 1 C_R to not divide by "channel 0."
- > **BPASSPRM(9)** 1 C_R to interpolate over flagged channels.
- > **BPASSPRM(10)** 6 C_R to normalize the amplitude solutions using the *power* solutions rather than the actual solutions which are voltages.

- > **ICHANSEL** *chanbeg, chanend, 1, IFnum, ... C_R* to set channels for entire bandwidth for normalization, if this left 0 then the inner 75% is used and can cause up to 15% error in the amplitude, so unfortunately **ICHANSEL** must be explicitly set to all the channels in the IF, for every IF.

> **GO C_R** to run the program.

Be careful with the adverb **SMOOTH**. If you smooth, or do not smooth, the data while finding a bandpass solution, then you must apply the same **SMOOTH** adverb values whenever you apply that bandpass solution to the data. The only exception is that you may smooth the data after applying the bandpass solution with **SMOOTH(1)** values 5 through 8 when you did no smoothing in **BPASS**.

This will produce a BP table containing the antenna-based bandpass functions to be applied to the data. determined. You should check your results very carefully. The BP tables can be plotted with **POSSM** or printed with **PRTAB**. Note that this task merely creates a BP table. To use this BP table, set **BPVER** and **DOBAND** as described in § 4.7.4 when running any later *AIPS* tasks.

Note, if your bandpass calibrator is not stable or strong enough during the observation (your bandpass calibrator data should have a better S/N than the data you're trying to correct with the bandpass), you could consider using the phase reference source (if there is one) and use **SOLINT** = -1 (include all scans to make one bandpass solution). If the bandpass calibrator is strong enough, but the average phase varies through the scan, then divide each record by "channel 0" by setting **BPASSPRM(5)** = -1 to adjust phase only. You should select a range of channels that have similar phases to be averaged as channel 0 using adverb **ICHANSEL**.

The normalization of the bandpass correction can be checked by running **ACSCL** after **BPASS**. This task is a version of **ACCOR** which applies the current calibration tables to the data before forcing the autocorrelations to unity. Be sure to set **DOBAND** to apply the bandpass by your preferred method. **ACSCL** will write an SN table with amplitude corrections which should be very close to one. Apply that table with **CLCAL**. However if you run **VLBAAMP**, **ACSCL** and **CLCAL** will be run for you, see § 9.5.4.6.

In 31DEC16, a new data editing task **BPEDT** appeared to flag visibility data based on bad bandpass solutions. It is a graphical editor much like **EDITR** and **EDITA** except that it displays bandpass solutions as a function of spectral channel. It is a quick way to check your bandpass table and, if necessary, to flag bad channels in your calibration source. If you do generate new data flags, you should run **BPASS** over again to generate a corrected BP table. Set **BPASSPRM(9)** = 1 to interpolate over any fully flagged channels.

9.5.4.6 Continuum amplitude calibration

After bandpass calibration is determined, the auto-correlations can be slightly offset from unity. To correct this offset **ACSCL** can be run, applying all the previous amplitude and bandpass calibration. Described below is a procedure that is part of **VLBAUTIL** called **VLBAAMP** that runs **ACSCL** as well as the final amplitude calibration steps.

The TY table can be examined using **SNPLT** with **OPTYPE='TSYS'** or '**TANT**' or **LISTR** with **OPTYPE='GAIN'**; the GC table is examined with **PRTAB**. **SNEDT** may also be used to inspect the TY table and even used to delete or smooth some of the measurements. **TYSMO** can be used to delete discrepant system temperatures and replace the bad and/or good values with time-smoothed values. Note, however, that anomalously high system temperatures may indicate possible bad data (*e.g.*, due to weather) rather than bad measurements of the system temperature. On occasion, **RFI** is some IFs can render the values for those IFs in the TY table unusable. Task **TYCOP** will copy TY data between IFs and polarizations in a variety of ways in an attempt to replace the bad data with something reasonable even if not exactly correct for that antenna and IF. Note that flagging bad *uv* data will not change the appearance of the TY table since no flags are applied in plotting this table. **EDITA** may also be used to examine the TY data interactively and, if bad system temperatures are

found, to flag the associated *uv* data. (This task does flag displayed T_{sys} when the data are flagged.) Tasks **UVCOP** and **SPLAT** will apply a flag table not only to the *uv* data, but also to the **TY** and **SN** tables. **APCAL** can then be used to derive an amplitude calibration (**SN**) table.

Atmospheric opacity becomes significant at high frequencies (≥ 15 GHz). **APCAL** can fit for an opacity correction, if needed, using weather information and system temperature. The weather information can be taken from the **WX** table by setting **INVERS** (after October 7, 2003) or loaded from disk by setting **CALIN**. In order to have **APCAL** fit for opacity set **OPCODE** to 'GRID', 'OPAC' or 'LESQ' and **DOFIT** to 1 (both of these are necessary for an opacity fit). **OPCODE** 'GRID' and 'OPAC' need an initial guess for the receiver temperature (**TRECVR**) and zenith opacity (**TAU0**). **APCAL** will estimate the initial guesses from the data if **TRECVR** and/or **TAU0** are 0. If **OPCODE** is set to 'GRID', 'OPAC' or 'LESQ' and **DOFIT** ≤ 0 then the opacity correction is applied using the provided **TRECVR**. This is a good option if you have a reliable measurement of the receiver temperature. The fits are fairly robust, but the plots that **APCAL** makes should be examined. Note: a large number of bad T_{sys} values can make the fits unreliable. **APCAL** will warn you if fits appear to be incorrect. The procedure **VLBAAMP** runs **ACSCL**, smooths the results with **SNSMO**, runs **APCAL**, and applies the solutions using **CLCAL**:

- | | |
|---|---|
| > RUN VLBAUTIL C_R | to acquire the procedures; this should be done only once since they will be remembered. |
| > INDISK n ; GETN ctn C_R | to specify the input file. |
| > FREQID ff ; SUBAR ss C_R | to select the frequency ID and subarray numbers — only one of each per execution. |
| > INP VLBAAMP C_R | to review the inputs. |
| > DOFIT 1 C_R | to enable the opacity correction; ≤ 0 disables it. |
| > VLBAAMP C_R | to run the procedure. |

VLBAAMP will normally be run after correcting the bandpass and loading any gain curves or system temperature data for non-VLBA antennas using **ANTAB**.

If you have not used **VLBAAMP** and do not want to correct for atmospheric opacity, typical inputs to **APCAL** are:

- | | |
|---|---|
| > TASK 'APCAL' ; INP C_R | to review the inputs. |
| > INDISK n ; GETN ctn C_R | to specify the input file. |
| > ANTENNAS 0 ; SUBARRAY 0 C_R | to select all antennas and subarrays. |
| > SOURCES '' ; STOKES '' C_R | to select all sources and Stokes. |
| > BIF 1 ; EIF 0 ; FREQID 1 C_R | to select all IFs of frequency ID 1. |
| > TIMERANG 0 ; OPCODE '' C_R | to select all times and use no opacity solutions. |
| > TYVER 0 ; GCVER 0 C_R | to use the latest TY and GC tables. |
| > SNVER 0 C_R | to create a new SN table. |
| > GO C_R | to run the program. |

If atmospheric opacity correction is desired, set the following inputs as well as the above:

- | | |
|--|--|
| > OPCODE 'GRID' C_R | do a grid search. |
| > DOFIT 1 C_R | to fit the opacities. |
| > INVERS 1 C_R | to use WX table 1. |
| > TAU0 0 ; TRECVR 0 C_R | to let APCAL estimate initial values. |
| > GO C_R | to run the program. |

The resulting solution (**SN**) table can be smoothed and clipped using **SNSMO** and applied using **CLCAL** as described in § 9.5.1.1. Substantial smoothing of the **TY** table (task **TYSMO**), especially for VLBA-only observations, is not generally recommended since variations of the system temperature often reflect a real response to the weather. Smoothing can be useful for data from the phased-VLA, when the amplitude calibration information reflects low signal-to-noise. For non-VLBA antennas, it is important to check with **SNEDT** that the **TY** information is associated with the correct source and **SNEDT** can be used to delete

occasional bad system temperature measurements before they are applied to the data. When opacities are fit, **APCAL** generates plots of the receiver temperature versus zenith angle and the opacity versus time. **APCAL** will warn you about bad fits, but the plots should be checked for problems with the data or the fits.

On occasion, it has been noted that plots of system temperature show unreasonable variation between IFs. This suggests that the T_{cal} values may have systematic errors as a function of frequency. **APCAL** has, in 31DEC14, an option to bring the T_{sys} values into better agreement with either their average or with a user-chosen IF, thought to be probably accurate. **APARM(6) > 0** selects this function while **APARM(7)** may be used to specify a specific IF to calibrate the others. Note that a single value is used for each antenna/IF over all times.

The DiFX correlator is now capable of correlating multiple phase-stopping positions within a single antenna beam (pointing position). In 31DEC13 there is task called **CLVLB** designed to apply corrections to a CL table (writing a new one) for the offset of phase-stopping and pointing position. The task has internal tables for the beam shape and squint of the VLBA antennas and can take user input for these parameters for other antennas. This is a new task and should be used with caution — early results with it were less than satisfying — although the corrections it applies are certainly in the right direction.

9.5.5 Spectral-line Doppler correction

Normally, when observing, you will have kept the frequency constant throughout the run for ease of observing. Therefore, although your data will have the same frequency, the center velocity of your spectrum will change with time and the spectral-line signals will wander backwards and forwards through the spectrum. To ensure that the velocity is constant throughout the data you should run **SETJY** and then **CVEL**. The VLBA correlator compensates for the revolution of the antennas relative to the center of the Earth. In this case, the only movement which **CVEL** should compensate is the rotation of the antennas together with the Earth around the Sun. This movement should be the same for all antennas and gives a smaller effect than the rotation relative to the center of the Earth. Nonetheless, **CVEL** is required for the VLBA correlator at least to be able to compare observations made at different times. Without **CVEL** such comparisons will show the velocity shift of the Earth's orbit about the Sun.

SETJY will insert the velocity information required in the SU table:

- | | |
|--|---|
| > TASK 'SETJY'; INP C_R | to review the inputs. |
| > INDISK n1 ; GETN ctn1 C_R | to select the data. |
| > SOURCE 'OH127.8' , ' C_R | to specify the line source whose velocity is to be specified. |
| > OPTYPE '' C_R | to switch off flux modification. |
| > SYSVEL -66.0 C_R | to specify the velocity of the "center" of the band in km/s. |
| > APARM 65, 0 C_R | to specify which spectral pixel is the "center" of the band, actually the pixel to which SYSVEL refers. |
| > RESTFREQ 1612e6, 231.09e3 C_R | to give the rest-frequency in Hz, e.g., that of the OH transition. Note that the two single-precision adverb numbers are summed in double precision inside SETJY . |
| > VELTYP 'LSR' C_R | to select the rest frame of the velocity. |
| > VELDEF 'OPTICAL' C_R | to define velocities by the optical convention. |
| > GO C_R | to run the program. |

Then run **CVEL**:

- | | |
|--|-----------------------------|
| > TASK 'CVEL'; INP C_R | to review the inputs. |
| > INDISK n ; GETN ctn C_R | to select the data. |
| > OUTDISK 3 ; OUTCLASS 'CVEL' C_R | to specify the output file. |

- | | |
|--|--|
| > SOURCE 'OH127.8' , , C_R | to select the source(s) to be shifted, all others will be passed un-shifted. |
| > DOBAND 1 C_R | to apply the bandpass correction — important. |
| > BPVER 1 C_R | to specify the version of BP table to use. |
| > GO C_R | to run the program. |

After applying the BP table, **CVEL** will not copy it to the output file to protect you from applying it twice. Although **CVEL** allows you to select which sources are to be shifted, the BP table, if **DOBAND** is set appropriately, will be applied to all sources found.

9.5.6 Spectral-line amplitude calibration

The calibration strategy suggested in *AIPS* for spectral-line VLBI data utilizes the total-power spectra method described in Lecture 12 of *VLBI, Techniques and Applications*, eds. Felli and Spencer, published by Kluwer Academic Publisher, 1988. The continuum method using Tsys values (see § 9.5.4.6) can also be used. The first step is to generate a so-called template spectrum. This is a high quality spectrum from the most sensitive antenna in the array that has been corrected for the effects of the bandpass filter. For example:

- | | |
|--|---|
| > TASK 'SPLIT' ; INP C_R | to review the inputs. |
| > INDISK <i>n</i> ; GETN <i>ctn</i> C_R | to specify the input file. |
| > SOURCE 'OH127.8' , , C_R | to write the program source. |
| > BCHAN 0 ; ECHAN 0 C_R | to write all spectral channels. |
| > DOCALIB -1 C_R | to avoid applying any calibration. |
| > DOBAND -1 C_R | to skip the bandpass correction since the data will have been corrected already in CVEL . |
| > TIMERANG 0 22 0 0 0 22 30 0 C_R | to select the data from a range of times when the antenna elevation was high and the source spectrum of high quality. |
| > APARM 0, 0, 0, 0, 1 C_R | to pass only self-spectra. |
| > GO C_R | to run the program. |

You should then run **ACFIT** to do a least-squares fit of the template total-power spectrum to the total-power spectra of all other antennas and to write the resulting time-dependent amplitude gain correction factors into an SN table.

- | | |
|--|--|
| > TASK 'ACFIT' ; INP C_R | to review the inputs. |
| > INDISK <i>n</i> ; GETN <i>ctn</i> C_R | to specify the input file. |
| > IN2DISK <i>n</i> ; GET2N <i>ctn</i> C_R | to specify the template file. |
| > CALSOUR 'OH127.8' C_R | to select the source to use for calibration. |
| > DOCALIB -1 C_R | to avoid applying any previous calibration. |
| > DOBAND -1 C_R | to skip the bandpass correction since it was done when CVEL was run. |
| > SOLINT <i>n</i> C_R | to average the self-spectra over <i>n</i> minutes (e.g., 10) before doing the least-squares fit. |
| > REFANT 1 C_R | to select the desired reference antenna from the template file. |
| > BCHAN 50 ; ECHAN 70 C_R | to set the range of spectral channels over which the fit is performed. |
| > BPARM 80, 120 C_R | to set up to 5 pairs of start and stop channels to use in determining the baseline polynomial to be removed from the source spectra. The order of the polynomial is specified in APARM(1) . |

- > **CPARM** 80, 120 C_R to set up to 5 pairs of start and stop channels to use in determining the baseline polynomial to be removed from the template spectra. The order of the polynomial is specified in **APARM**(2).
- > **XPARM** 45.1, 48.0, 50.1, 49.5 C_R to specify Tsys values in the first polarization for each IF for the template scan of the reference antenna. **YPARM** provides an equivalent list for the data, if any, from a second polarization.
- > **APARM** 0, 0, 50, 0, 0.72 C_R program control: **APARM**(1) and **APARM**(2) specify the orders of the polynomial spectral baseline to remove from the source and template spectra; **APARM**(3) and **APARM**(4) specify the sensitivity of the template antenna (in Jy/deg) in the first and second (if needed) polarizations; **APARM**(5) and **APARM**(6) specify the minimum and maximum relative antenna gains allowed, with defaults to allow all positive values; **APARM**(7) specifies the maximum allowed gain error, with 0 meaning all; **APARM**(8) specifies the print level, with 0 providing minimal information, 1 providing useful information on the gains determined for each antenna and solution interval and 2 giving the gory details for each fit; **APARM**(9) specifies that the fits are done after subtracting a spectral baseline (0) or without a baseline (1); and **APARM**(10) controls whether baseline-subtracted spectra are written to an output file.
- > **SNVER** 0 C_R to create a new SN table into which the solutions are to be written.
- > **GO** C_R to run the program.

ACFIT will generate an SN table, which has to be applied to the CL table. If needed, run **SNSMO** to smooth the amplitude correction factors determined by **ACFIT** before running **CLCAL**. A 30-minute smoothing interval for **SNSMO** (set using **CPARM** 0.5,0) should be sufficient.

9.5.7 Phase calibration

After carrying out amplitude calibration, the remaining calibration steps involve correcting the phase between the different integration periods. This will allow averaging of the data over both frequency and time without loss of coherence. The phase offsets may be corrected using *a priori* ‘phase-cal’ measurements if available and/or by directly fitting to the data.

After removing instrumental phase offsets from each IF, the data will in general still contain frequency and time dependent phase variations. The purpose of “fringe-fitting” is to determine these phase errors and then remove them from the data.

The primary *AIPS* task for fringe-fitting is **FRING**. This task estimates time variable station-based delays (phase derivatives wrt frequency) and rates (phase derivatives wrt time) using a self-calibration-like algorithm. Once these delays and rates are determined, the task **CLCAL** is used to produce the phase correction that should be applied to each integration period and spectral channel to correct for delay and rate effects. This use of **FRING** and **CLCAL** is discussed in detail in § 9.5.7.7. Two alternatives to **FRING** are the tasks **BLING** and **BLAPP** and the experimental version of **FRING** named **KRING**. **BLING** and **BLAPP** are discussed briefly in § 9.5.7.8. **KRING** provides a superset of the functionality in **FRING** with numerous enhancements such as: the use of extremely small scratch files, a parsimonious use of memory, possible solution extrapolation both backwards and forwards in time and a rationalized definition of SNR. For more information about **KRING**, type **HELP KRING** from within *AIPS*. When fringe-fitting to many small

scans, **KRING** can be substantially slower than **FRING**. When fringe-fitting data sets with large numbers of spectral channels and long solution intervals, **KRING** can be substantially faster than **FRING**.

The process of fringe-fitting, and then interpolating the solutions using **CLCAL**, can be a very time consuming process. Although it depends a lot on the size and structure of the data set, *the fringe-fitting time can equal or exceed the observing time for a large data set*. For this reason it is probably wise to run through the fringe-fitting procedure described in § 9.5.7.7 on a small amount of data first (say 30 minutes' worth) before attempting to process the whole data set. This is especially true if this is your first time processing multi-IF, multi-channel VLBI data. It is probably simplest to use **UVCOP** to copy out a short time range of data from your main file and to work only on this initially. Doing so also avoids the possible confusion of having many versions of extension tables.

9.5.7.1 Special considerations: SVLBI

The existing fringe-fitting tasks within *AIPS* have been enhanced to improve their performance when dealing with SVLBI data. In addition, several new tasks have been written to address problems specific to SVLBI fringe-fitting. The primary SVLBI fringe-fitting tasks in *AIPS* are **BLING** and **FRING** and are discussed in § 9.5.7.7–§ 9.5.7.9. The tasks **COHER** and **FRPLT**, described in § 9.4.6 and § 9.4.7, may be of particular interest when reducing SVLBI data.

There may be delay discontinuities in the recorded data for a variety of reasons such as tracking station handoffs, clock glitches, etc. The recommended method for dealing with such discontinuities is to force scan boundaries at such events. The task **INDXR** can be used to generate a new **NX** table with scan boundaries at desired locations using an input text file. In practice, either **INDXR** will do the right thing by design, or your P.I. information letter should have contained instructions on how to construct a text file in the proper format for **INDXR**.

A new task, **OBEDT**, is available which allows selection of specific orbital parameter ranges, through the creation of an output flag (**FG**) table. This can be used to constrain initial fringe searching.

SVLBI data often contain tracking passes three or four hours in length, for which fringes are mostly (or wholly) not apparent. Typically, the space-ground baselines will have the highest correlation coefficients near perigee, when those baselines are shortest. However, the imperfectly known orbit will cause high fringe rates and short coherence times. Near apogee, the coherence time is longer, and may be limited by the atmosphere above the ground telescopes, but the correlated flux also is much lower. Sometimes, it may be possible to find fringes for only 15 or 20 minutes, but that's better than nothing.

If no fringes are seen anywhere during a tracking pass, a useful trick is to set **APARM(7) = 0.01** to let through the highest SNR for each solution interval, and set **DPARM(5) = 1** to turn off the least squares solution. Then run **FRING** for an entire tracking pass, and use **SNPLT** with **OPTYP = 'DELA'** and **OPTYP = 'RATE'** to look for repeating values (usually easier to see in delay than in rate). Also, make plots with **OPTYP = 'SNR'** to see if slightly higher SNRs are found at a time when there seems to be some consistency in delay values. It may be necessary to try this process with several values of **SOLINT** in order to arrive at a guess for the fringe location. Use **VPLOT** to plot *uv* distance versus time for the space-ground baselines, then search using the **TIMERANG** and **REFANT** that provide the shortest projected baselines. Another possibility is to set **REFANT = ANTPNUM('MK')**, since the atmospheric coherence time should be longest at Mauna Kea, and increase **SOLINT** to a fairly large value in hopes something will show up. (Note, however, that a large **SOLINT** with a wide-open search window in delay and rate may require a large-memory computer or great inefficiency due to page faulting.)

If fringes are found somewhere, use **CLCOR** to center the fringes (see § 9.5.4.4), then run **FRING** again with small delay and rate windows (e.g., **DPARM(2) = 200** to **400** and **DPARM(3) = 40** to **80** at 1.6 GHz, or ~ 200

at 5 GHz). Set low SNR thresholds with `APARM(7) = 3.5`, and turn the least-squares solutions back on with `DPARM(5)=0`. Usually, it's best to turn on the exhaustive antenna search with `APARM(9) = 1`, since only a few space-ground baselines may show fringes. It can be helpful to use `SEARCH` to order the search from shortest to longest space-ground baselines.

It generally doesn't work well to use one tracking station to predict the results of another, because clock initialization offsets are typically relatively large and have unrelated errors, and fringe-rate errors also may be unrelated.

9.5.7.2 Special considerations: spectral-line

Delay and fringe-rate calibration of spectral-line VLBI data must be handled differently. The residual delay cannot be estimated from the source itself because, due to the very nature of the source, the delay is a rapidly varying function of frequency. The continuum calibrator, observed for this purpose, is first used to determine residual delays and fringe-rates which are then applied to the spectral-line source. A suitable channel or range of spectral-line channels “on-source” is then used to determine the residual fringe-rates. It is very important to note that in some situations, the residual fringe-rates determined from the calibrator may not be applicable to the line source because the fringe-rate residuals towards the two sources may be quite distinct. In such situations, the residual fringe-rates determined from the continuum source should not be applied to the line source. See § 9.5.7.10 for more details.

9.5.7.3 Special considerations: polarization

In addition to phase calibrating the LL and RR data separately, for polarization data the R-L phase and delay offsets must also be determined. This is outlined in § 9.5.7.11. After fringe-fitting, all parallel-hand fringe solutions need to be re-referenced to the same antenna.

9.5.7.4 Special considerations: phase-referencing

The process of phase referencing for VLBI data is conceptually very simple. Unfortunately, the technical difficulties in conducting a successful phase-referencing observation are primarily in setting up the schedule. So by the time you get around to reading this section, your project is either guaranteed to succeed or guaranteed to fail, depending upon how well your observations were designed. See the lecture by A. Beasley and J. Conway in “VLBI and the VLBA”, 1995, (ASP), and VLBA Scientific Memo No. 24 (2000, by J. Wrobel, C. Walker, J. Benson, and A. Beasley) for more details on how to design phase-referencing observations.

In “phase referencing,” the phase calibration for your target source is derived from a calibrator, or phase referencing, source observed for that purpose. First, you apply any available *a priori* phase-cals to both the target and cal source. Next, you fringe-fit, self-calibrate, and/or hybrid-map the cal source — whatever is needed to complete the phase calibration for that source. Finally, you apply the phase corrections so determined to the target source. In practice this is done by specifying the target as well as the cal source in the `SOURCES` adverb list whenever an `SN` table containing phase corrections for the cal source is applied using `CLCAL`. See § 9.5.1.2 for a specific example of how to run `CLCAL` to transfer phase corrections determined using a cal source to a target source.

Certain “instrumental” corrections such as those for unmodeled zenith delay may have subtle but significant effects on phase referencing. See § 9.5.7.5 for a discussion.

You can perform a “hybrid” form of phase referencing in some instances. It may be that your target source is too weak for initial fringe-fitting. In this case, you can fringe-fit the cal source data to determine the phase corrections to be applied to the target source data. Then, after averaging in frequency, the target source data may have adequate signal-to-noise to allow rate corrections to be determined for it by fringe-fitting. In this mode, you may or may not wish to zero the rate corrections determined on the cal source. If the cal source is “far” from the target source, the rate corrections may do more harm than good for the target source and should be zero’ed. On the other hand, your target source may be entirely too weak to fringe-fit on at all. In this case, you must rely on determining the phase corrections solely using your cal source.

If you are attempting phase-referenced astrometry, you may have a target source that is brighter than your cal source(s). In this case, you simply fringe-fit on the target source and transfer the solutions to the cal source(s). Be careful, if your goal is to extract absolute positional information, not to independently self-calibrate the cal and target sources.

9.5.7.5 Correcting for atmospheric delays

VLBI correlators remove some estimate of the non-dispersive atmospheric delay at the elevation and frequency of the observation from the data. These a priori models are usually fairly good, but careful observations can improve upon them. *AIPS* offers a number of options to deal with this problem. **DELZN** uses multi-band delay (see § 9.5.7.6) in an SN table to fit for the zenith tropospheric delay and the clocks as a function of time. It works best if the observations include data on a variety of calibrators well distributed around the sky. **DELZN** applies a correction to a CL table or writes a file to disk with zenith atmospheric delays and possibly clock offsets that can be used by **CLCOR**, **OPCODE='ATMO'** to correct a CL table. The second option is for the situation where the data used by **DELZN** is in a different file from the data that needs to be corrected.

To correct an attached CL table, the typical inputs for **DELZN** would be:

- > **TASK 'DELZN' ; INP C_R** to review the inputs.
- > **INDISK n ; GETN ctn C_R** to specify the input file.
- > **SNVER snin C_R** to select the SN table containing multi-band delays.
- > **GAINVER clin C_R** to select the CL version to which solutions are to be applied.
- > **APARM(4) 1 C_R** to create a new CL table.
- > **APARM(5) 1 C_R** to solve for atmosphere and clocks
- > **SOURCES 'DA913' , '' C_R** to specify the sources to be corrected.
- > **CALSOR** to specify the calibrator sources observed at a large variety of
'0103+337', '0140+412', '0150-334', '0159+418', elevations
'0202+319', '0244-297', '0358+210', '0425+174',
'0641+392' C_R
- > **OPTYPE 'MDEL' C_R** to use multi-band delay
- > **DOTV -1 C_R** to make PL files
- > **GO C_R** to run the program.

This will generate a CL table and several PL files that show the data and the fitted model. You are **strongly encouraged** to examine these.

To create an output file rather than correct a CL table use the same inputs as above except:

- > **APARM(4) 0 C_R** to create no CL table.
- > **SOURCES '' C_R** to correct no sources
- > **CALSOR '' C_R** to use all calibrator sources.
- > **OUTFILE 'MYVLB:BZ199.DELZN' C_R** output file name

Again, you are advised to examine the resulting plot files which show both the data and the fitted model. The output file can be read in with `CLCOR (OPCODE='ATMO'; INFILE='MYVLB:BZ199.DELZN')` to correct a CL table.

There is another task designed to deal with the effects of zenith delay in phase-referencing observations. Phases for the target source in phase referencing are corrected by the phases at the calibrator which usually is at a different elevation. Task `DFCOR` is a special version of `CLCOR` which applies the 'ATMO' operation to correct the CL table for the difference in elevation between the target source and adjacent calibration sources without applying the full atmospheric delay correction.

If the SN table contains dispersions as well as multi-band delays, `DELZN` may also be run to fit a zenith angle model to the dispersions (`OPTYPE = 'DISP'`). Like the 'MDEL' operation, the zenith and time function found by `DELZN` can be applied to a CL table or written to a text file for later application by `CLCOR` with `OPCODE = 'DISP'`.

9.5.7.6 Finding multi-band delays

For astrometric and geodetic experiments and to use `DELZN` (see § 9.5.7.5), the multi-band delay must be determined. The multi-band delay is the delay caused by errors in the station positions and the difference between the correlator model and reality for clocks and the troposphere. The multi-band delay is best determined over IFs which are widely spaced in the frequency band. After the instrumental phases have been corrected (§ 9.5.4.3–§ 9.5.4.4), the multi-band delay can be determined in one of two ways. For strong sources do a global fringe fit as described in § 9.5.7.7 setting `APARM(5)=0`, and then run `MBDLY` on the resulting SN table.

Typical inputs for `MBDLY` would be:

> <code>TASK 'MBDLY'; INP C_R</code>	to review the inputs.
> <code>INDISK n; GETN ctn C_R</code>	to specify the input file.
> <code>INVERS SN table from FRING</code>	to select input SN table
> <code>OUTVERS 0 C_R</code>	make new SN table
> <code>BIF 0; EIF 0 C_R</code>	to select all IFs.
> <code>SUBARRAY 0 C_R</code>	to select all subarrays.
> <code>APARM 0 C_R</code>	the defaults are generally O.K.
> <code>GO C_R</code>	to run the program.

This will produce a new SN table with the MBDELAY columns filled in.

For weak sources, use `APARM= 2 0 0 1 1` in `FRING`. This averages each IF and fits a multi-band delay across them. Note that this *does not* solve for single-band delays, unlike the previous method. This will also produce a new SN table with the MBDELAY columns filled in.

Note that `MBDLY` can fit the single-band delays for a multi-band delay plus a dispersion (phase varies with wavelength rather than frequency) with `OPTYPE = 'DISP'`. `FRING` can do a very similar fit, after finding the single-band delays, if you set `APARM(10) = 1`. Wide bandwidth observations at low frequencies may benefit from this.

9.5.7.7 Antenna-based fringe-fitting

To see an example of the residual phase errors in your data, use `POSSM` to view the phase on a short calibrator scan (at some time other than that used to solve for the phase-offsets in § 9.5.4.4). In general, there will be

a gradient in phase between the IFs (due to the “multi-band” delay) and also small gradients within each IF (caused by small residual “single-band” delays). These time-variable phase gradients are mainly due to inaccuracies in the geometrical time delays that the correlator assumed for the time of arrival of the wavefront at each antenna. These inaccuracies arise from propagation effects through the troposphere and ionosphere, inaccurate Earth geometry, etc. and give *phase* errors which are proportional to frequency. Such phase errors prevent integration of the data over frequency (or cause a loss of coherence if you do). Similarly, **VPLT** will show, on any single IF and spectral channel, phases which change rapidly with *time*. Again, these are due to unavoidable inaccuracies in the correlator model; such large “phase rates” prevent integration over time. Both of these points are illustrated in Figure 9.1.

You will want to run **FRING** to correct for these residual rates. You can help these tasks by making sure that the reference pixel in frequency is in the center of the band (Nchan/2+1 is best). Use task **CENTR** or any of the tasks with the **FQCENTER** adverb to fix data sets for which this is not true.

FRING and **KRING** use a global fringe-fitting algorithm described by Schwab and Cotton, 1983, *Astron. J.*, 88, 688. Unfortunately, these are large and complicated tasks. The procedure **VLBAFRNG** is available to simplify access to **FRING** and **VLBAKRNG** access to **KRING**. Versions of these procedures for phase-referencing experiments are called **VLBAFRGP** and **VLBAKRG**. For all these procedures, if the **SOURCES** adverb is set, then **CLCAL** is run once to apply the results of **FRING** (or **KRING**) for each source in **SOURCES**. For the phase-referencing procedures (**VLBAFRGP** and **VLBAKRG**), any source that is in the **SOURCES** list that is *not* in the **CALSOEUR** list will be phase referenced to the *first* source in the **CALSOEUR** list. Note that, if every source in the **SOURCES** list occurs in the **CALSOEUR** list, **VLBAFRNG** and **VLBAKRNG** will run identically to **VLBAFRGP** and **VLBAKRG**, respectively. If the **SOURCES** list is empty, **VLBAFRNG** and **VLBAKRNG** will run **CLCAL** once over all sources, while **VLBAFRGP** and **VLBAKRG** will run **CLCAL** once referencing all the sources to the first source in **CALSOEUR**. These procedures will produce new (highest numbered) SN and CL tables.

Sample inputs for procedure **VLBAKRNG** are:

> RUN VLBAUTIL C _R	to acquire the procedures; this should be done only once since they will be remembered.
> INDISK n ; GETN ctn C _R	to specify the input file.
> TIMERANGE 0	to include all times.
> BCHAN 0 ; ECHAN 0 C _R	to use all frequency channels.
> GAINUSE CLin C _R	to use the CL table with all the calibration up to this point.
> REFANT n C _R	to specify an antenna that is present most of the time as the reference antenna.
> SUBARRAY 0 C _R	to use all subarrays.
> SEARCH 0 C _R	to try all antennas as a reference antenna if fringes cannot be found using REFANT . <i>This is different from FRING; in FRING this must be set to try other reference antennas.</i>
> OPCODE ' ' C _R	to leave all solutions in the output SN table.
> CPARM 0 C _R	to use defaults for KRING steering parameters; this is okay for strong sources.
> CPARM(1) x C _R	to specify the minimum integration time in seconds.
> CPARM(8) 1 C _R	to avoid re-referencing solutions; do this <i>only</i> for polarization experiments.
> CALSOEUR 'src1', 'src2' C _R	to specify the sources to fringe fit using KRING .
> SOURCES 'src1', 'src2' C _R	to have CLCAL run for each source using the interpolation method given below.
> INTERPOL 'AMBG' C _R	to use the “AMBG” interpolation method (linear phase connection using rates to resolve phase ambiguities).
> BADDISK 0 C _R	to use all disks for scratch files.

> **VLBAKRNG** C_R to run the procedure.

Procedure **VLBAKRG** sets the same adverbs as **VLBAKRNG** *except*

- > **SOURCES** 'src1', 'src2', 'src3' C_R to have **CLCAL** run for each source using the interpolation method given by **INTERPOL**. Any source here that is not in the **CALSOEUR** list will be phase referenced to the first source in the **CALSOEUR** list. In this example, *src3* is phase referenced to *src1*.
- > **VLBAFRNG** C_R to run the procedure.

VLBAFRNG and **VLBAFRGP** are identical except there is no **OPCODE** (it is equivalent to **DPARM(8)**) and **DPARM(4)** and **DPARM(7)** in **FRING** are the same as **CPARM(1)** and **CPARM(8)** in **KRING**, respectively. Also note the different use of **SEARCH** in **FRING** and **KRING**.

Suitable inputs for the fringe-fitting task **FRING** are, in detail:

- > **TASK** 'FRING' ; **INP** C_R to review the inputs.
- > **INDISK** *n* ; **GETN** *ctn* C_R to specify the input file.
- > **CALSOEUR** '' C_R to find solutions for all sources.
- > **TIMER** 0 C_R to find solutions for all times.
- > **DOCALIB** 1 C_R to apply the most complete calibration file including amplitude calibration and IF and channel phase offsets to both the visibilities and the weights.
- > **GAINUSE** 3 C_R to use CL table 3.
- > **SNVER** 2 C_R to write solutions into SN table 2.
- > **BCHAN** 0; **ECHAN** 0 ; **CHINC** 1 C_R to use all spectral channels within each IF channel.
- > **FLAGVER** 0 C_R to apply the most recent flag table.
- > **SMODEL** 0 ; **CLR2NAME** C_R to use a point-source at the origin model for the sources, rather than a Clean-component model.
- > **SOLINT** 3 C_R to set the solution interval in minutes; do not exceed the atmospheric coherence time (see below). Setting **SOLINT** to 0 sets solution intervals equal to scan lengths.
- > **SEARCH** *ref2*, *ref3*, ... C_R to specify the order over which antennas are searched for fringes when the exhaustive search is requested. If **APARM(9)** is set, all antennas will be searched for fringes with **SEARCH** controlling the order of the search. Note, **REFANT** should be **SEARCH(1)**. The use of **APARM(9)** and **SEARCH** makes **FRING** much more likely to find fringes to weak antennas and is highly recommended.
- > **ANTWT** 0 C_R to apply no additional weights to the antennas before doing the solutions. If the amplitude calibration was incorrect, you can use this option to force antenna weights up or down to control the weight **FRING** gives to data to each station when making the global solutions. Unless the **SEARCH** option described above is chosen, **ANTWT** also controls the order in which antennas are tried as secondary reference antennas after failing to find fringes on the **REFANT**. Give higher weight to antennas you want to see used as secondary references.

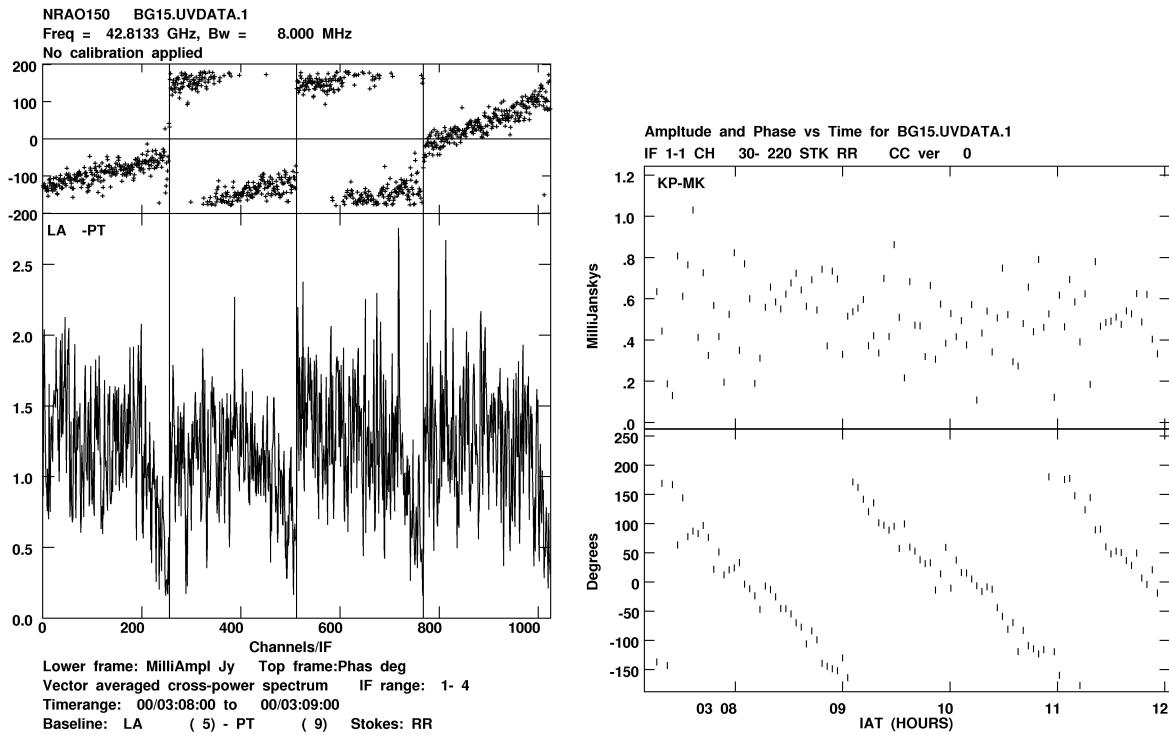


Figure 9.1: *left:* A [POSSM](#) plot of the 43-GHz spectrum of the quasar NRAO150 on the Los Alamos to Kitt Peak baseline. The plot shows that the observation was performed with 4 IFs, with 256 spectral channels within each IF. The upper frame shows the phase variation with frequency; within each IF, the small phase slope is caused by a residual delay error. The phase offsets between the IFs can be clearly seen as well. Both the residual delay error and the phase offsets must be determined and removed before the data can be spectrally averaged (see § 9.5.7). *right:* A [VPLT](#) plot of the uncalibrated amplitude (upper frame) and phase (lower frame) as a function of time for the source NRAO150 at 43 GHz on the baseline Kitt Peak to Mauna Kea. Note how the phase varies as a function of time; this variation is equivalent to a residual fringe rate of 8.3 mHz. Unless the fringe rate is determined and removed (see § 9.5.7), the data cannot be averaged in time.

> [REFANT](#) 5 C_R

to choose an antenna that will give fringes for most of the scans. This is important: [FRING](#) will search for fringes to this antenna first. If it fails for some reason, it will select another reference antenna, based on the [ANTWT](#) data, and, if it still fails, give up (see however the discussion of the new [SEARCH](#) adverb above). In this case, you should look for scans with no fringes or a bad reference antenna may be causing the problem. A big, sensitive antenna is often used as [REFANT](#) (e.g., Effelsberg). Occasionally, it may be helpful to split your data set up into 2 or 3 sections, which are fringe-fitted with different [REFANT](#) (e.g., a “European” and a “US” part of the observations). Changes in reference antenna should, in general, not cause problems.

> [APARM](#)(1) 2 C_R

to accept solutions when only 2 antennas are present; default is 6.

> [APARM](#)(2) 0 C_R

to have the data divided by the model before fitting fringes; [APARM](#)(2) > 0 tells [FRING](#) that the data have already been divided by a model.

- > **APARM(3) 0 C_R** to treat polarizations separately; **APARM(3) > 0** averages RR and LL.
- > **APARM(4) 0 C_R** to use the frequencies individually within each IF; **APARM(4) > 0** causes the frequencies within each IF to be averaged before the solution.
- > **APARM(6) 1 C_R** to get some useful, but limited, messages, including the SNR.
- > **APARM(7) 9 C_R** to avoid false detections by setting a moderately high minimum for the SNR accepted. You may wish to use a lower threshold (especially for SVLBI) although **APARM(7) less than about 3** is probably not useful.
- > **APARM(8) 0 C_R** to set the maximum number of antennas (if no AN table).
- > **APARM(9) 1 C_R** to enable the exhaustive search mode.
- > **APARM(5) 0 C_R** to do least-squares fits in each IF. **APARM(5) ≤ 0** means to solve separately for the rate, single band delay and phase of each IF. **1.5 > APARM(5) > 0** means to solve for one single rate and multi-band delay affecting all IFs. If **2.5 > APARM(5) > 1.5**, the task additionally solves for the difference between the multi-band delay and single-band delay, *i.e.*, it allows for a different gradient of phase versus frequency within an IF than between IFs. Note, however, that unlike **APARM(5)=0**, this option assumes that the single-band delay is the same in each IF; it therefore solves for a single value for the difference between multi-band and single-band delay affecting all IFs. Normally users should use **APARM(5)=0** for multi-IF data. Primarily for the **VLA**, a fourth option, **APARM(5) = M** has been added, with **M > 2**. It divides the IFs into **M – 1** groups and solves for one delay per group.
- > **ANTENNAS 0; DOFIT 0 C_R** to allow all antennas to be fit. Only baselines between antennas listed in the **ANTENNAS** adverb are used in the fringe search. If any antennas are specified in **DOFIT** however, a solution is made only for those antennas; all other selected antennas are assumed to be already calibrated and are passed through with no additional corrections. See the **HELP** file for **FRING** under **DOFIT**. **DOFIT** is only active if **APARM(9)** is set.
- > **DPARM(1) 1 C_R** to use one baseline combination in the initial coarse (**FFT**) fringe search. This provides a starting guess for the least-squares solution. If you are searching for weak fringes you should consider using two and three baseline combinations in the search; this can improve sensitivity in the initial fringe search. See Lecture 19 in *Synthesis Imaging in Radio Astronomy*, edited by R. Perley, F. Schwab, and A. Bridle, for an explanation of how this global multi-baseline searching works. Note that if your source structure is complex and you have not divided the data by an accurate source model, then setting this parameter to one is safest. **DPARM(1)>1** works even when the integration times are not equal.

- > **DPARM(2) 0 C_R** to set the full width of the delay window in nsec, centered around 0, to search; the default, chosen here, is to use the full Nyquist range defined by the frequency spacing. A smaller search window can permit a lower SNR threshold to be set, but can also result in lost data due to failed fringe searches. For the VLBA, a **DPARM(2) = 1000** window is usually adequate.
- > **DPARM(3) 0 C_R** to set the full width of the fringe-rate window in mHz, centered around 0; the default, chosen here, is to use the full Nyquist range defined by the integration time. A smaller search window can permit a lower SNR threshold to be set, but can also result in lost data due to failed fringe searches. For the VLBA, a **DPARM(3) = 200** window is usually adequate.
- > **DPARM(4) 2 C_R** to specify the correlator integration time in seconds; use **DTSUM** to find the correct value. For data from the VLBA correlator, **DPARM(4)=0** will cause **FRING** to determine the correct integration time by examining the data file directly.
- > **DPARM(5) 0 C_R** to do both the coarse and the least squares solutions; set to 1 if you require only **FFT** solutions.
- > **DPARM(6) 1 C_R** to keep, for single-source files, frequencies separated in the output file; the default is to average frequencies within IFs. This parameter does not affect multi-source files.
- > **DPARM(7) 0 C_R** to re-reference solutions to a common reference antenna; when processing polarization data, set this to 1 to avoid the re-referencing.
- > **DPARM(8) 0 C_R** to disable the zero'ing options — see the **HELP** file. **WARNING, DPARM(8) > 0 will discard parts of the final solution. Be sure to use this option with extreme care.**
- > **DPARM(9) 0 C_R** to allow fitting for rate. **DPARM(9)> 0**, causes the task to suppress rate fitting entirely rather than zeroing it after the fact as in **DPARM(8)**.
- > **INP C_R** to check the inputs.
- > **GO C_R** to do the fit — finally.

Note that **FRING** finds solutions in two steps. First approximate solutions are found in the **FFT** step using combinations of one, two, or three baselines (see **DPARM(1)** above). Then, as long as **DPARM(5) < 1**, a least-squares algorithm uses these approximate values as a starting point for refining the solutions. Especially for weak sources, the least-square solution may wander outside narrow constraints set by **DPARM(2)** and **DPARM(3)**.

Note that the **SOLINT** interval should be selected with consideration of the atmospheric coherence time, but must be long enough that high signal-to-noise-ratio solutions are achieved. For observations between 1.6 and 15 GHz, solution intervals of 3–6 minutes (and often longer) should be fine. At other frequencies shorter solution intervals may be required. In these cases, experiment with different length solution intervals on short sections of data. Note that solution intervals greater than the scan length will never be used; the scan lengths are listed in the **NX** table and may be examined using **PRTAB** or **LISTR** with **OPTYPE='SCAN'**.

If the source is complex, and especially if the visibility phase of the source changes during **SOLINT**, it is useful to divide the data by a Clean model derived from previous observations or from an earlier attempt at processing the data. This Clean model can be specified by filling in **IN2NAME et al.** Situations where this is useful include observations of equal doubles (where there are zeros in the amplitude and, hence, rapidly

changing phases) or very large sources (of order arcseconds). If you are using multi-baseline searching (*i.e.*, **DPARM(1)** > 1), then solutions may be more sensitive to source structure and an input model may be useful if the structure phases are larger than one radian on many baselines. When using a model, convergence may be improved by weighting the data by $1/\sigma$ rather than $1/\sigma^2$; set **WEIGHTIT** = 1.

You should check the SNRs found by **FRING** carefully; they are printed if **APARM(6)** > 0. The SNRs estimated during the **FFT** search are used to determine if the SNR of a solution is \geq the threshold set in **APARM(7)**. If they are not, then that solution is flagged before being passed to the more accurate least-squares routine. Users should check that the SNRs found in the LSQ routine match those expected. If the detected SNRs are too low, **SOLINT** may be too long or too short or other parameters may be set wrongly.

Be warned that proper scaling of the SNRs by **FRING** depends upon whether or not the data weights have been properly calibrated. Task **FIXWT** may be used to calibrate the weights, but changing the SNR threshold for **FRING** directly (**APARM(7)**) usually produces satisfactory results.

The final delay and rate solutions and their SNRs should be inspected using **LISTR**:

- > **TASK 'LISTR' ; INP C_R** to review the inputs.
- > **INDISK n ; GETN ctn C_R** to specify the input file.
- > **OPTYPE 'GAIN' C_R** to list gain solutions.
- > **INEXT 'SN' ; INVER 2 C_R** to list SN table 2 (as above).
- > **DPARM 6, 0 C_R** to list delay; use 7 for rate, 1 for phase, and 8 for SNR.
- > **GO C_R** to run the program.

Alternatively, the solutions can be plotted against time to make sure they are sensible. Use **SNPLT**:

- > **TASK 'SNPLT' ; INP C_R** to review the inputs.
- > **INDISK n ; GETN ctn C_R** to specify the input file.
- > **OPTYPE 'RATE' ; DOBLANK 1 C_R** to plot rate solutions including failed ones, **OPTYPE 'DELA'** for delay solutions.
- > **OPCODE '' C_R** to plot each polarization and IF in separate plots.
- > **NPLOTS 5 C_R** to plot five antennas/IFs/polarizations per page.
- > **INEXT 'SN' ; INVER 2 C_R** to plot SN table 2 (as above).
- > **GO C_R** to run the program.

It is a good idea to plot the solutions for **OPTYP 'RATE'** and '**DELA**' (single-band delay) as well as the associated '**SNR**'s. They should be smoothly varying functions. Delays and rates should be found only within your specified windows. Check for suspicious detections at the limits of the search windows — for instance, they could be detections of side lobes of the main fringe. If you used windows smaller than Nyquist, you may want to check your detections using bigger windows. Gaps in detections with time can occur if, *e.g.*, tapes were bad, antennas were off source, the source visibility is in a minimum, or the reference antenna choice was bad. It pays to investigate such problems at this point before proceeding.

Discrepant SN solutions can be removed using the interactive task **SNEDT** or the non-interactive task **SNSMO**. If, in the latter, the **CPARM** values are set, then the SN solutions will be clipped if they differ from the running mean by amounts which you can specify. If the **BPARM** values are set, the solutions are then smoothed and clipped entries can be replaced with mean values based on a boxcar- or median-window-filter average. See the explain file for details. Although **SNSMO** allows the option of modifying the input SN table, it is safest to have it create a new one. If you have a lot of discrepant SN values, you should also consider using option **INTERPOL** = '**POLY**' in task **CLCAL** (see below).

If there isn't enough disk space to run **FRING** on all the data at once (because of the large scratch file that **FRING** insists on creating), you can run **FRING** multiple times specifying time ranges and explicitly setting **SNVER** to the same SN table.

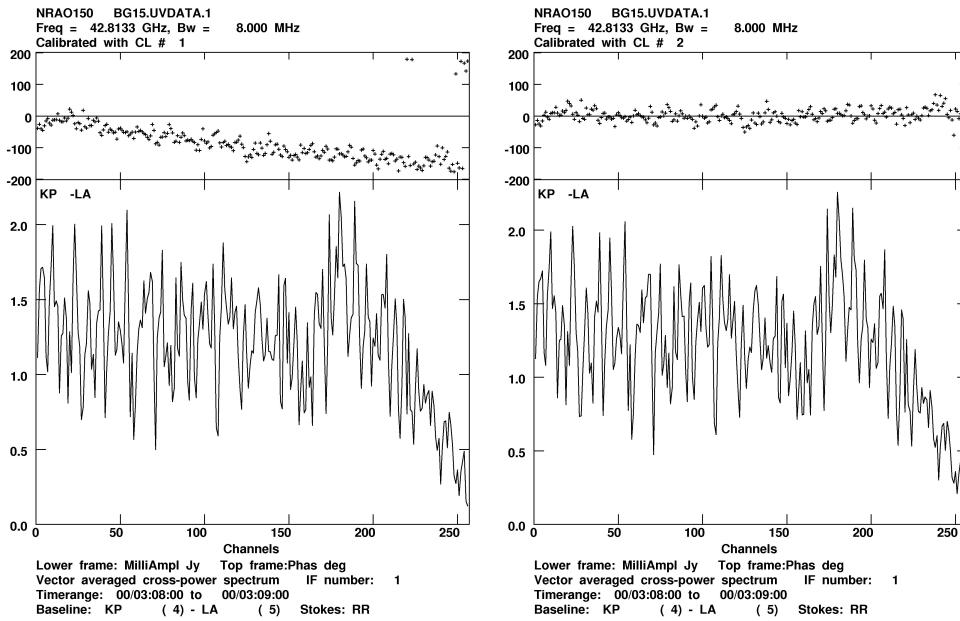


Figure 9.2: *left:* A [POSSM](#) plot of the uncalibrated spectrum of NRAO150 at 43 GHz on the baseline Kitt Peak to Los Alamos. The plot shows the spectrum for a single IF to show the effects of the residual delay error more clearly. The phase slope as a function of frequency is clear evidence for a small delay error in the correlator model. *right:* The same data as shown on the left, but corrected for a delay error of -55 nanosec and a residual fringe-rate of -2.0 milliHz. Note how the phase as a function of frequency is now flat and centered around zero degrees. These data can now be averaged in frequency, if desired.

If, for some reason, you set the parameters of [FRING](#) or [SNSMO](#) wrongly and the resulting SN table is unusable, it is wise to avoid confusion by deleting it using [EXTDEST](#) and starting over again.

Once a valid SN table has been produced, the next step is to interpolate the solutions found onto the finer grid of entries in a CL table using the task [CLCAL](#) with [INTERPOL](#) = 'AMBG' as described in § 9.5.1.1.

Once a final CL table is generated, its effect on the data can be viewed using tasks [VPLOT](#) and [POSSM](#) by setting [DOCAL](#)=1 and [GAINUSE](#) to the version number of the final CL table; see § 9.4. Optionally, in [VPLOT](#), one can average over spectral channels and/or IF channels before plotting. Use [VPLOT](#) to plot a time range covering a few [FRING](#) solution intervals on a strong source. Phase variations should be small with no jumps. If this is not the case, check the inputs to [FRING](#) (especially [SOLINT](#)) and [CLCAL](#). A comparison of before and after phases is shown in Figure 9.2.

9.5.7.8 Baseline-based fringe-fitting

Baseline-based fringe-fitting, implemented in [BLING](#) and [BLAPP](#), is an alternative to using [FRING](#) and [CLCAL](#) described above. Whereas [FRING](#) searches and solves for station rates and delays globally, [BLING](#) makes independent fits to each baseline for delays and rates, creating a BS table of baseline-based solutions.

In most cases, the global fringe-fitting described in § 9.5.7.7 should be used since [FRING](#) should be able to fringe-fit weaker sources more reliably. However, there are some instances in which baseline-based fringe-fitting is to be preferred. Note that [FRING](#) may be used to do baseline-based fringe-fitting by running it many times, each time specifying only 2 antennas. [BLING](#) has not been actively maintained or used, and so may be less reliable. Amongst the advantages of the baseline-based fringe-fitting are:

1. **BLING** may be more robust than **FRING** even in the absence of an accurate source structure model.
2. Fringe solutions can be found for cross-polarized fringes without editing the *uv* header.
3. As presently implemented, for a given number of IF and spectral channels, **BLING** can solve for longer scans than can **FRING**.
4. **BLING** has the option of adjustable, non-zero centered fringe-search windows, which can be controlled from an external file. This option may be important in fringe-fitting Space VLBI data.

BLING is distinguished from **FRING** by the ability to directly control the fringe search on each separate baseline, and by the ability to solve for fringe acceleration if required. Separate fringe windows in delay, rate and acceleration, and different solution intervals can be set for each individual baseline. The fringe windows may have non-zero offsets and can be specified using the input adverbs or, more flexibly, by drawing up an external ASCII control file of fringe-prediction windows and **BLING** control parameters. The latter option allows the specification of time-variable fringe-search parameters across the observing file. The general algorithm follows that described by Alef and Porcas, 1986, (*Astron. Astrophys.*, **168**, 365); **BLING** also allows the stacking of data from different baselines, as discussed by Schwab and Cotton, 1983, *Astron. J.*, **88**, 688. In addition, model division is possible before the fringe-fitting is performed and cross-polarized fringe searches can also be conducted without editing the *uv* header. **BLING** writes the results to a BS table. These baseline-based solutions can be converted into antenna-based corrections using the separate task **BLAPP** and then applied to the data.

Acceleration may be solved for by conducting a coarse search in the specified acceleration window. The results of this search are then interpolated to estimate the final solution. Fringe acceleration searches can considerably increase the amount of CPU time it takes to run **BLING**. Therefore, you may wish to turn off the acceleration search (**DPARM(7)** to **DPARM(9)**) unless you need it for space VLBI.

Full details concerning **BLING** input parameters can be found by typing **EXPLAIN BLING**. This includes information concerning the format required for the external control file. Earlier versions of **BLING** are discussed in *AIPS Memo 89* (1994, “Baseline-Oriented Fringe Searches in *AIPS*” by Chris Flatters). Typical input parameters to **BLING** are given below:

> TASK 'BLING' ; INP C _R	to review the inputs.
> INDISK <i>n</i> ; GETN <i>ctn</i> C _R	to specify the input file.
> CALSOUR 'DA193' , ' ' C _R	to specify the calibrator source.
> STOKES 'LL' C _R	to select the Stokes.
> TIMERANG 0, 10, 5, 0, 0, 11, 0, 0 C _R	to limit the time range.
> ANTENNAS 3 ; BASELINE 0 C _R	to select all baselines to antenna 3.
> SUBARRAY 1 C _R	to use subarray 1.
> FREQID 1 ; BIF 1 ; EIF 0	to use frequency ID 1 with all IFs.
> BCHAN 1 ; ECHAN 0 C _R	to use all spectral channels.
> DOCAL 1 ; GAINUSE <i>cln</i> C _R	to apply amplitude calibration before fringe fitting.
> CLR2N ; NMAPS 0 C _R	to do no model division.
> SOLINT 0.5 C _R	to use a 30-second fringe solution interval.
> INFIL ' ' C _R	to use the adverbs rather than an external control file.
> APARM(1) 2 ; APARM(2) 0 C _R	to set the integration time; no model division.
> APARM(3) 0 C _R	to do no stacking of baselines.
> APARM(4) 0 ; APARM(5) 0 C _R	to set minimum acceptable SNR to 5 and accept coherence of 20%
> DPARM 0 C _R	to use the default fringe windows and no acceleration search.
> DOUVCOMP 1 C _R	to use compressed scratch files.

> **BADDISK 0** to use all disks for scratch files.
 > **GO C_R** to run the program.

Note that baseline-stacking (**APARM(3)**) is not implemented for data sets with unequal integration times. Also, note that the fringe-rejection criteria specified using **APARM(4)** and **APARM(5)** are important parameters. The use of compressed scratch files is recommended and is not believed to have a significant impact on precision. Note, if model division is required, **APARM(2)** should be set and the source should be entered explicitly.

BLING will execute with a summary line marking the start of each baseline processed. The resulting BS table can be examined using task **BSPRT**, with input parameters:

> TASK 'BSPRT' ; INP C_R	to review the inputs.
> INDISK n ; GETN ctn C_R	to specify the input file.
> INVERS bsin C_R	to specify the BS table version number.
> DOCRT -1 C_R	to send output to an external file.
> OUTPRINT 'FITS:BSPRT.LIS' C_R	to define the output file name.
> GO C_R	to run the program.

For printing to the screen, select **DOCRT=1**.

The BS table output includes the estimated fringe parameters and their associated errors. Note that due to changes in **FFT** interpolation the errors may be an overestimate. The **BLING** solutions are interpolated and factorized into antenna-based gain solutions using **BLAPP**. This task either writes a solution (SN) table which can be applied using **CLCAL**, or allows a CL table to be updated with the calibration information directly. These options are selected using **OPCODE='SOLV'** or **OPCODE='CAL'** respectively. **BLAPP** can interpolate solutions with unequal time sampling and includes the acceleration term in interpolation if it is available. Typical inputs to **BLAPP** are:

> TASK 'BLAPP' ; INP C_R	to review the inputs.
> INDISK n ; GETN ctn C_R	to specify the input file.
> INVERS bsin C_R	to specify the input BS table version number.
> SOURCES '' ; STOKES 'LL' C_R	to do all sources and a specific Stokes.
> FREQID 1 C_R	to select frequency ID 1.
> TIMERANG 0 ; ANTENNAS 0 C_R	to do all times and antennas.
> SUBARRAY 1 ; REFANT 3 C_R	to do subarray 1 with reference antenna 3.
> ANTWT 0 C_R	to use equal antenna weights.
> OPCODE 'SOLV' C_R	to solve for a SN table.
> GAINVER clin C_R	to specify the input CL table which defines the times for which solutions are desired.
> BADDISK 0 C_R	to use all disks for scratch files.
> GO C_R	to run the program.

The resulting solution (SN) table can be plotted using **SNPLT** in the standard fashion. For **OPCODE='CAL'** the output CL table also needs to be specified using **GAINUSE**. If an SN table is generated, it can be smoothed or clipped using task **SNSMO** and applied using **CLCAL** as described in § 9.5.1.1.

9.5.7.9 SVLBI-specific techniques

An alternative approach to direct fringe detection of each individual baseline to the orbiting antenna is to first calibrate the ground array using conventional fringe-fitting techniques, then coherently combine all ground antennas to improve the fringe detection sensitivity to the spacecraft. Several incarnations of

this approach exist within *AIPS*. The *AIPS* tasks **FRING**, **BLING**, and **KRING** all allow baseline stacking which can be used to fringe fit the space baseline using composite baselines. It was shown in VLBA Scientific Memo No. 13 (1996, “Global ground VLBI network as a tied array for space VLBI”, by L. Kogan) that the method of phasing a group of ground-based antennas and the method using global fringe fitting with baseline stacking give the same minimum detectable flux density. Therefore, baseline stacking with **DPARM(1)=3** in **FRING** should yield the best possible sensitivity. There are other options which also may be explored. The adverb **DOFIT** in **FRING** and **KRING** can be used to solve for subsets of the available antennas in order to find good solutions for the ground antennas in a dual round of fringe-fitting. The exhaustive baseline search mode, used by default in **BLING** and **KRING** and activated in **FRING** by setting **APARM(9)=1**, allows more baselines to the spacecraft to be searched.

9.5.7.10 Spectral-line fringe-fitting

The determination of the delay and fringe-rate calibration is a two- or three-step process for spectral-line VLBI data. First, the residual delay and fringe-rates are estimated for each antenna from the continuum calibrators. Then, residual fringe-rates must be determined again for the line source using a “strong” channel or range of channels. As an intermediate step, the phases of the line source should be examined to check that the calibrator’s residual fringe-rates haven’t destroyed phase coherence; if so, then the calibrator’s residual fringe-rates should not be applied to the line source.

If computer memory is limited, **FRING** must trade off the number of spectral channels against the length of the solution interval. For continuum calibrators in a spectral-line dataset, the large number of spectral channels may force **FRING** to require too short a solution interval. (**FRING** now allocates memory dynamically and may be able to handle large cases so long as the computer is adequately equipped with real and swap memory.) This limit can be overcome by running **UVCOP** to extract the continuum calibrators into a separate data file and then running **AVSPC** with **AVOPTION 'SUBS'** to average spectral channels coherently within each IF. **INDXR** should be run to regenerate an **NX** table. **FRING** will then allow more reasonable solution intervals.

Run **FRING** as follows only on the continuum calibrators to determine residual delays and fringe-rates:

- | | |
|--|--|
| > TASK 'FRING' ; INP C_R | to review the inputs. |
| > INDISK n ; GETN ctn C_R | to specify the input file. |
| > CALSOUR 'BLLAC','DA193' C_R | to select continuum calibrator sources. |
| > DOCALIB 1 C_R | to apply the current calibration. |
| > GAINUSE clin | to specify which CL table to use. |
| > SMODEL 0 C_R | to use the null source model (points at the origin). |
| > FLAGVER 0 C_R | to apply the most recent flag table. |
| > BCHAN 10 ; ECHAN 115 C_R | to exclude the edges of the band; normally the data in these channels are corrupted by the bandpass filters. |
| > REFANT 5 C_R | to select a reference antenna (see continuum discussion). |
| > SOLINT 6 C_R | to set the solution interval in minutes. It should not exceed the coherence time. |
| > APARM(6) 1 C_R | to get some useful, but limited, printout; gives SNR. |
| > APARM(7) 7 C_R | to avoid false detections by setting the minimum acceptable SNR. <i>Warning:</i> solutions with lower SNR will be flagged as bad which will ultimately flag the affected data. |
| > DPARM(1) 1 C_R | to use one-baseline combination in initial, coarse fringe search (FFT). This provides starting points for the least-squares solutions. |

- > **DPARM(2) 10000 C_R** to select a delay window in nsec, centered around 0(!). The default is to use the Nyquist range. For a 250kHz-bandwidth observation, setting this value to 10000 nsec is equivalent to setting the search window to 5 delay channels, which is usually sufficient.
- > **DPARM(3) 200 C_R** to select a fringe-rate window in mHz.
- > **DPARM(4)= 1 C_R** to tell **FRING** the correlator integration time.
- > **DPARM(5) 0 C_R** to do the least-squares solution.
- > **SNVER 0 C_R** to write solutions in a new SN table.
- > **GO C_R** to do the fit.

If the calibrators had been extracted to a separate data set, use **TACOP** to copy the resultant SN table back to the line data set. **SNCOP** may be used when the number of IFs in the two data sets is different. This is more likely to arise when you use a strong line signal in one IF to solve for the delays of a larger data set.

Run **CLCAL** to apply the delay and fringe-rate solutions to all sources, as is described in § 9.5.1.1, and then carefully examine the phase coherence of the line source in a suitable line channel (or group of channels) using **POSSM** or **COHER** before and after applying the new CL table. It may be that the fringe-rate solutions have made the phase coherence worse. In this case, you must run **SNCOR** using the 'ZRAT' option to zero the fringe-rates and then re-run **CLCAL**.

After this point, the calibrator data is usually of little or no interest. But, if you do plan to use the calibrator data further, remember to be careful to juggle the SN and CL tables correctly. Even if you decide not to apply the calibrator fringe-rates to the line-source, they are still applicable for the calibrator itself. The CL table created using the un'ZRAT'-ed SN table contains the proper corrections for the calibrator data while the CL table created using the 'ZRAT'-ed SN table contains the proper corrections for the line source.

Now re-run **FRING** to determine the residual fringe-rates for the line source, this time selecting a suitable line channel or group of channels:

- > **TASK 'FRING' ; INP C_R** to review the inputs.
- > **INDISK n ; GETN ctn C_R** to specify the input file.
- > **CALSOUR 'OH127.8', ' ' C_R** to select the spectral-line source.
- > **DOCALIB 1 C_R** to apply the previous amplitude and delay/rate calibration.
- > **GAINUSE clin** to specify which CL table to use.
- > **FLAGVER 0 C_R** to apply the most recent flag table.
- > **SNVER 0 C_R** to write a new solution table.
- > **BCHAN 72 ; ECHAN 72 C_R** to select the strongest and/or simplest spectral channel as a reference.
- > **REFANT 5 C_R** to try to use the same reference antenna as in the previous run of **FRING**.
- > **SOLINT 6 C_R** to set the solution interval in minutes. It should not exceed the coherence time.
- > **APARM(6) 1 C_R** to get useful, but limited printout.
- > **APARM(7) 9 C_R** to avoid false detections by setting the minimum acceptable SNR. *Warning:* solutions with lower SNR will be flagged as bad which will ultimately flag the affected data.
- > **DPARM(1) 1 C_R** to use one-baseline combination in initial, coarse fringe search (**FFT**).
- > **DPARM(2) = -1 C_R** to prohibit a search in the delay domain by setting the delay window to a negative number. Remember setting this to 0 means to use the Nyquist value, which is not what we want.

- > **DPARM(3) 0 C_R** to search for fringes over the full Nyquist fringe-rate window since we don't know where the fringes are.
- > **DPARM(4)= 0 C_R** to let **FRING** automatically determine the correlator integration time (if this fails, you will have to set the proper value here — see **DTSUM**).
- > **DPARM(5) 0 C_R** to do the least-squares solution.
- > **GO C_R** to do the fit.

Then run **CLCAL** again to apply these solutions to the previous calibration tables. You have then generated a full set of calibration tables and data. Remember that the calibration information for the continuum and line sources are stored in different CL tables.

9.5.7.11 Polarization-specific fringe-fitting

The phase and delay corrections obtained using RR and LL data can only remove R-R and L-L offsets between different antennas. There still may be R-L phase and multi- and single-band delay offsets. The R-L delay corrections can be determined using the RL and LR data and a task named **RLDLY** is available for determining these effects. This still leaves an overall R-L phase offset.

Once the cross-hand calibration has been completed, the instrumental polarization (otherwise known as the feed D-terms), can be determined. Several different methods are available for this purpose, implemented in the tasks **PCAL**, **LPCAL** and **SPCAL**. The last task is designed for polarization calibration of spectral-line data sets, but **PCAL** is now fully capable of doing a channel-dependent polarization solution. **PCAL** is discussed briefly below and further details for each of these tasks can be found in the appropriate **EXPLAIN** files.

The determination of the absolute polarization position angle is equivalent to the determination of the absolute R-L phase difference. For an unresolved source this can be measured in much the same way as for **VLA** data (see § 4.6) using task **RLDIF** to determine the cross-polarized phase on the polarization calibrator. Alternatively, a source with known polarization properties can be used, e.g., 3C286 or 3C279. For a resolved polarization calibrator a sum of Clean components is required to compare to the integrated polarization position angle as measured by the **VLA** or single-dish observations nearby in time. Once the R-L phase correction is known it is applied using task **CLCOR** with **OPCODE='POLR'** or task **RLCOR** (see § 4.6). See also VLBA Scientific Memo No. 26, "Polarization Angle Calibration Using the **VLA** Monitoring Program," by G. Taylor and S. Myers, October, 2000.

After feed calibration and the determination of the absolute polarization position angle, the final Stokes Q and U images can be formed directly. Note that task **PCNTR**, which is used for displaying polarization images, allows an arbitrary rotation of all polarization vectors under control of input adverb **ROTATE**. This is only necessary if **CLCOR** wasn't used to correct the R-L phase; also the polarization angle specified for **PCNTR** is half the R-L phase difference.

9.5.7.12 R-L delay calibration

RLDLY is a task which replaces **RUN** file procedures **VLBACPOL** and **CROSSPOL**. It determines R-L delay differences and produces an SN table which corrects for these effects (after **CLCAL**). It is best run on a set of calibrator data for which the source is at least moderately polarized (source polarization dominates instrumental polarization). Several baselines should be averaged but RL or LR fringes (or both) must be detectable on each baseline to the reference antenna. This task should leave a single R-L phase difference that must be determined from a calibrator of known polarization angle.

- > **DEFAULT RLDLY ; INP C_R** to set the task name and examine the inputs.

> INDISK <i>n</i> ; GETN <i>ctn</i> C_R	to specify the input file.
> OUTDI 1 C_R	to use disk 1 for temporary files.
> FLAGVER 0 C_R	to use the highest numbered flag table.
> GAINUSE <i>CLin</i> C_R	to use the CL table with all calibration up to this point; <i>no default</i> .
> SUBARRAY 0 C_R	to do all subarrays.
> BASELINE 0 C_R	to use all antennas.
> REFANT <i>A_{ref}</i> C_R	to select the reference antenna; any antenna may be used but all baselines to it should have RL and LR fringes. REFANT 0 will loop over all possible (not necessarily good) reference antennas, averaging the result.
> APARM 6, 1, 10 C_R	to use only solutions with signal-to-noise ratio above 6, to not write a CL table even if there is only one calibration scan, and to omit all solutions with rms < 10 from the average over antennas.
> CALSOR 'cal1' , ' ' C_R	to specify the calibrator source to use.
> TIMERANGE <i>d1 h1 m1 s1 d2 h2 m2 s2</i> C_R	to specify a time range with high SNR for RL and LR.
> SOLINT 0 C_R	to specify the minimum integration time in seconds; 0 causes it to be found from the data.
> GO C_R	to run the task.

RLDLY should be done after parallel-hand instrumental delays are removed (**VLBAPCOR**). It may be done before or, with a slight preference, after fringe fitting (**VLBAFRNG**, **VLBAKRNG**, **VLBAFRGP**, or **VLBAKRG**). The corrections should be checked with **VLBACRPL**, by setting **STOKES** to 'RL' and/or 'LR'. The RL and LR phases should be continuous across the bandpass on each baseline and be flat if the RR and LL phases are flat (no residual delays).

9.5.7.13 Feed D-term calibration

The feed D-terms, or instrumental polarization terms, can be determined using **PCAL**. **SOLTYPES** 'ORI-' and 'RAPR' are appropriate for VLBI data. 'ORI-' uses a non-linear orientation-ellipticity feed model and is appropriate if instrumental polarization exceeds a few percent (e.g., EVN, or VLBA at 18 or 13 cm). 'RAPR' uses a linearized "D-term" model — this is faster but less accurate.

Typical inputs for **PCAL** would be:

> TASK 'PCAL'; INP C_R	to review the inputs.
> INDISK <i>n</i> ; GETN <i>ctn</i> C_R	to specify the input file.
> CALSOR 'DA193' , ' ' C_R	to select the source.
> TIMERANG 0 C_R	to use all times.
> SELBAND 0; SELFREQ 0; FREQID -1 C_R	to select all frequencies.
> BIF 0; EIF 0 C_R	to select all IFs.
> ANTENNAS 0; UVRANGE 0 C_R	to select all antennas and baselines.
> SUBARRAY 0 C_R	to select all subarrays.
> FLAGVER 0 C_R	to apply the most recent flag table.
> DOCALIB 1 C_R	to apply the previous amplitude and delay/rate calibration.
> GAINUSE <i>cln</i>	to specify which CL table to use.
> IN2DISK <i>n</i> ; GET2N <i>ctn</i> C_R	to specify Clean images as models for the I, Q, and U polarizations.

> INVERS 1 C _R	to select a CC table version.
> REFANT 1 C _R	to use the same reference antenna as in the previous run of CALIB .
> NCOMP 77; NMAPS 1 C _R	to specify the number of Clean components to use and to explicitly set the number of Clean images supplied.
> SOLINT 6 C _R	to set the solution interval in minutes. It should not exceed the coherence time.
> SOLTYPE 'RAPR' C _R	to specify the type of feed model.
> PRTLEV 0 C _R	to print minimal information
> BPARM(1) 0 C _R	to use the initial feed model if found in the AN table.
> BPARM(3) 0 C _R	to not fit for R-L phase difference.
> BPARM(4) 0 C _R	to not specify the initial R-L phase.
> BPARM(5) 0 C _R	to not solve for Vpol.
> BPARM(6) 0; BPARM(7) 0 C _R	to solve for the orientations of both polarizations of the reference antenna.
> BPARM(8) 0 C _R	to solve for all orientations.
> BPARM(9) 0 C _R	to solve for all ellipticities.
> BPARM(10) 0 C _R	to fit for the source polarization model parameters.
> CPARM(1) 0 C _R	to find separate solutions for each IF.
> CPARM(8) 0 C _R	to not limit the number of iterations.
> CPARM(9) 0 C _R	to use the default convergence tolerance.
> CPARM(10) 0 C _R	to use default convergence criterion.
> BADDISK 0 C _R	to specify which disks to avoid for scratch.
> GO C _R	to run the program.

The instrumental polarization may vary rapidly with frequency and independent solutions may be necessary in each IF. Both 'ORI-' and 'RAPR' model the source Q and U as scaled versions of I which is generally only true in the limit of unresolved or unpolarized sources. The D-terms can be determined iteratively by subtracting estimates of source polarization (Q, U Clean components) in **UVSUB**. Note: this should be done on data for which instrumental polarization corrections have *not* been applied.

Tasks **LPCAL** and **SPCAL** also can be used to compute D-term corrections for continuum and spectral-line polarization data respectively. Be forewarned that both of these tasks use linearized D-term models.

9.5.8 Complex Bandpass

For spectral line experiments and continuum observations where a high dynamic range is required, it is a good idea to do a complex bandpass at this point. In § 9.5.4.5 a scalar bandpass was done; *i.e.*, only the amplitude was calibrated but not the phase. Now that the phases have been calibrated in time (by fringe-fitting), a complex bandpass may be solved for. This will take out any dependence of the phase on frequency. To do this, run **BPASS**, again applying the **CL** table that includes all the calibration. With the inputs we recommend here, the phases *must be stable in time over the entire bandpass calibrator scan(s)*. Check this with **VPLOT** before proceeding. Then

> TASK 'BPASS'; INP C _R	to review the inputs.
> INDISK <i>n1</i> ; GETN <i>ctn1</i> C _R	to select the multi-source visibility data as the input file.
> CALSOUR 'BLLAC', 'DA193' C _R	to specify the continuum source(s) which were observed for the purpose of bandpass calibration.
> DOCALIB 1 C _R	to apply calibration.

> GAINUSE <i>CLin</i> <i>C_R</i>	to indicate the CL table with all calibration up to this point.
> BPVER -1 <i>C_R</i>	do not apply previous bandpass table
> SOLINT 0 <i>C_R</i>	to average data over whole scans before determining the bandpass.
> BPASSPRM 0 <i>C_R</i>	to do a complex bandpass and set rest to 0.
> BPASSPRM(5) 1 <i>C_R</i>	to not divide by “channel 0.”
> BPASSPRM(9) 1 <i>C_R</i>	to interpolate over flagged channels.
> BPASSPRM(10) 6 <i>C_R</i>	to normalize the amplitude and phase of the bandpass solutions, using <i>power</i> rather than voltage for amplitudes.
> ICHANSEL <i>chan_{beg}</i> , <i>chan_{end}</i> , 1, <i>IF_{num}</i> , ... <i>C_R</i>	to set channels for entire bandwidth for normalization, if this left 0 then the inner 75% is used and can cause up to 15% error in the amplitude.
> GO <i>C_R</i>	to run the program.

As recommended in § 9.5.4.5, you should look at the bandpass with **POSSM** and **BPEDT**. If necessary, use **BPEDT** to flag channels in your bandpass calibration data and re-make the BP table. This is the bandpass that should be applied when the data is calibrated and averaged (*e.g.*, with **SPLIT**).

Note, if your bandpass calibrator is not stable or strong enough during the observation (your bandpass calibrator data should have a better S/N than the data you’re trying to correct with the bandpass), you could consider using the phase reference source (if there is one) and use **SOLINT** = -1 (include all scans to make one bandpass solution). If the bandpass calibrator is strong enough, but the average phase varies through the scan, then divide each record by “channel 0” by setting **BPASSPRM(5)** = -1 to adjust phase only. You should select a range of channels that have similar phases to be averaged as channel 0 using adverb **ICHANSEL**.

9.5.9 Baseline-based errors

Baseline-based non-closing phase and amplitude errors can limit the dynamic range of the final images. One way to proceed is to try to solve directly for the non-closing effects using bright, point-like calibrator observations and the task **BLCAL**. This task writes a BL table containing the estimated non-closing baseline-based errors which can later be applied in **SPLIT**, or any of the other calibration tasks. To use **BLCAL** the noise in the calibrator-source images should approach the theoretical limit. Furthermore, the signal-to-noise ratio in the visibility data must be at least 100:1 on baseline-averaged data. **BLCAL** will divide your data by your best model and then write a BL table containing the baseline-based corrections. Use this task carefully only after reading the **EXPLAIN** file thoroughly. As an example:

> TASK 'BLCAL' ; INP <i>C_R</i>	to review the inputs.
> INDISK <i>n1</i> ; GETN <i>ctn1</i> <i>C_R</i>	to select the multi-source visibility data as the input file.
> IN2DISK <i>n2</i> ; GET2N <i>ctn2</i> <i>C_R</i>	to select your best image as the input model file.
> SOURCE 'DA193' <i>C_R</i>	to select your calibration source.
> DOCALIB 1 ; GAINUSE <i>cln</i> <i>C_R</i>	to select the CL table to use.
> BLVER 1 <i>C_R</i>	to create BL table version 1.
> SOLINT 1440 <i>C_R</i>	to determine one complex gain correction per baseline for the whole observation.
> GO <i>C_R</i>	to run the program.

Set **ICHANSEL** to select only the most desirable spectral channels to average. An experimental task **BLCHN** is available to compute baseline-based errors on a channel-by-channel basis. It writes a BD table which can be plotted by **POSSM** and **BPLOT**, but the calibration itself is applied only in the output file produced by **BLCHN**.

9.6 After initial calibration

9.6.1 Applying calibration

Having obtained your best possible calibration CL table (and BP and BL tables if bandpass or baseline-dependent errors were found), you finally get to make a calibrated data set. This is done with [SPLIT](#), which applies the calibration and splits the database into separate files, one for each source observed.

> TASK 'SPLIT' ; INP C_R	to review the inputs.
> INDISK n ; GETN ctn C_R	to specify the input file.
> SOURCE '' C_R	to write all sources.
> DOCALIB 1 ; GAINUSE $clin$ C_R	to specify which CL table to use.
> DOBAND 1; BPVER 1 C_R	to apply BP table 1 if present.
> BLVER $blin$ C_R	to apply BL table $blin$ if present.
> APARM(1) 1 ; NCHAV 0 C_R	to have all spectral channels within each IF averaged: read the help file closely, other useful averaging options are available.
> APARM(2) 2 C_R	to have an amplitude correction made for the correlator integration time (in seconds)
> GO C_R	to run the program.

The options for [SPLIT](#) given above will apply calibration and then average the spectral channels within each IF, but not average IF channels together. To average over IF channels as well, set [APARM\(1\) = 3](#) in [SPLIT](#). (The task [AVSPC](#) no longer averages IFs although it is useful in averaging spectral channels in calibrated data sets.)

The task [SPLAT](#) can be used instead of [SPLIT](#) to do time averaging and different options in spectral averaging. [SPLAT](#) can be used also to assemble the selected sources into a multi-source file after applying the specified calibration and averaging. This option allows the user to continue calibration on a smaller data set.

At this point, it is well worth spending time to examine your output visibility data carefully. You may plot the data against time with [VPLT](#), [IBLED](#), [EDITR](#), or [UVPLT](#), and list them with [LISTR](#), [PRTUV](#), or [UVPRT](#). [POSSM](#) is now no longer useful since you have averaged your data in frequency.

9.6.2 Time averaging

It is now convenient to average the data in time using [UVAVG](#) both to reduce the bulk of the data and to increase the signal-to-noise for subsequent iterations of the self-calibration/imaging cycle. However, it is important to realize that the fringe-fitting process to this point has only removed gradients of phase over the fringe-fitting solution interval. There will still be stochastic atmospheric (and clock) phase errors affecting the data on short time scales. These phase errors can be significant over minutes at frequencies of 22 GHz and above (and possibly even at 15 GHz) and a reduction in amplitude can occur if data are directly averaged. The ionosphere can cause similar problems at lower frequencies. Self-calibration should remove such phase errors.

For data at frequencies below 15 GHz (and \approx minute integrations), it should be safe to proceed with [UVAVG](#) (see below). For higher frequency data, it may be worth your while to examine the phase coherence of the data first. [VPLT](#) can be used to examine your target or calibrator data to see directly the level of residual phase error over your chosen averaging time. Alternatively, the task [IBLED](#) allows you to view the degree of coherence of data averaged over different averaging times. If there are coherence problems (and the target data has enough SNR), [CALIB](#) can be run to align the phases prior to coherent averaging. Try:

- > **TASK 'CALIB'** ; **INP C_R** to review the inputs.
- > **INDISK n** ; **GETN ctn C_R** to specify the single-source input file.
- > **OUTNA INNA** ; **OUTCL 'ALIGN' C_R** to specify the output file.
- > **CALSORC ''** ; **SMODEL 1, 0 C_R** to use the source with a point-source model.
- > **DOCALIB -1** ; **GAINUSE 0 C_R** to not apply any tables to the input data.
- > **SOLTYPE '' C_R** to use normal least squares.
- > **SOLMODE 'P' C_R** to solve for phase.
- > **SOLINT (10.0/60.0) C_R** to solve for phase in 10-second intervals. This should probably be set as low as the strength of the source will allow. The limit is the integration time that gives a SNR > 2 on most baselines.

- > **ANTWT 1 C_R** to use weights from calibration with no additional weights applied to the antennas. For the purposes of phase alignment, it is appropriate to use the data weights; this allows the noise in the solution to be distributed over the noisiest baselines. This may not be the case when using **CALIB** for self-calibration in the hybrid mapping sense (see § 9.7).

- > **APARM(1) 3 C_R** to require 3 antennas present for solution.
- > **APARM(6) 0 C_R** to skip diagnostic printout.
- > **APARM(7) 1 C_R** to set the minimum allowed SNR. This limit should be low since the SNR is calculated as a phase difference from model and this can be large. Start with a value ≤ 1 .

- > **GO C_R** to run the program.

Note that **CALIB** will only give valid solutions if the signal-to-noise over the solution interval on most baselines is greater than 2 (and preferably much higher). At high frequencies on weak sources, it may not be possible to select a solution interval long enough that the signal-to-noise satisfies this criterion, yet short enough to follow the atmospheric phase variations. In such cases, it is probably best not to attempt to self-calibrate the data, but instead to use a short averaging time and to live with any coherence losses in the data.

When the data are sufficiently phase coherent, they should be averaged over time down to a reasonable size using **UVAVG**:

- > **TASK 'UVAVG'** ; **INP C_R** to review the inputs.
- > **INDISK n** ; **GETN ctn C_R** to specify the **CALIB** output file as the **UVAVG** input file.
- > **OUTNA INNA** ; **OUTCL 'UVAVG' C_R** to specify the output file.
- > **YINC 30.0 C_R** to set the time-averaging interval to 30 seconds.
- > **OPCODE '' C_R** to enable the averaging operation. There are several options controlling the averaging interval selection and reported times.

- > **GO C_R** to run **UVAVG**.

If **SPLAT** was used to assemble the selected sources into a multi-source file (while applying the preliminary calibration), **CALIB** will write an **SN** table which must be converted to a **CL** table by **CLCAL**. This new **CL** table can be applied by **SPLAT** with time averaging.

9.6.3 Verifying calibration

Before proceeding to image your data, it's worth checking that the calibration performed in § 9.5 is sensible. For each of your sources, produce a plot of the correlated flux density against *uv* distance using **UVPLT**. As well as identifying bad data which can then be deleted with **IBLED**, **EDITR**, **WIPER**, or **UVFLG**, these amplitude versus distances plots (especially those of your calibrator sources) can be used to identify stations where the amplitudes are too high or too low. Furthermore, by fitting simple models to the calibrator data, constant correction factors can be determined for each station which can be used to correct the amplitude calibration. It is often the case that the amplitude calibration to a certain station (particularly non-VLBA stations) is out by a constant factor, either due to uncertainties in the antenna gain or in the noise calibration signal.

Most VLBI calibrator sources can be adequately described by one or two Gaussian components. The task **UVFIT** can be used to fit such a model while finding constant correction factors for the antenna gains. Note that **UVFIT** can handle no more than 2,500,000 visibilities, so further averaging with **UVAVG** may be required. The following example shows how to fit antenna gains and a single elliptical Gaussian model of known position and flux to one of the single-source data sets produced by **SPLIT** or **SPLAT** (with spectral averaging)

> TASK 'UVFIT' ; INP C _R	to review the inputs.
> INDISK n ; GETN ctn C _R	to specify the input <i>uv</i> data set.
> OPCODE 'GAUS' ; NGAUS 1 C _R	to specify 1 Gaussian component.
> GMAX 1.2 ; DOMAX FALSE C _R	to fix the flux at 1.2 Jy.
> GPOS 0 ; DOPOS FALSE C _R	to hold the position fixed at the origin.
> ewidth 0.002 , 0.001 , 45. C _R	to provide an initial guess of the Gaussian widths as 2×1 mas at a position angle of 45 deg.
> DOWIDTH TRUE C _R	to fit for size.
> GAINERR 1 C _R	to fit for all antenna gains with initial guess = 1.
> NITER 50 C _R	to limit the fitting to 50 iterations.
> IMSIZE 0.0005 , 0.01 C _R	to limit sizes to be in the range 0.5 to 10 mas.
> DOCAT TRUE ; INVER 1 C _R	to save the solution in a CC file of version number 1.
> INP C _R	to check the inputs.
> GO C _R	to run UVFIT .

UVFIT can also be applied to the multi-source file, but **SOURCE**, **DOCAL**, **GAINUSE** etc. must then be set. It can handle up to 20 sources to be fit, using a channel-dependent input text file and can write the results in a compact text file form.

Another way to test your amplitude calibration is to use the task **UVCRS** for bright sources with long tracks. This task calculates correction factors for the amplitudes of the stations using regions of the *uv* plane where *uv* tracks cross. **UVCRS** can write the correction factors into an SN table.

Once the scale factors are determined, there are a number of options for correcting the data. The simplest option is to apply the correction factors to the single-source data sets using task **VBCAL**. Alternatively, the correction factors can be incorporated into the highest version CL table of the multi-source data file and task **SPLIT** run again to make new calibrated data files. The corrections to the CL table are done with **CLCOR**. Unlike **VBCAL**, **CLCOR** must be run separately for each antenna whose calibration you wish to alter; **ANTENNA** must be set to the antenna number you wish to change, **OPCODE** = 'GAIN', and **CLCORPRM**(1) set to the amplitude scale factor found in **UVFIT**; note that these are *voltage* gains. All higher values in the array **CLCORPRM** should be zero. The effect of the altered calibration can be viewed using **UVPLT** with **DOCAL** = 1. If it is satisfactory, **SPLIT** can be re-applied to the data. We recommend using **CLCOR** to perform such amplitude corrections. It now produces a new CL table each time it is used unless you specify both **GAINVER** and **GAINUSE** as having the same, non-zero version number.

Another way of incorporating amplitude corrections is to edit the calibration text files used by **ANTAB**. This

can be accomplished by setting the FT parameters for affected stations. For instance, if the Bonn scale factor is 1.043, set the FT parameter on the BONN TSYS card to FT=(1.043*1.043). If amplitude calibration was carried out *after* fringe-fitting (*not recommended!*, then it is only necessary to rerun [ANTAB](#), delete the latest CL table containing amplitude calibration and rerun [APCAL](#) using the highest CL table produced in the fringe-fitting step. If however, as we have described in this chapter, the amplitude calibration was done prior to fringe-fitting, then correcting the amplitudes is more involved. It is probably best to delete all CL tables except the first one and start again at § 9.5. However, it may not be necessary to carry out the time consuming [FRING](#) solutions again. If the amplitude changes are small, the phase, rate and delay solutions will be essentially unchanged. Therefore, with care, the existing SN tables can be used in lieu of re-running [FRING](#).

9.7 Self-calibration, imaging, and model-fitting

We are now in a position to make images. As with [VLA](#) data, we do this by iteratively self-calibrating the data and deconvolving using Clean, MEM *et al.* We describe below a typical self-calibration and imaging sequence for VLBI data. The tasks used are described in more detail in Chapter 5.

1. [CALIB](#) self-calibrates the *uv* data.
2. [IMAGR](#) images and Cleans. [IMAGR](#) is now the preferred task for imaging VLBI data in *AIPS*.
3. [SCMAP](#) images, Cleans, and self-calibrates. [SCMAP](#) is meant to provide the functionality of the popular VLBI data analysis package *difmap*. [SCIMG](#) is a multi-field version of [SCMAP](#).

The main difference between the processing of VLBI and [VLA](#) data is that, unless phase-referencing is used, absolute phases of VLBI data are un-calibrated. Therefore, many more iterations around the imaging loop are required for the VLBI case; dozens of self-calibration iterations are not uncommon. Given this, it may be convenient to use the procedure [HYB](#) which executes a whole cycle of hybrid mapping (*i.e.*, an [IMAGR](#) plus [CALIB](#)). It also plots images and allows editing of Clean component files prior to self-calibration. Type [HELP HYB CR](#) for more information. Note that it does not use the superior imaging algorithms available in [IMAGR](#), [SCMAP](#), and [SCIMG](#).

Note that VLBI imaging is not an exact science and there are a number of different views on the “correct” imaging method and the “correct” software to accomplish this method. Some users take their *AIPS* data into a package written at CalTech to use the *difmap* program (see <ftp://phobos.caltech.edu/pub/difmap/>). Others use the *AIPS* tasks [SCMAP](#) and [SCIMG](#) (see § 5.4), while still others follow the older [HYB](#) path. It is beyond the scope of this document to explain in detail all aspects of VLBI imaging. For more details, see Craig Walker’s chapter on “Practical VLBI Imaging” in the publication *VLBI and the VLBA*, 1995, edited by A. Zensus, P.J. Diamond, and P.J. Napier which is available on the World-Wide Web (www.cv.nrao.edu/vlbabook). Here we make a few suggestions on how to control [CALIB](#) and [IMAGR](#). Again, please note that the latter is preferable to all previous imaging tasks; [SCIMG](#) and to a lesser extent [SCMAP](#) offer the same improved imaging techniques. They do not offer several experimental algorithms found in [IMAGR](#) including Clean component filtering, the SDI Clean algorithm, and multi-resolution Cleaning.

9.7.1 CALIB

1. Start by correcting antenna phases only, *i.e.*, use [SOLMODE = 'P' CR](#). Switch on the amplitude correction only after you have converged to a fairly good image. On the first iteration, you will need to invent an input model. For most extragalactic continuum sources, a point-source model is a good choice. Set [SMODEL\(1\)](#) to the zero-spacing flux density as extrapolated by eye using [UVPLT](#) and

consider using a circular Gaussian model at the origin to reduce the impact of the longest spacings. Start with the so-called SNR parameter **APARM(7)** small (≤ 1) and gradually increase it as the image improves. If this parameter is large during early iterations, when the model used is far from correct, then large portions of your data in the output file may be flagged. This is not too important if you use the original input *uv* file as the input to **CALIB** in all iterations. You can also set **APARM(9) = 1** to leave data affected by failed solutions uncalibrated.

2. On subsequent iterations, use the Clean image as produced by **IMAGR** as your input model. VLBI applications usually require Clean components well beyond the first negative component to be used in calculating the source model. One possibility is to use **PRTCC** to find the point where a significant fraction (*e.g.*, one third) of all new Clean components are negative. An alternative is to use all of the Clean components, but to use tight windowing in **IMAGR** — which can now be done interactively on the TV as **IMAGR** progresses. Alternatively, use tight windowing and clipping of the Clean components in **IMAGR** (**IMAGRPRM(8)** and **IMAGRPRM(9)**) or afterward with **CCEDT** or **CCSEL** before running **CALIB**. Tight windowing is especially important when *uv* coverage is poor. Editing Clean components after **IMAGR**, but before **CALIB**, can be effective in removing possibly spurious features; if they are real they will usually reappear in later iterations. **IMAGR** offers a filtering option to remove weak, isolated components when requested from the TV and at the end before the components are restored to the image. This removes much of the need for **CCSEL**.
3. When carrying out the next **CALIB** iteration with the new Clean model, you can either self-calibrate the original data set or, alternatively, self-calibrate the data set (output by **CALIB**) which was used to produce that Clean model. It is advantageous to use the original data set at least until you turn on amplitude calibration. At that point, you should stick with the file produced from the original data with the best phase-only solution. Amongst other things, this can prevent the telescope amplitudes from “wandering” (see below).
4. When the model contains extended structure, there may be problems with convergence when weighting the data “correctly” (*i.e.*, by $1/\sigma^2$.) The **WEIGHTIT** adverb allows you to use less extreme weights, such as $1/\sigma$ or $1/\sqrt{\sigma}$. If your array contains antennas that have a wide range of sensitivities, *e.g.*, the VLBA plus the phased-VLA and/or the Effelsberg 100-m, it is helpful to alter the weights of the antennas in your **CALIB** solutions. If this is not done, then your solution will be dominated by only a few baselines and the uniqueness of the solution is not guaranteed. Use **PRTUV** to inspect the weights of your data. Then set the **CALIB** input array **ANTWT**, which provides multiplicative factors adjusting the weights for each antenna prior to the **CALIB** solution. Set these parameters so that the effective range of baseline weights is only 10 to 100. Alternatively, use **WTMOD** to raise the original weights to a power between 0.25 and 0.5.
5. As you iterate, keep an eye on how the model image is converging to fit the data. Use **VPLOT**, **CLPLT**, **CAPLT**, and **UVPLT**.
6. When your source has a lot of extended structure and/or your VLBI array has relatively few short spacings, you should consider setting **UVRANGE** to only include the range of spacings in which the model provides a good fit to the data. However, given the relatively small number of antennas in most VLBI observations, you may need to compromise to allow in enough baselines to get good self-cal solutions. Set **WTUV > 0**.
7. When you are finally ready to solve for amplitude corrections, you should first apply all previous phase calibration including the final phase-only self-calibration solution. Then run **CALIB** setting **SOLMOD 'A+P'**, initially setting the solution interval (**SOLINT**) to a longer time than used for phase-only solutions (*e.g.*, 3 times). Try to prevent the antenna amplitudes from “wandering,” which can sometimes happen if there is still a significant amount of short spacing flux density missing from the source model. Setting **UVRANGE** is useful, as is setting **CPARM(2) = 1** to constrain the mean amplitude solutions over all antennas to be one. You can also set **SOLMODE='GCON'** and the array **GAINERR** to the expected standard deviation of the gains for each antenna. This constrains amplitude solutions to conform to the expected statistics. Setting the gain constraint factor **SOLCON** to values larger than 1 will

increase the importance of these gain error constraints. Finally, going back and self-calibrating starting with the original data set and the best available Clean model is useful way to prevent amplitude wander.

CALIB has been enhanced to improve its usefulness for SVLBI data sets. This has involved the implementation of improved antenna selection and partial array calibration, through the new adverb **DOFIT**. The possibility of solving only for a subset of selected antennas has been implemented in this manner and may prove useful for SVLBI data. The implementation of adverb **DOFIT** in **CALIB** is analogous to its implementation in **FRING** (see § 9.5.7).

9.7.2 IMAGR, SCIMAG, and SCMAP

1. Before using **IMAGR** or **SCMAP** or **SCIMG**, print out and read the **EXPLAIN** file. They are powerful and complicated tasks with many adverbs — some of which are new — and shouldn't be used blindly.
2. The quality of images produced may depend on the type of weighting used. With VLBA-only experiments, the best quality images are often produced using natural (**UVWTFN** 'NA' C_R) weighting in **IMAGR**. These images will represent the extended structure of the source better. If the highest resolution is required, try uniform (**UVWTFN** 'UN' C_R) weighting. The **ROBUST** parameter allows weightings intermediate between these two extremes often with both good signal-to-noise characteristics and a narrow synthesized beam. It may also be worth experimenting with the **UVBOX** parameter to allow smoothing of weights over larger areas of the *uv* plane (*i.e.*, to use "super-uniform weighting"). If the array contains antennas with very different sensitivities, (for instance, if it includes Effelsberg, the phased-**VLA**, and/or HALCA), then it may be advantageous to alter the weights of baselines to these antennas. Although this increases the thermal noise in the image, it will improve the *uv* coverage, which, otherwise, will contain effectively only the baselines to the most sensitive antennas. One way of doing this is to use **UVWTFN** = 'UV' C_R in **IMAGR**. This option takes the fourth root of the input weights before applying uniform weighting. **SCMAP** and **SCIMG** also support these weighting options. Another flexible (but deprecated) approach is to use task **WTMOD** to change the weights in the data set prior to running **IMAGR**.
3. After an initial self-calibration against a point-source starting model, the deconvolved image will often show spurious symmetric structure. Convergence can be speeded up by placing Clean boxes **CLBOX**) around the side showing the brighter structure. Note that **CLBOX** may be used to produce circular as well as rectangular windows for use with **IMAGR**. Alternatively, **CCEDT** can be used to edit the Clean components after **IMAGR**, but before the next **CALIB**. **IMAGR**, **SCIMG**, and **SCMAP** allow this to be done interactively at the start of each major Clean cycle including the first.
4. Use **DOTV** = 1 C_R to view the residuals and possibly modify the Clean boxes as you Clean. You can stop Cleaning if you feel that you are including spurious structure into your model or if you feel you need to reset a Clean box to include a new feature. Note the **BOXFILE** and **OBOXFILE** options which allow you to retain interactively set Clean boxes for use in the next self-cal iteration.
5. **SCMAP** and **SCIMG**, at each self-calibration cycle, offer a powerful interactive data editing tool which displays input and residual data from up to 11 baselines simultaneously. This is the same editor as found in task **EDITR**.

The hints outlined above are by no means the whole story when it comes to self-calibrating and imaging VLBI data. Unfortunately, it can still be somewhat of an art form. Very experienced users can produce noise-limited images, but there is no simple recipe that will enable inexperienced users to do the same.

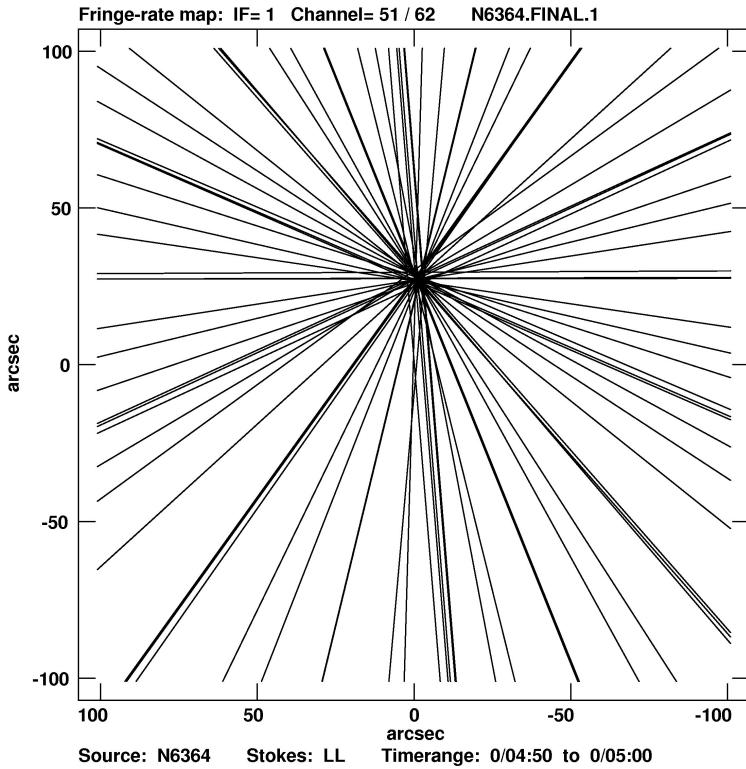


Figure 9.3: An example of a preliminary determination of the source location using fringe-rate analysis. The position of the source in channel 51 is shifted relative to the reference channel (62) by ≈ 25 asec to the north. Conventional imaging can be carried out near this position.

9.7.3 Non-conventional methods of imaging

Since fringe rate is a function of source position, we can use measurements of the fringe rate to estimate the positions of sources. Fringe rate imaging is widely used with maser sources which are usually widely separated point objects. The angular resolution of fringe rate images is not as good as conventional aperture synthesis imaging, but it can be used for an initial determination of the location of emitting clusters. This will help in setting field shifts and Clean windows for use by [IMAGR](#). Having measured the fringe rate $FR(i)$ for each sample in (baseline-time) and computed the derivative of fringe rate ($UDOT(i)$) wrt to right ascension (X) and the derivative ($VDOT(i)$) wrt declination (Y), we obtain a set of equations of straight lines

$$FR(i) = UDOT(i) * X + VDOT(i) * Y$$

describing the loci of constant fringe rate for each observation. The positions of the source component(s) can be determined by finding the places of highest density of crossing lines. (Details can be read in Giuffrida,T.S, 1977 Ph.D.thesis MIT and in Walker R.C., 1981, *Astr. J.*, **86**, 9, 1323.)

This algorithm is implemented in *AIPS* as the task [FRMAP](#). The task plots the straight lines on the TV or prepares a plot file. Then it determines the positions of higher density of crossing lines. The coordinates in RA and DEC of the components found are written in an output file. The fringe rate method is especially attractive in SVLBI because of the better angular resolution due to faster movement of the orbiting antenna. Figure 9.3 shows an example of a fringe rate image.

[FRMAP](#) can be used for more accurate definition of the source coordinates that is required by the correlator. In this case the correlator carries out two passes. After the first one (short) [FRMAP](#) is used to find a more

accurate position of the source. The new coordinates are then used in the second pass covering the whole experiment.

9.8 Summary of VLBI calibration tables

Several different tables are supplied with a VLBI data set created by the Socorro and other correlator; various other tables (*e.g.*, SN tables) are created by the calibration process. A list of these tables is given below for your edification. Not all tables will be present with all files.

AN Antenna table. Contains a list of the antenna names and station coordinates. Also contains instrumental polarization terms.

AT Antenna characteristics table. Contains additional information about antenna properties, including some time variable quantities.

BL Baseline offset table. Contains non-closing baseline-dependent phase and amplitude errors as determined by [BLCAL](#).

BS Baseline solution table. Contains baseline delay, rate and phase solutions as determined by [BLING](#).

CL Calibration table. Version 1 contains, amongst other things, the default calibration parameters for the amplitude (usually unity), phase, and single-band delay (usually zero) for each source for each IF as a function of time. It also contains polynomial coefficients allowing the correlator delay and phase models to be recomputed. As calibration proceeds, higher versions of this table are created which incorporate more and more calibration effects into the phase, delay, and amplitude entries.

CQ Correlator parameter frequency table. Contains VLBA correlation parameters for each *AIPS* IF, and activates VLBA delay decorrelation corrections. Type [EXPLAIN FXVLB](#) for further information.

CT [CALC](#) table. Contains the input parameters passed to [CALC](#) to generate the polynomials recorded in the **IM** table.

FG Flag table. Contains information used to delete selected portions of the data.

FQ Frequency table. Contains information about the IF frequencies, channel spacings, bandwidths, etc.

GC Gain Calibration table. Contains the expected zenith gain and gain-elevation curve for each antenna. It is used for amplitude calibration.

HF Haystack FRNGE table. Contains information generated from the *AIPS* tables that can be exported to the [CALC](#) and [SOLVE](#) package.

IM Interferometer model table. Contains the actual polynomial coefficients which the VLBA correlator used to calculate the geometrical model. Unlike the coefficients in the CL table, these have not been re-interpolated onto the CL time grid, but have time stamps corresponding to the times at which the correlator computed the geometrical model.

MC Model Components table. Contains the various components of the geometric model used in the VLBA correlator to generate the **IM** table.

NX Index File. Contains information about the time, source, sub-array and location within the data file of each observation or “scan.” It is used by some *AIPS* tasks in accessing the main data file and subsidiary tables.

OB Spacecraft Orbit table. Contains information about the positions and velocities used by the correlator for an orbiting antenna.

PC Phase-calibration table. Contains phases within each IF computed from the injected phase calibration signals. It is used to determine the phase offsets and single-band delays for each IF channel.

SN Solution table. Contains antenna delay, rate, phase, and amplitude corrections solved for by **CALIB** and **FRING** and other tasks.

SU Source table. Contains a list of the sources found within the multi-source file, including information on source positions and flux density.

TY System temperature table. Contains the system temperature as a function of time for each antenna and IF channel. It is used for amplitude calibration.

VT VLBA Tape table. Contains tape playback statistics for use mainly by the VLBA correlator group.

WX Weather table. Contains weather-related information for each station.

9.9 Additional recipe

9.9.1 Banana relish

1. Cut 12 **bananas**, 1 pound **dates**, and 2 pounds **Bermuda onions** into small pieces.
2. Add 2/3 cup **molasses**, 1/2 teaspoon ground **ginger**, 1 teaspoon **salt**, 1 teaspoon **allspice**, 1 cup **water**, and 2 cups **vinegar**; mix well.
3. Turn into a large stone jar or crock, bake in a slow oven till rich brown, seal in jars while hot.

9.9.2 Cranberry banana bread

1. In a large saucepan, bring 2 cups **sugar** and 1 cup **water** to a boil, stirring to dissolve the sugar. Add 4 cups fresh **cranberries** and simmer over low heat for 10 minutes or until berries pop open. Cool. Drain the berries, reserving the juice and measuring 1 cup of berries for use in the bread.
2. Sift together 1 3/4 cup **flour**, 1/2 teaspoon **salt**, 2 teaspoon **baking powder** and 1/4 teaspoon **baking soda**.
3. In a large bowl, combine 2/3 cup **sugar**, 1/3 cup melted **butter**, 2 beaten **eggs**, 1/2 cup chopped **walnuts**, 1 cup mashed **banana**, and 1 cup cooked berries.
4. Add the flour mixture to the berry mixture, stirring until blended. Pour the mixture into a greased and lightly floured 9 x 5 x 3-inch loaf pan. Bake in a preheated, 350° F oven for 1 hour or until a toothpick inserted in the center comes out clean.
5. For a topping (optional), combine 1/4 cup **cranberry juice** from cooked berries, 2 tablespoons **sugar** and 2 tablespoons **Grand Marnier** in a small saucepan and stir over low heat until heated through. Poke a few holes in the baked loaf and pour on the topping.
6. Cool 10 minutes in the pan. Turn the loaf out on a rack and cool completely. Wrap in foil and store one day before slicing.

Thanks to Tim D. Culey, Baton Rouge, La. (tsculey@bigfoot.com).

9.9.3 Churros de Plátano

1. Heat about 1 inch of salad (or part salad and part olive) **oil** in a large frying pan.
2. Peel and split 3 large, green-tipped **bananas** lengthwise. Then cut each piece in half and dip in **lemon juice**.
3. Separate 4 **eggs**. Beat the egg yolks until thick and light. Then add 1/4 cup **flour** and 1/2 teaspoon **salt**.
4. Beat the egg whites until stiff, but not dry, and fold into yolk mixture.
5. Drop the drained banana pieces one at a time into the batter. Pick up with a spoon and slide into the hot oil.
6. Cook over medium heat, turning almost at once, until brown on both sides. Drain on paper towels.

9.9.4 Roasted turkey quesadillas with banana

1. Place 6 corn or whole wheat flour **tortillas** flat.
2. Sprinkle with 6 ounces grated low-fat **Jack** or **cheddar cheese**, 2 tablespoons chopped fresh **cilantro** or **parsley**, 1/2 pound shredded roasted **turkey** or **chicken** meat, 2 seeded and minced **jalapeño peppers**, 1 cup **alfalfa sprouts**, and 2 medium **bananas**, sliced into thin circles.
3. Place 6 **tortillas** on top and press firmly.
4. Place on a lightly oiled cookie sheet; cover with another cookie sheet of similar size. Bake in a pre-heated 350° F oven for 15 minutes until soft and melted. Cut into wedges and serve with hot sauce and salad.

Thanks to Chiquita Bananas. See <http://www.jaetzel.de/tim/chiquit.htm>.

9.9.5 Banana Bombay salad

1. Puree 3 **bananas**.
2. Whisk with 1/4 cup **lemon juice**, 1/4 cup **mayonnaise**, 1/4 cup **plain yogurt**, and 1/8 – 1/4 ounce **taragon**. Refrigerate at least 2 hours.
3. Cut 2 pounds cooked **turkey** or **chicken breast** into bitesize pieces.
4. Add 1/2 cup **raisins**, 3 **green apples** cut into pieces, and 1/2 cup chopped **walnuts**. Mix.
5. Add banana puree and mix. Cut 2 **bananas** into thick chunks and add. Serve chilled.

Thanks to Chiquita Bananas. See <http://www.jaetzel.de/tim/chiquit.htm>.

10 Single-Dish Data in *AIPS*

AIPS was not originally intended as a reduction package for single-dish data and cannot be considered as such today. However, because of the similarity of single-dish data taken at “random” pointings on the sky to interferometric data taken at “random” locations in the *uv* plane, *AIPS* was seen as a system to be used to solve large imaging problems arising from single-dish observations. Many of the *AIPS* *uv*-data tasks are able to do something sensible — or even desirable — with single-dish data and a few special tasks to process single-dish data have been written. The present chapter contains a discussion of the representation of single-dish data in *AIPS* followed by a description of how such data may be calibrated, corrected, converted into images, and analyzed by *AIPS*. A final section on using single-dish observations to improve the imaging of interferometric data represents what little we now know about this potentially important process.

10.1 *AIPS* format for single-dish data

Single-dish data in *AIPS* is treated as *uv* data with different, but related random parameters and with the imaginary part of the visibility replaced by an additive calibration or offset. The *u* and *v* random parameters are replaced by parameters labeled RA and DEC, although other labels such as ELON, ELAT, GLON, and GLAT are also recognized. (Conversion between these coordinate systems is not provided in the “*uv*” plane although some conversion can be done on images.) The random parameter data are the sample coordinates in degrees. The TIME1 random parameter is the time (IAT) since midnight on the reference date in days as with real *uv* data. The BEAM random parameter corresponds to **BASELINE** and is used to separate data which should be edited and calibrated separately (*e.g.*, separate beams of a multi-feed system, different polarizations or observing runs of a multi-polarization system). The actual beam number is recorded as 257 times the desired number so that visibility-data tasks will recognize the “baseline” as auto-correlation data. Two other random parameters, **SCAN** and **SAMPLE**, have no relation to any visibility parameters and are simply used to retain the “scan” number and sample number within the scan which are traditional in single-dish observations. Very little is made of these, but **INDXR** will make a new index entry when the scan number changes and **PRTSD** will display the scan and sample numbers. **SDVEL** uses the scan numbers to determine when to update the reference velocity in some observing modes. Single-dish data may be stored in compressed form, in which the weight and compression scale are stored as random parameters exactly as in true visibility data. This should *not* be done if the applied offset is large as in beam-switched continuum observations.

The measured single-dish flux, usually in units of degrees Kelvin, appears in the real part of the “complex visibility.” The imaginary part of the visibility is sometimes used to hold an offset which can be applied to the data to remove, for example, a time-variable bias. The data weight is used to weight the data and should be proportional to σ^{-2} , where σ is the uncertainty in the flux. The visibility sample can contain multiple polarizations, described with the **STOKES** axis (values 1 through 4 for I, Q, U, and V, respectively). The sample can also contain multiple spectral-line frequencies, described with a **FREQ** axis giving the observed reference frequency and increment in Hz.

The *uv*-data header in an *AIPS* data set is expected to contain the reference (usually central) longitude and latitude given either as 1-pixel coordinate axes and/or in the “observed” coordinate location. The convolution size is usually used to hold the single-dish beam width (fwhm) and rest frequency and velocity information should also appear with spectral-line data. Many of the parameters can be added to the header by the user if they are missing and needed. Verbs **ADDBEAM**, **ALTDEF**, and **PUTHEAD** are useful for this purpose. A complete data set will also have an antenna extension file giving the location of the antenna. This allows tasks to compute things like zenith angles and Doppler corrections when needed.

10.1.1 On-the-fly data from the 12m

At the present time, the only reliable routes for single-dish data into *AIPS* are provided by the tasks [OTFBS](#) and [OTFUV](#). These tasks work only on beam-switched continuum and on spectral-line observations, respectively, from the NRAO 12m telescope. They use files in the UniPops native format and do not read the FITS table format written by UniPops. Both programs are designed for “on-the-fly” or “OTF” observing modes in which the telescope takes data rapidly while continuously changing its pointing position.

10.1.1.1 Listing OTF input files

To read OTF files, you must first define an environment variable to point to the disk area in which your data resides. This environment variable and your file names should be in upper case letters, but there is an AIPS “feature” which allows you to use lower case. On Unix systems, you may set the environment variable and rename the files to upper case with

% cd /my/disk/directory <i>C_R</i>	to switch to the disk directory containing your data.
% setenv MYAREA `pwd` <i>C_R</i>	to define \$MYAREA under c shell, or
% export MYAREA='pwd' <i>C_R</i>	to define \$MYAREA under Bourne, bash, korn shells.
% mv mysdd.file MYSDD. <i>FILE</i> <i>C_R</i>	to rename the data file to upper case letters.
% mv mygsdd.file MYGSDD. <i>FILE</i> <i>C_R</i>	to rename the gain file to upper case letters.

Then start your AIPS session.

To review the contents of your data set, use the task [OTFIN](#) which will list SDD modes, IF and scan numbers, times, coordinates, velocities, and number of samples. This output should help in setting the range of scan numbers to be loaded by [OTFUV](#) or [OTFBS](#). Type:

> TASK 'OTFIN' ; INP <i>C_R</i>	to list the required inputs on your screen.
> DATAIN 'MYAREA:MYSDD. <i>FILE</i> ' <i>C_R</i>	to specify the name of the 12m raw data file, where MYAREA is an environment variable which points at a disk data area and MYSDD. <i>FILE</i> is the name of your file in that area. See § 3.10. If your environment variable and/or your file name contain lower case letters, type the name carefully with the correct case for all letters and leave off the second (close) quote mark. When you use this “feature” of the AIPS compiler, you cannot type anything following the DATAIN name (or other string adverb) on that line.
> BCOUNT 0 ; ECOUNT 0 <i>C_R</i>	to include all 12m scans in the file.
> BIF 0 <i>C_R</i>	to include all SDD “IFs.”
> DOCRT -1 <i>C_R</i>	to print the listing on the line printer, or
> DOCRT 1 <i>C_R</i>	to view the listing, one page at a time, on your terminal window. The width given (if > 72) should match the width desired; a width of < 72, as given here, uses the actual width of the window and so maximizes the information per line.
> INP <i>C_R</i>	to review the parameters.
> GO <i>C_R</i>	to run the task.

10.1.1.2 Reading spectral-line OTF files into AIPS

To run **OTFUV** after running **OTFIN**, type

```
> TASK 'OTFUV' ; INP CR
> DATAIN 'MYAREA:MYSD.DFILE' CR
> DATA2IN 'MYAREA:MYGSDD.FILE' CR
> BCOUNT n1 ; ECOUNT n2 CR
> BIF 0 ; EIF 0 CR
> DOUVCOMP TRUE CR
> XINC 1 ; YINC 1 CR
> DOWEIGHT 1 CR
> DETIME 0 CR
> BCHAN 0; ECHAN 0 CR
> CHANSEL 0 CR
> INP CR
> GO CR
```

to list the required inputs on your screen.

to specify the name of the 12m raw data file, where MYAREA is an environment variable which points at a disk data area and MYSDD.FILE is the name of your file in that area. See § 3.10.

to specify the name of the 12m gain file corresponding to the file specified with **DATAIN**.

to include 12m scans *n*₁ through *n*₂ in the output file.

to include all SDD “IFs” matching the lowest numbered one found. IFs which do not match in central frequency or channel width are skipped.

to write the data in a compressed format. This reduces the size of the file by nearly a factor of 3 with no significant loss of information in this case.

to write out all data samples with no time averaging. One can smooth by **YINC** samples and write out the data every **XINC** sample times in order to reduce the size of the output data set and improve the signal-to-noise of the individual samples with only a minor loss of information..

to use offs and gains interpolated to the time of each observation. This seems to produce better results.

to add no offset to the actual observation times.

to include all spectral channels.

to flag no channels. **CHANSEL** 31,34,3 C_R, for example, would mark channels 31 and 34 as bad. Data may be edited later more selectively.

to review the parameters.

to run the task.

While **OTFUV** runs, it will show you (on the message monitor or your window) the name and location of the output **AIPS** file created and then provide a list of the scans and IFs read and the gain scans used upon them.

In many cases, the 12m in OTF mode observes two separate polarizations using the same center frequency and spectral resolution. In the UniPops/12m nomenclature, these are separate “IFs.” A similar nomenclature is used to distinguish the feeds in the multi-feed system. **OTFUV** can now read up to eight IFs at the same time, avoiding the necessity of multiple runs of **OTFUV**, followed by a data sort to restore time order. **OTFUV** will distinguish the IFs not by an **AIPS** “IF axis,” but by assigning them beam numbers equal to the SDD IF number (or autocorrelator baseline number equal to the SDD IF number with itself).

You may append data from another IF in the first input data set or data from another OTF pass on the source to the **AIPS** data set created above, by entering new **DATAIN** and **DATA2IN** names and new **BCOUNT** and **ECOUNT** ranges, if needed and

```
> BIF m1 ; EIF m2 CR
> DOCONCAT TRUE CR
> OUTDISK n ; GETONAME m CR
```

to load IFs *m*₁ through *m*₂.

to enable the concatenation mode.

to select the output file, where *n* and *m* are the output disk and catalog slot number used by the first run of **OTFUV**.

> **FQTOL** *ff* **C_R** to allow data sets within *ff* MHz of each other to be concatenated. Doppler tracking will cause two OTF passes to appear to be at separate frequencies. Narrow-band, wide-field observations should not be concatenated in this way; see the discussion of **SDVEL** below (§ 10.2.4).

> **GO** **C_R** to run the task appending the additional data.

Another way to concatenate two 12m IFs — or multiple observing runs — is to create two output files with **OTFUV** and then concatenate them with **DBCON**. If the two **OTFUV** files are in time order, then **DBCON** will actually merge the two data sets, retaining the time order. Avoid the use of multiple sub-arrays, which are a useless complication in this case, by setting **DOARRAY** = 0. To have the most “complete” antenna file, put the data set with the higher 12m IF in the first input name set (**INNAME**, **INCLASS** etc.)

10.1.1.3 Reading continuum OTF files into *AIPS*

The NRAO 12m telescope can observe in a beam-switched continuum on-the-fly mapping mode. Such data may be read into *AIPS* and reduced, in a somewhat experimental fashion, into images. To read in the data (after using **OTFIN**), enter

> TASK 'OTFB' ; INP C_R	to list the required inputs on your screen.
> DATAIN 'MYAREA:MYSD.DFILE' C_R	to specify the name of the 12m raw data file, where MYAREA is an environment variable which points at a disk data area and MYSDD.DFILE is the name of your file in that area. See § 3.10.
> BCOUNT <i>n₁</i> ; ECOUNT <i>n₂</i> C_R	to include 12m scans <i>n₁</i> through <i>n₂</i> in the output file.
> BIF 0 ; EIF 0 C_R	to include all SDD “IFs” matching the lowest numbered one found. IFs which do not match in central frequency or channel width are skipped.
> INP C_R	to review the parameters.
> GO C_R	to run the task.

While **OTFB** runs, it will show you (on the message monitor or your window) the name and location of the two output *AIPS* files created (one for “plus” and one for “minus” beam throws) and then provide a list of the scans and IFs read with the number of samples. The two output files will have the same names except for a “+” and a “-” as the sixth character of the output class.

10.1.2 Other input data formats

Another method for getting single-dish data into *AIPS* is through the use of FITS-format binary tables. If the data are able to be put in a usable table, then the *AIPS* FITS reading tasks such as **FITLD** (see § 5.1.2) can be used to read them into a disk table attached to a catalogued file. Then **SDTUV** can be used to convert the table into the *uv* format described above applying a variety of calibrations along the way. Unfortunately, the non-*AIPS* program that did the UniPops to FITS conversion has been lost and the *AIPS* FITS readers cannot handle the FITS tables written by UniPops. There are two problems with the latter: *AIPS* is unable to handle tables with more than 128 columns while UniPops writes tables with around 200 columns. Even if *AIPS* could be extended in some special task, it would be unable to handle the current UniPops tables since the parameters given do not correctly describe the contents. Specialized unpublished knowledge about each receiver is required to disentangle the coordinate information and data structure.

The task **SDTUV** expects a sequence of related tables each with a number of keywords giving useful information such as scan, observer, telescope, object, scan start UT date and time, sample rate, velocity, and the like. The data are then a regular time sequence with each row of the table containing the right

ascension, declination, and data for N receivers. Breaks in the time sequence are assumed to be new scans found in the next table. [SDTUV](#) has the ability to apply receiver position offsets and pointing corrections and to fit and remove receiver baselines using a sliding median window and spline fit. Interference rejection, lateral defocusing corrections, and a priori baseline removal are also offered. At present [SDTUV](#) is an example of what can be done rather than a directly usable task. It is limited to continuum problems currently and is moderately restricted in the number of data samples that can be read in any one scan.

Therefore, it will be necessary to write some sort of program in addition to those in the standard [AIPS](#) release to get single-dish data into [AIPS](#). We encourage anyone who develops such a program to provide it to the [AIPS](#) group so that we may offer it to other single-dish users.

10.2 Single-dish data in the “uv” domain

Once you have gotten your data into [AIPS](#), a wide range of tasks become available to you. In addition to the single-dish specific tasks discussed below, these include data movement tasks ([UVCOP](#), [UVSRT](#), [DBCON](#)), data averaging ([AVER](#), [UVAVG](#), [AVSPC](#)), non-interactive editing ([CLIP](#), [UVFLG](#)), interactive editing ([SPFLG](#), [EDITR](#), [TVFLG](#)), data backup and restore ([FITTP](#), [FITLD](#)), and data display ([PRTAN](#), [PRTUV](#), [UVPRT](#), [UVPLT](#)).

10.2.1 Using PRTSD, UVPLT, and POSSM to look at your data

In the process of calibrating, modeling, editing, and imaging of single-dish data, there are occasionally problems that seem to arise because users are not aware of the data that they actually have. [PRTSD](#) is the task for such users. It displays the data with or without calibration for selected portions of your data set. This will help you identify what pointing positions actually occur in your data, which channels are highly variable or bad, and the like. [SPFLG](#), [UVPLT](#), and others are good for looking at the data set as a whole, but [PRTSD](#) really shows you what you have.

To run it, type:

- > **TASK 'PRTSD'** ; **INP C_R** to list the required inputs on your screen.
- > **INDISK n** ; **GETN ctn C_R** to select the single-dish “uv” file to be displayed.
- > **DOCRT 1 C_R** to select the on-screen display at its current width; make sure your window is at least 132 characters across for the best results.
- > **DOCELL -1 C_R** to look at the data values; **DOCELL > 0** causes the offsets that have been removed (usually 0) to be displayed.
- > **CHANNEL m C_R** to display channels m through $m + 5$.
- > **DOCAL FALSE C_R** to apply no calibration. Note that the 12m off scans and instrumental gains are applied by [OTFUV](#); this parameter applies only to any additional calibration contained in CS files. See § 10.2.3.
- > **TIMERANG 0 C_R** to look at all times.
- > **ANTENNAS a₁, a₂, ... C_R** to look at beams/IFs a_1, a_2, \dots only.
- > **BPRINT bb C_R** to begin the display with the bb^{th} sample in the data set *before* application of the other selection criteria ([TIMERANG](#), [ANTENNAS](#), etc.)
- > **NPRINT 2000 C_R** to shut off the display interactively or after a lot of lines.

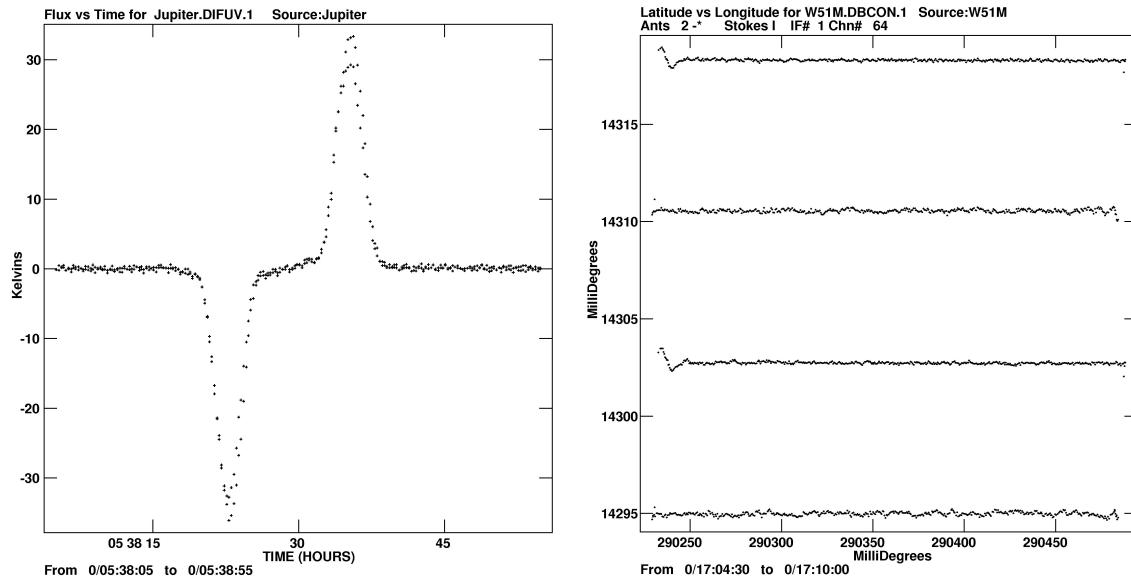


Figure 10.1: *left:* **UVPLT** display of 12m beam-switched continuum data on Jupiter. The time range is set to display one row of the OTF observation and the “minus” beam throw data have been subtracted from the “plus” throw. *right:* **UVPLT** display of the right ascension and declination of each sample in spectral-line OTF data set over a limited time range.

- > **XINC** x C_R to display only every x^{th} sample of those selected by the other criteria.
- > **INP** C_R to review the inputs.
- > **GO** C_R to start the task.

PRTSD will start and, after a pause to get through any data not included at the start of the file, will begin to display lines on your terminal showing the scan number, time, coordinates, and data for six spectral channels. After 20 or so lines, it will pause and ask if you want to continue. Hit C_R to continue or type **Q** C_R or **q** C_R to quit. If you decide to get hard copy, set **DOCRT** = -1 and the output will be printed. To save the display in a text file, without printing, set **DOCRT** = -1 and give the name of the file in the **OUTPRINT** adverb. See § 3.2 and § 3.10.1 for more information on printing.

There are a number of tasks which plot *uv* visibility data; see § 6.3.1. The most basic of these is **UVPLT**, which can be useful for single-dish data sets. For example, to generate the plot of flux versus time in 12m OTF beam-switched continuum differenced data seen in the accompanying figure (Figure 10.1), the parameters given below were used:

- > **TASK** 'UVPLT'; **INP** C_R to review the inputs.
- > **INDI** n ; **GETN** ctn C_R to select the disk and catalog entry of the data set.
- > **DOCALIB** FALSE C_R to apply no calibration; **UVPLT** does not understand single-dish calibration.
- > **BPARM** = 11,9,0 C_R to plot time in hours on the x axis and flux in Kelvins on the y axis. The other parameters can be used to specify fixed scales on one or both axes, but are just self-scaled in this example.
- > **XINC** 1 C_R to plot every selected sample.
- > **BCHAN** 1; **ECHAN** 1 C_R to plot only “spectral channel” 1, the actual data values.
- > **ANTENNA** 1,0; **BASELINE** 0 C_R to do all baselines with antenna 1, namely 1–1 or, in 12m nomenclature, IF 1..
- > **TIMER** = 0, 5, 38, 5, 0, 5, 38, 55 C_R to restrict the times to a single scan.

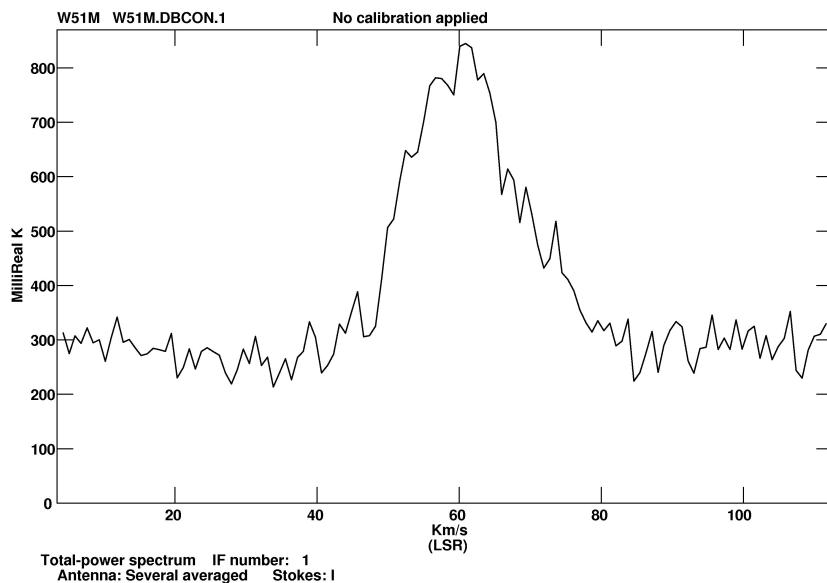


Figure 10.2: POSSM display of all of a 12m observation taken on W51. All samples (on and off the actual source) and both “antennas” are averaged together.

> **DOCRT** = -1 ; **GO** C_R to make a plot file of these data.

After **UVPLT** is running, or better, after it has finished:

> **PLVER** 0 ; **GO** LWPLA C_R to plot the latest version on a PostScript printer/plotter.

The second plot in Figure 10.1 was generated with **BPARM** = 6, 7 and shows where samples occur on the sky in a different data set.

With spectral-line data, **POSSM** will plot observed spectra averaged over selected “antennas,” time ranges, and the like. Thus,

- > **TASK** 'POSSM' ; **INP** C_R to review the inputs.
- > **INDI** n ; **GETN** ctn C_R to select the disk and catalog entry of the data set.
- > **BCHAN** 0 ; **ECHAN** 0 C_R to plot all channels.
- > **ANTENNAS** 0 ; **BASELINE** 0 C_R to average all 12m IFs.
- > **TIMERA** 0 ; **SOLINT** 0 C_R to average all times into one plot.
- > **APARM(7)** = 2 C_R to have velocity labels on the x axis.
- > **GO** C_R to run the task.

LWPLA was then used to make a PostScript version of the plot seen in Figure 10.2.

10.2.2 Using UVFLG, SPFLG, and EDITR to edit your data

Editing is the process by which you mark data samples as “unreliable” or “bad.” In *AIPS*, there are two methods for doing this. The simplest is to have the editing software alter the weight of the sample to indicate that it is flagged. If the data are not compressed, this is a reversible operation. If the data are compressed, however, then the data themselves are marked as “indefinite” and the operation is not reversible. The second method is the use of a flag (FG) extension table attached to your *uv* data set. This method requires that the data be sorted into time order for large FG tables and is supported by most, but not all, tasks. Small (< 6000 row) FG tables may be used with data in any sort order. If the task does not

have the **FLAGVER** adverb, then it does not support flag tables. However, since flag tables can be applied to the data by **SPLIT**, we use them in the recipes below.

To sort the data into “time-baseline” (TB) order,

> TASK 'UVSRT' ; INP C_R	to review the inputs.
> INDI n ; GETN ctn C_R	to select the disk and catalog entry of the data set.
> SORT 'TB' C _R	to sort into time-baseline order.
> ROTATE 0 C _R	to avoid damage to the coordinates.
> INP C_R	to check the parameters, <i>e.g.</i> , the output name.
> GO C_R	to run the task.

The most direct flagging task is **UVFLG**, which puts commands into the flag table one at a time (or more than one when read from a disk text file). To use this task to flag channel 31 from 7 to 8 hours on the first day of observation from the second input (single-dish nomenclature) IF:

> TASK 'UVFLG' ; INP C_R	to review the inputs.
> INDI n ; GETN ctn C_R	to select the disk and catalog entry of the sorted data set.
> OUTFGVER 1 C _R	to select the desired flag table.
> TIMERANG 0, 7, 0, 0, 0, 8, 0, 0 C _R	to set the time range from 7 to 8 hours.
> BCHAN 31 ; ECHAN 31 C _R	to flag only channel 31.
> BF 0 ; EIF 0 C _R	to do all <i>AIPS</i> IFs.
> ANTEN 2, 0 ; BASELIN 2, 0 C _R	to select “baseline” 2-2, the 2 nd IF in 12m nomenclature.
> APARM 0 C _R	to ignore amplitude in flagging.
> OPCODE 'FLAG' C _R	to flag the data.
> REASON 'Bad channel' C _R	to store away a reason.
> INP C_R	to check the full set of adverbs.
> GO C_R	to add one line to the flag table, creating one if needed.

Multiple runs of **UVFLG** may be done to incorporate what you know about your data into the flagging table. Use **PRTSD** and the plot programs to help you find the bad data. If you have a long list of flagging commands, you may find it easier to use the **INTEXT** option of **UVFLG** to read in up to 100 flagging instructions at a time from a free-format text file.

The task **CLIP** is popular on interferometer data sets since it automatically flags all samples outside a specified flux range without interaction with the user. This blind flagging is often acceptable for interferometer data since each *uv* sample affects all image cells so that the damage done by a few remaining bad samples is attenuated by all the good samples. However, a bad sample in single-dish data affects only a few image cells and is hence not attenuated. Thus it is important to find and remove samples that are too small as well as those that are too large. For this reason, we do not recommend **CLIP**, but suggest that you look at your data and make more informed flagging decisions.

The best known of the interactive editing tasks is **TVFLG** (§ 4.4.3). This task is not suitable for single-dish data since it displays multiple baselines along the horizontal axis. The data on these baselines are related in interferometry, but, in single dish, they are from separate feeds or polarizations and hence neither numerous nor necessarily related. For spectral-line single-dish data, the task **SPFLG** is an ideal task to examine your data and to edit portions if needed. **SPFLG** is a menu-driven, TV display editing task in which spectral channel varies along the horizontal axis of the TV display and time along the vertical. (The spectral channels for each interferometer IF are displayed on the horizontal axis, but single-dish data in *AIPS* has only 1 of this sort of IF.) The data may be displayed with as much or as little time averaging as desired and is very useful for examining your data even if you do not think that editing is needed.

To run **SPFLG**, type

> TASK 'SPFLG' ; INP C_R	to review the inputs.
--	-----------------------

> INDI n ; GETN ctn C _R	to select the disk and catalog entry of the sorted data set.
> FLAGVER 1 C _R	to select the use of a flag table <i>on the input data</i> .
> OUTFGVER 0 C _R	to write a new flag table including all flags applied to the input data.
> BCHAN 0 ; ECHAN 0 C _R	to view all spectral channels.
> DOCALIB FALSE C _R	to inhibit <i>interferometer calibration</i> of your data.
> IN2SEQ 0 ; DOCAT FALSE C _R	to create a new, but temporary “master file” each time.
> ANTEN 0 ; BASEL 0 C _R	to include all “baselines.”
> DPARM 0, 1, 0, 0, 0, 0.1 C _R	to include autocorrelation data and to set the fundamental interval used to average data into the master file. The defaults for these parameters are not suitable for single-dish data. The other DPARM parameters may be ignored since they can be altered during the interactive session.
> INP C _R	to review the inputs.
> GO C _R	to begin the interactive display and editing.

The task will then read your data to determine which times occur in the included portions (you may set **TIMERANG**, restrict autocorrelations, etc.) and then construct a master grid file with spectral channel as the first axis, pseudo-regular times on the second axis (gaps are mostly suppressed), and, if needed, baseline number on the third axis. **SPFLG** tells you the size of the resulting file, *e.g.*, SPFLG1: Basic UV image is 128 14079 pixels in X,Y (Ch,T).

At this point, **SPFLG** selects an initial *display* smoothing time long enough to fit all of the master grid onto your TV window. It then averages the data to this interval and creates a display not unlike that seen in Figure 10.3. Move the TV cursor to any menu item (it will change color to show which has been selected) and press button D for on-line help information or press buttons A, B, or C to select the operation. Normally, you will probably begin by reducing the smoothing time (**ENTER SMOOTH TIME** menu option followed by typing in the new smoothing multiple on your AIPS window). Note that the display does not change other than to add an asterisk after the smoothing time to indicate that that will change on the next image load. This behavior is to allow you to alter a number of choices before doing the potentially expensive TV display. In this typical example, you would either **ENTER BLC** and **ENTER TRC** by hand and finally **LOAD** the sub-image or you can do this interactively with **SET WINDOW + LOAD**. You may examine data values (like **CURVAL**) and flag data with the options in the fourth column. Flagged data are removed from the display. You may review the flags you have prepared, undo any that you dislike, re-apply the remaining ones to make sure the display is correct, and modify the appearance of the display with the options in the first column. The image may be shown in zoom only during editing in order to give you greater accuracy in examining the data values and locations. If you are doing some time smoothing within **SPFLG**, the **DISPLAY RMS** option allows you to view images of the rms rather than the value of the time average. Such a display allows you to find excessively noisy portions of the data quickly.

Finally, when you are done, select **EXIT**. If you have prepared any flagging commands, **SPFLG** will ask you if you wish to enter them into your input data set. Answer yes unless you want to discard them or you have set **DOCAT TRUE** to catalog the master file in order to use it for multiple sessions. If you set **OUTFGVER** to zero, then the flag commands are put into a new flag table which can be deleted later if you wish.

SPFLG is not useful on continuum data; the interactive editor of choice for such data used to be the task **IBLED**, but is now **EDITR**. These tasks are also useful for spectral-line data in that they can display the average (and rms) of a selected range of channels. The spectral averaging should let you see more subtle level problems than can be seen on individual channels (*i.e.*, in **SPFLG**). **EDITR** is a menu-driven, TV display editing task, but it does not use grey-scales to show data values. Instead, it plots time on the horizontal axis and data value on the vertical axis. The full data set for the chosen baseline is displayed initially in a potentially crowded area at the bottom of the TV window. This area is available for editing. If **DOTWO** is true, then it also displays above the edit area a second observable (initially the difference between the

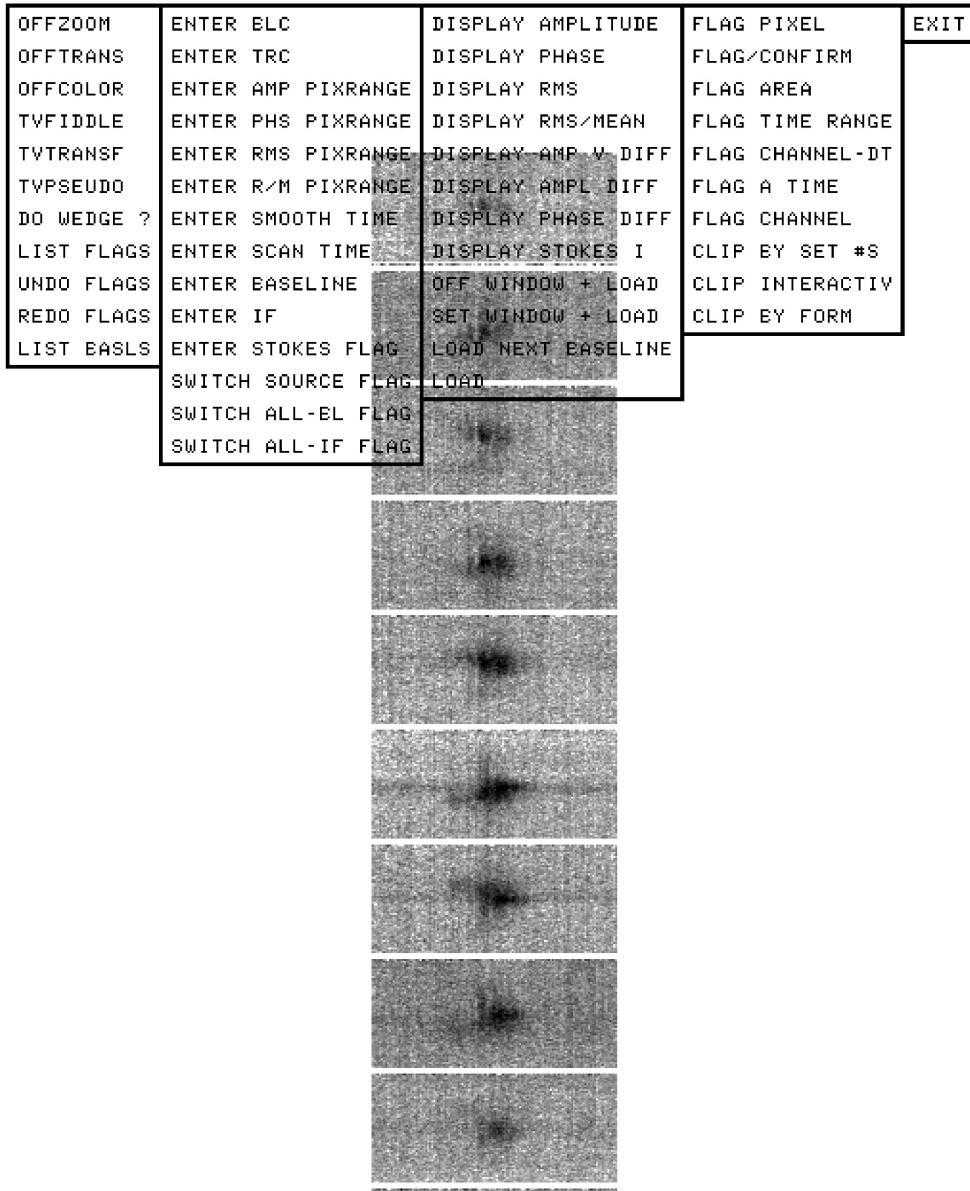


Figure 10.3: A display of a sample TV screen from [SPFLG](#) on single-dish data, made using the *AIPS* task [TVCPS](#) to produce a negative black-and-white display. The [SPFLG](#) menu (in the boxes) and status lines (at the bottom) are displayed in a graphics plane which is normally colored light green. The data are grey scales in a TV memory and may be enhanced in black-and-white or pseudo-colored. The data actually displayed range in intensity from -1.7 to 5.2 Kelvins (as stated during the image loading) and have been averaged to 0.8 seconds. The entire master grid contains 14079 times, but the current window includes only times 5403 through 9538. Flag commands generated at the moment illustrated will flag all source names, all IFs (in the *AIPS* sense), only the displayed baseline, and all Stokes. Note that the menu displayed is now out of date, more options are available in later versions.

amplitude and a running mean of the amplitude) for the primary baseline. **EDITR** allows you to display up to ten other “baselines” (e.g., 12m-antenna IFs) in frames above the active editing frame. These should speed the process of editing and guide you in the choice of flagging one or all baselines at the time of the observation. A smaller time range or window into these full data sets may be selected interactively to enable more detailed editing. Be sure to set **SOLINT** to specify an appropriate averaging interval. Unlike **SPFLG**, no further time averaging is possible. The menu options allow you to work your way through all of your data, selecting time windows and baselines as desired. Consult § 5.5.2 for more details about **EDITR**.

EDITR has the ability to display a second data set for reference in parallel with the one being edited. This option is likely to prove useful for beam-switched continuum observations. Select one of the beam throws for editing and the other for reference display. Then, if editing is required, reverse the roles. It may also be useful to look at your beam-switched data in its differenced form. The task **DIFUV** may be used to difference the plus and minus throws, followed by **EDITR** (or any other *AIPS* uv-data task) to look at the differences. Be sure to tell **DIFUV** that the time difference between the plus and the minus beam throws should not be considered significant, *i.e.*, **SOLINT** = 1 / 8 / 60 or a little bit more to avoid round-off effects.

10.2.3 Using **CSCOR** and **SDCAL** to calibrate your data

The current calibration routines for single-dish data in *AIPS* are fairly rudimentary. The concept is similar to that used for interferometers. Corrections are developed in an extension table (called CS in single-dish, CL in interferometry) which can be applied to the data by some tasks. In particular, the single-dish tasks **PRTSD** and **SDGRD** are able to apply the CS table to the data without modifying the data as stored on disk. They do this using the **DOCAL** and **GAINUSE** adverbs. Other *uv* tasks, designed primarily for interferometry, also use these adverbs, but do not understand or apply CS tables. For such tasks, you should carefully turn off the calibration option. If you do not, such tasks will fail.

There are two tasks which can create CS tables: **SDTUV** discussed above and **INDXR**. To use the latter, enter

- > **TASK 'INDXR'** ; **INP C_R** to review the inputs.
- > **INDI n** ; **GETN ctn C_R** to select the disk and catalog entry of the data set.
- > **CPARM T₁, T₂, ΔT C_R** to set the largest gap (T₁) and longest scan (T₂) times expected in the data set (for the index table) and to set the time interval (ΔT) in the CS table, all in minutes.
- > **GO C_R** to run the task to create an index (**NX**) and a calibration (CS) table attached to the main data set.

Note that this task requires the data to be in time order and expects an antenna (AN) table. You may set **CPARM(5)** to the maximum antenna number (beam number) in your data set and, with a few grumbles, **INDXR** will still create and initialize a CS table when you do not have an antenna table.

At this writing, the CS table may be used to correct the recorded right ascension and declination (*i.e.*, the pointing) and to correct the amplitudes for atmospheric opacity and other gain as a function of zenith angle effects. To add an atmospheric opacity correction to the CS table produced by **INDXR**, type:

- > **TASK 'CSCOR'** ; **INP C_R** to review the inputs.
- > **TIMERAN 0** ; **ANTENN 0 C_R** to do all times and antennas.
- > **GAINVER 1** ; **GAINUSE 2 C_R** to modify the base table, producing a new table.
- > **OPCODE 'OPAC'** C_R to do the opacity correction.
- > **BPARM O_z, 0 C_R** to specify the zenith opacity in nepers.
- > **GO C_R** to run the task.

Note that **CSCOR** only writes those records in the output file that you have selected via **TIMERANG**, **ANTENNAS**, etc. To make a new CS table to work for the full data set, you should first use **TACOP** to write the new table and then set **GAINVER** and **GAINUSE** to both point at the new table. **CSCOR** needs to compute the zenith angle

and therefore needs to have an antennas file. If your data set does not have one, you may give the antenna longitude and latitude in the **CPARM** adverb. The other operations offered by **CSCOR** are **GAIN**, **PTRA**, and **PTDC** which apply as second-order polynomial functions of zenith angle corrections to the gain, right ascension, and declination, respectively. The format of the CS table allows for an additive flux correction as well. There are no tasks at this time to determine such a correction.

The basic single-dish tasks **PRTSD** and **SDGRD** can apply the CS table to the data as they read them in. Other *uv* tasks which are more directed toward interferometry data cannot do this. If you need to use such tasks with corrected data, then you must apply the corrections with **SDCAL** and write a new “calibrated” data set. To do this:

> TASK 'SDCAL' ; INP C_R	to review the inputs.
> INDI n ; GETN ctn C_R	to select the disk and catalog entry of the data set.
> TIMERA 0 ; FLAGVER 1 C_R	to do all times and apply any flagging.
> BCHAN 1 ; ECHAN 0 C_R	to get all channels.
> DOCAL TRUE ; GAINUSE 0 C_R	to apply the highest numbered CS table.
> APARM 0 C_R	to do no averaging of spectral channels.
> GO C_R	to run the task.

The output file from **SDCAL** can then be fed to **UVPLT**, **SPFLG**, or any other *uv*-data task including of course **PRTSD** and **SDGRD**.

10.2.4 Using SDLSF and SDVEL to correct your spectral-line data

It may be convenient to remove a spectral baseline from each sample before the imaging step. Doing so may allow you to skip the removal of a spectral baseline from the image cubes (as described in § 10.4.1). To do this, type:

> TASK 'SDLSF' ; INP C_R	to review the inputs.
> INDI n ; GETN ctn C_R	to select the disk and catalog entry of the data set.
> NCOUNT 1 C_R	to solve for a slope as well as a constant in the baselines.
> DOALL 1 C_R	to fit a single baseline to all samples taken at a particular time. This is useful for single-beam, multi-polarization data, but, for multi-beam data, it is found that instrumental problems dominate weather and require DOALL = -1 C _R instead.
> DOOUT -1 C_R	to avoid writing a continuum data set.
> FLUX 0 ; CUTOFF 0 C_R	to write all data with no flagging.
> CHANSEL s₁, e₁, i₁, s₂, e₂, i₂ ... C_R	to use every <i>i</i> ₁ channel from <i>s</i> ₁ through <i>e</i> ₁ , every <i>i</i> ₂ channel from <i>s</i> ₂ through <i>e</i> ₂ , and so forth to fit the baseline. Be sure to avoid dubious channels, if any, at the ends and any channels with real line signal. It is important to have regions at both ends of the spectrum to fit the slope.
> INP C_R	to review the inputs.
> GO C_R	to run the task.

You may, and probably should, use **FLUX** and **CUTOFF** to flag those data having excessive noise or excessive signals in individual channels. These “excesses” are measured only in the channels selected by **CHANSEL** for fitting the baseline.

If you have observed a wide field with relatively narrow spectral channels, there is an effect which you should consider. The “velocity” corresponding to a particular frequency of observation depends on the velocity definition (e.g., LSR or heliocentric), the direction at which the telescope pointed, the time of year, the time of day, and the location of the telescope. Most telescopes adjust the observing frequency to achieve

the desired velocity for some reference time and position and many adjust the frequency periodically to account for time changes. However, few, if any, can adjust the observing frequency for every pointing direction and time in a rapidly scanned on-the-fly observing mode. The 12m telescope now sets the frequency once per image with respect to the reference coordinate (usually the image center). In this mode, the maximum velocity error in a 2 degree by 2 degree image is about 1.16 km/s (in LSR velocities) and 0.79 km/s (heliocentric). Since mm lines are often narrow, this can be a significant effect. Fortunately, single-dish OTF data may be fully corrected for this effect so long as your spectra are fully sampled in frequency. The task **SDVEL** shifts each spectrum so that the reference channel has the reference velocity for its pointing position. The **DPARM** adverb array is used to tell the task how the telescope set reference velocities and to ask the task to report any excessive shifts and even flag data having really excessive shifts. The latter are to detect and/or remove times in which the telescope pointing was significantly in error (*i.e.*, high winds). **DPARM(1)** should be set to 0 for 12m data taken after 5 May 1997 and to 2 for data taken before that date. The task **VTEST** was written to help you evaluate the magnitude of this effect.

10.2.5 Using SDMOD and BSMOD to model your data

It is sometimes useful to replace your actual data with a source model or, if your continuum levels are well calibrated, to add or subtract a model from your data. The task to do this is called **SDMOD** and allows up to four spatially elliptical Gaussians (or an image) to replace the data, or to be added to the data, with either a Gaussian or no frequency dependence. When the data are replaced, a random noise may also be added. **SDMOD** has options for modeling beam-switched continuum data (set **BPARM(1) = 1**) as well as for spectral-line data. For example, to see what a modestly noisy point source at the origin would look like after all of the imaging steps:

> TASK 'SDMOD'; INP C_R	to review the inputs.
> INDI n ; GETN ctn C_R	to select the disk and catalog entry of the data set.
> BCHAN n ; ECHAN n C_R	to get one channel only.
> NGAUSS 1 ; APARM 0 C_R	to get one Gaussian with no frequency dependence.
> GWIDTH 0 ; GPOS 0 C_R	to do a point source (convolved with the single-dish beamwidth in the header) at the coordinate center.
> GMAX 1, 0 ; FLUX 0.05 C_R	to do a 1 K object with rms noise of 0.05 K.
> GO C_R	to run the task.

The output file from **SDMOD** can then be fed to **SDGRD**, **BSGRD**, or any other appropriate task as if it were regular data. The input model is convolved with the single-dish beamwidth given in the *uv* data header before being used to replace or add to the input data. The history file will show in detail what was done.

Beam-switched observations may be modeled with task **BSMOD**. No input data set is needed. Instead two regular grids of switched data are constructed from a specified model plus noise and a variety of instrumental defects.

10.3 Imaging single-dish data in AIPS

10.3.1 Normal single-dish imaging

The process of imaging in single-dish is a process of convolving the “randomly distributed” observations with some convolving function and then resampling the result on a regular image grid. Tasks **SDGRD** and **SDIMG** combine the data calibration, selection, projection, sorting, and gridding in one task capable of imaging all spectral channels into one output data “cube.” They are relatively easy to run, but selecting

the correct input adverb values is more difficult. Choose **SDGRD** for most single-dish applications; **SDIMG** is very similar but can handle larger output images at the cost of making a sorted copy of the entire input data set (which can be very large). Type:

> TASK 'SDGRD'; INP C _R	to review the inputs.
> INDI <i>n</i> ; GETN <i>ctn</i> C _R	to select the disk and catalog entry of the data set.
> TIMERA 0; FLAGVER 1 C _R	to do all times and apply any flagging.
> BCHAN 1; ECHAN 0 C _R	to get all channels.
> DOCAL FALSE C _R	to apply no calibration.
> OPTYPE '-GLS' C _R	to make the image on a “global sinusoidal” kind of projection.
> APARM 0 C _R	to use the observed right ascension and declination given in the header as the center of the image. For concatenated data sets, use APARM to specify a more appropriate center.
> REWEIGHT 0, 0.05 C _R	to have an “interpolated” or best-estimate image for output, cutting off any cells with convolved weight < 0.05 of the maximum convolved weight.
> CELLSIZE <i>c</i> C _R	to set the image cells to be <i>c</i> arc seconds on a side.
> IMSIZE <i>N_x</i> , <i>N_y</i> C _R	to make the image of each channel be <i>N_x</i> by <i>N_y</i> pixels centered on the coordinate selected by APARM .
> XTYPE 16; XPARM 0,0,0,0,50 C _R	to select convolution function type 16 (a round Bessel function times Gaussian) with default parameters and 50 samples of the function per pixel. The default of 20 samples/cell is probably adequate.
> INP C _R	to review the inputs.
> GO C _R	to run the task.

SDGRD begins by reading the data selecting only those samples which will fit fully on the image grid. It reports how many were read and how many selected. If you have made the image too small, with **IMSIZE** or **CELLSIZE**, then data will be discarded. Use **PRTSD** with a substantial **XINC** to determine the full spatial distribution of your data. It does not hurt to have the output image be a bit bigger than absolutely necessary. If you are uncertain about the parameters to use, try running **SDGRD** on a single channel to begin with since it will be much faster.

A number of these parameters require more discussion. **REWEIGHT**(1) selects the type of output image. The data are multiplied by their weights (which depend on the system temperature), convolved by the sampled convolving function and then summed at each image pixel. **REWEIGHT**(1) = 1 selects the result, which is not calibrated in any way since its scaling depends on the scaling of the data weights and the convolving function and on the distribution of data. While the program “grids” the actual data it also does the same process on the data replaced by 1.0. That result, the convolved weights may be obtained with **REWEIGHT**(1) = 2. The most meaningful image, which is obtained with **REWEIGHT**(1) = 0, is the ratio of the former to the latter. This is the interpolated or best-estimate image and will be similar to the convolved image in well-sampled regions except for having retained the calibration. **REWEIGHT**(1) = 3 tells **SDGRD** to compute an image of the expected noise (actually $1/\sigma^2$) in the output image of type 0; see **WTSUM** below (§ 10.4.2) for its use.

REWEIGHT(2) controls which pixels are retained in the output image and which are blanked by specifying a cutoff as a fraction of the maximum convolved weight. It is important to blank pixels which are either simple extrapolations of single samples or, worse, extrapolations of only a couple noisy samples. In the latter case, it is possible to get very large image values. Thus, if the output is $(W_1 D_1 + W_2 D_2)/(W_1 + W_2)$ where the *D*'s are data and the *W*'s are convolved weights and if, say $W_1 = 0.1001, D_1 = 1.0, W_2 = -0.1000, D_2 = -1.0$, then the output would be 2001. Such large and erroneous values will be obvious, but will confuse software which must deal with the whole image and will also confuse people to whom you may show the image. In simple cases, in which all data have roughly the same data weights (system

temperatures), setting **REWEIGHT(2) = 0.2** or even more is probably wise. However, if some portions of the data have significantly lower weights than others, then you may have to set a lower value in order to keep the low-weight regions from being completely blanked.

The choices of **CELLSIZE** and the widths of the convolving function are related to the spatial resolution inherent in your data, *i.e.*, to the single-dish beamwidth. If the pixels and function are too small, then data samples which are really from the same point in the sky will appear as if different in the output image. If, however, they are too large, then too much data will be smoothed together and spatial resolution will be lost. The latter may be desirable to improve signal-to-noise, but image smoothing can be done at a later stage as well. You may wish to experiment with these parameters, but it is usually good to start with a **CELLSIZE** about one-third of the beamwidth (fwhm) of your telescope. The default parameters (**XPARMs**) of all convolving functions may be used with this cell size. You may vary these parameters in units of cells or in units of arc seconds; enter **HELP UVnTYPE C_R** to look at the parameters for type *n* (*n* = 1 through 6). If you give **XTYPE = n + 10**, then you get a round rather than square function which is perhaps better suited to this type of data. If you wish to change the cell size, but retain the same convolving function in angular measure on the sky, you may give **XTYPE < 0** and specify the **XPARMs** in arc seconds rather than cells.

The choice of convolving function affects the noise levels and actual spatial resolution in the output image. In effect, the Fourier transform of the convolving function acts to modify the illumination pattern of the feed horn onto the aperture. Figure 10.4 shows slices through the Fourier transforms of six of the available convolving functions. The ideal function would be flat all the way across and then suddenly zero at the edges. Type 14 is the widest, but has a deep dip in the middle. This leaves out the center portion of the dish and illuminates the outer portions, effectively improving the spatial resolution of the image over that of the normal telescope, but with a noticeable loss of signal-to-noise ratio. The spheroidal functions, on the other hand, illuminate the center fully and leave out the outer portions. This degrades the spatial resolution, but noticeably improves the noise levels. Types 4 and 16 seem to be the best compromise. Type 16 is preferred since it is zero at the edges. Round functions require more computer memory than square ones, so type 4 would be preferred on computers with small memories.

Images may be built up from observations taken at significantly different times. The simplest way to do this is to concatenate the two “uv” data sets on disk with **OTFUV** or **DBCON** (§ 10.1.1.2) and then use **SDGRD** once to make the image. Some single-dish data sets are so large — or the time interval so great — that this is not practical. **SDGRD** combines observations taking into account the data weights which are based on the measured system temperatures. You can get the same weighted averaging in the image plane if you first compute a “weight” image and then use the task **WTSUM** to do the averaging. To get a weight image:

- > **TGET SDGRD C_R** to get the inputs used for the actual image cube.
- > **REWEIGHT(1) 3 C_R** to get the weight image which is proportional to $1/\sigma^2$ expected from the actual gridding done on the data whose weights are assumed proportional to their $1/\sigma^2$.
- > **BCHAN n ; ECHAN BCHAN C_R** to image a single channel when there is no channel-dependent data weights and flagging.
- > **GO C_R** to get the weight image.

See § 10.4.2 for details about **WTSUM**.

10.3.2 Beam-switched continuum imaging

The construction of images from beam-switched on-the-fly continuum observations is more properly a research question than one of production software. Observers in this mode should be aware that the optimal methods of data reduction are probably not yet known and that the methods currently provided require the user to determine three critical correction parameters. In this mode, the telescope is moved in a raster of offsets in azimuth and elevation with respect to the central coordinate. The beam is switched rapidly from

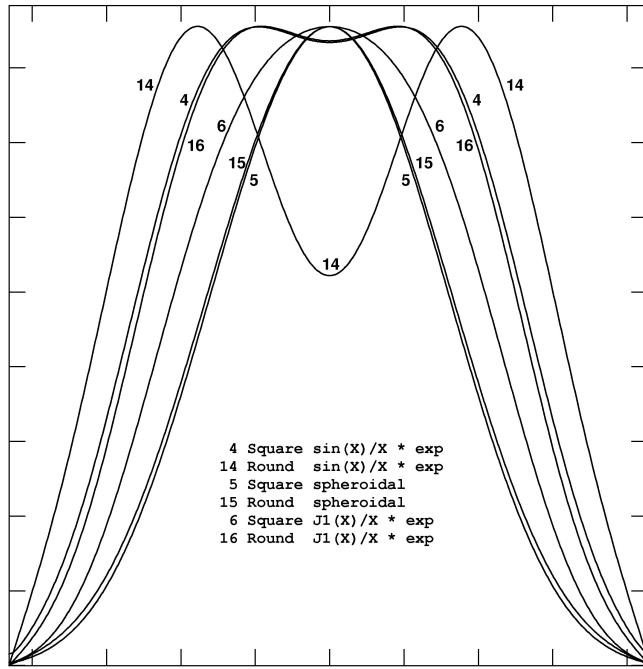


Figure 10.4: Slices through the Fourier transforms of six convolving functions using the `XTYPE` numbers shown on the plot with default values of the `XPARMS`.

a “plus” position to a “minus” position at constant elevation. On the 12m, there are four plus samples and four minus samples taken each second, all taken while the telescope is being driven rapidly in azimuth at a constant (relative to the central source) elevation. In principle, each pair of plus and minus points contain the same instrumental bias but different celestial signals. It is then the job of the software to disentangle the time variable bias from the two beams’ estimates of the sky brightness.

A technique for doing the disentangling was first described by Emerson, Klein, and Haslam (Astronomy and Astrophysics, 76, 92–105, 1979). The plus and minus samples are differenced removing the instrumental bias and creating two images of the sky, one positive and one negative. Problems arise because the two images potentially can overlap and because, in the OTF mode of observing, the telescope positioning is not exactly along rows of the output image and the relative positioning of the plus and minus beams varies both due to the wobbles in the telescope pointing and due to the reversing of the direction of telescope movement. The Emerson *et al.* technique involves a convolution of each row in the differenced image with a function which is a set of positive and negative delta functions (or $\frac{\sin x}{x}$ functions when the total beam throw is not an integer number of image cells). It turns out that the problem of image overlap is largely solved by this technique. Unfortunately, differences in the position of the plus and minus beam with respect to the source and to the image cells appear to limit the quality of the images produced with this technique.

The principal task used to produce images from data is called `BSGRD`. It makes two images from the two “uv” data sets written by `OTFBS`, gridding each sample at the coordinate at which it was observed (neglecting the throw but not the telescope movement between plus and minus). If the beam throw was not exactly along constant elevation, it then shifts the two images. Then it applies the Emerson *et al.* technique, fitting and removing baselines, differencing the two images, and convolving the difference image with an appropriate $\frac{\sin x}{x}$ function. Finally, `BSGRD` regrids the data from relative azimuth-elevation coordinates onto a grid in normal celestial coordinates. This task is a combination of four tasks, `SDGRD` described above to make the images, `OGEOM` to do the rotation correction, `BSCOR` to apply the Emerson *et al.* technique, and `BSGEO` to regrid the data onto normal celestial coordinates.

To use **BSGRD**, type

- > **TASK 'BSGRD'** ; **INP** C_R
- > **INDI** n ; **GETN** ctn C_R

- > **TIMERA** 0 ; **FLAGVER** 1 C_R
- > **DOCAL** FALSE C_R
- > **OPTYPE** '-GLS' C_R
- > **APARM** 0 C_R

- > **REWEIGHT** 0, 0.05 C_R

- > **CELLSIZE** c C_R
- > **IMSIZE** N_x, N_y C_R

- > **XTYPE** 16 ; **XPARM** 0,0,0,0,50 C_R

- > **FACTOR** f C_R
- > **ROTATE** ρ C_R
- > **DPARM** 1,1,x₁,x₂,x₃,x₄ C_R

- > **ORDER** 1 C_R
- > **DOCAT** -1 C_R
- > **INP** C_R
- > **GO** C_R

to review the inputs.

to select the disk and catalog entry of the data set. Note that the class name is assumed to have a plus sign in the sixth character for the plus throw data set and a minus sign in that character for the minus throw data set.

to do all times and apply any flagging.

to apply no calibration.

to make the image on a “global sinusoidal” kind of projection.

to use the observed right ascension and declination given in the header as the center of the image. For concatenated data sets, use **APARM** to specify a more appropriate center.

to have an “interpolated” or best-estimate image for output, cutting off any cells with convolved weight < 0.05 of the maximum convolved weight.

to set the image cells to be c arc seconds on a side.

to make the image of each throw be N_x by N_y pixels centered on the coordinate selected by **APARM**.

to select convolution function type 16 (a round Bessel function times Gaussian) with default parameters and 50 samples of the function per pixel. The same function is used in both convolutions.

to multiply the recorded throw lengths by f in doing the Emerson *et al.* correction.

to correct the throws for being ρ degrees off from horizontal.

to specify that the two beams have the same relative amplitude and to give the pixel numbers to be used to fit baselines in *both* images.

to fit a slope as well as a constant in the horizontal baseline in each row.

to delete the intermediate images created by **BSGRD**.

to review the inputs.

to run the task.

BSGRD takes three correction parameters which you must supply: the throw length error **FACTOR**, the throw angle error **ROTATE**, and the relative beam gain error **DPARM**(1). To estimate these, you will need data on a relatively strong point source. Use **SDGRD** to make an image of each throw of these data, setting **ROTATE** = 0 since rotation *must* be done later and setting **ECHAN** = 1 to eliminate the coordinate information which is confusing to **MCUBE** and used only by **BSGEO**. The tasks **IMFIT** and/or **JMFIT** (§ 7.5.2) may be useful in fitting the location and peak of the two beams. Since there is likely to be a significant offset from zero in these images, be sure to fit for the offset using a second component of **CTYPE** = 4. For reasons that are not clear, these tasks may not provide sufficiently accurate positions. Another approach then is to take the two images produced by **SDGRD** and then run **OGEOM** and **BSCOR** for a range of rotations and factors. Find the image that is most pleasing and put its parameters into **BSGRD** for the program source. Of course, it is not clear that these correction factors are constant with time or pointing, so this could all be bologna.

For example,

- > **TASK 'OGEOM'** ; **INP** C_R
- > **INDI** n ; **GETN** ctn₊ C_R
- > **APARM** 0 C_R

to review the inputs.

to select the disk and catalog entry of the plus image.

to do no shifts or rescaling.

- > **DOWAIT** 1 C_R to wait for a task to finish before resuming AIPS.
- > **OUTCLA** 'OGEOM+' C_R to set the output class to show the throw sign.
- > **FOR APARM(3) = -2 , 2.01 BY 0.1 ; GO; END** C_R to produce 41 plus images each with a slightly different rotation.
- > **GETN** *ctn_-* ; **OUTCLA** 'OGEOM-' C_R to select the minus image as input and specify the output class.
- > **FOR APARM(3) = -2 , 2.01 BY 0.1 ; GO; END** C_R to produce 41 minus images each with a slightly different rotation.

Then apply the Emerson *et al.* corrections to each of the 41 with

- > **TASK** 'BSCOR' ; **INP** C_R to review the inputs.
- > **INDI** *n* ; **GETN** *ctn_+* C_R to select the disk and catalog entry of one of the rotated plus images.
- > **IN2DI** *n* ; **GET2N** *ctn_-* C_R to select the disk and catalog entry of one of the rotated minus images.
- > **FACTOR** *f* C_R to multiply the recorded throw lengths by *f* in doing the Emerson *et al.* correction. Use 1.0 as an initial guess.
- > **DPARM** 1,1,*x₁,x₂,x₃,x₄* C_R to specify that the two beams have the same relative amplitude and to give the pixel numbers to be used to fit baselines in *both* images. The choice of the *x_n* is significant.
- > **ORDER** 1 C_R to fit a slope as well as a constant in the horizontal baseline in each row.
- > **FOR INSEQ = 1 : 41; IN2SEQ = INSEQ ; GO ; END** C_R to produce 41 “corrected” images.

It is convenient to look at the images with tools such as **TVMOVIE** (§ 8.5.4) and **KNTR** (§ 10.4.5). To build the “cube”, use **MCUBE** as:

- > **TASK** 'MCUBE' ; **INP** C_R to review the inputs.
- > **INDI** *n* ; **GETN** *ctn* C_R to select the disk and catalog entry of the first of the corrected images.
- > **IN2SEQ** 41 ; **IN3SEQ** 1 C_R to set the sequence number loop limit and increment.
- > **AXREF** 0 ; **AX2REF** 41 ; **NPOINTS** 41 C_R to set the locations of the images in the cube explicitly.
- > **DOALIGN** -2 C_R to have **MCUBE** ignore the differing image rotations.
- > **DOWAIT** -1 ; **GO** C_R to resume normal task functioning and to build the data cube.

Examine the cube to find the “best” plane and use the rotation of that plane to run similar tests varying the throw length correction factor. One of these cubes, testing rotation, is illustrated in Figure 10.5.

The determination of throw length is similar:

- > **TASK** 'BSCOR' ; **INP** C_R to review the inputs.
- > **INDI** *n* ; **GETN** *ctn_+* C_R to select the disk and catalog entry of the plus image at the best rotation.
- > **IN2DI** *n* ; **GET2N** *ctn_-* C_R to select the disk and catalog entry of one of the corresponding rotated minus image.
- > **OUTNA** 'ROTATE TEST' C_R to assign a new output name.
- > **DPARM** 1,1,*x₁,x₂,x₃,x₄* C_R to specify that the two beams have the same relative amplitude and to give the pixel numbers to be used to fit baselines in *both* images. The choice of the *x_n* is significant.
- > **ORDER** 1 C_R to fit a slope as well as a constant in the horizontal baseline in each row.
- > **DOWAIT** 1 C_R to run the task in wait mode.
- > **FOR FACTOR = 0.9 ; 1.101 BY 0.005; GO ; END** C_R to produce 41 “corrected” images.

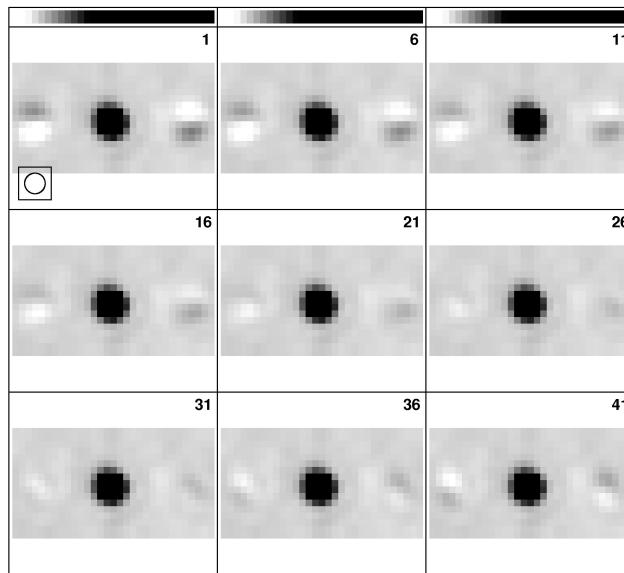


Figure 10.5: Images at selected rotations from 2.0 to -2.0 by -0.5 degrees. Rotations between -0.5 and -1.0 appear to minimize the artifacts due to the incomplete cancellation of the plus and minus beams.

It is convenient to look at the images with tools such as **TVMOVIE** (§ 8.5.4) and **KNTR** (§ 10.4.5). To build the “cube”, use **MCUBE** as:

- > **TASK 'MCUBE'; INP C_R** to review the inputs.
- > **INDI n ; GETN ctn C_R** to select the disk and catalog entry of the first of the new corrected images.
- > **IN2SEQ 41 ; IN3SEQ 1 C_R** to set the sequence number loop limit and increment.
- > **AXREF 0 ; AX2REF 41 ; NPOINTS 41 C_R** to set the locations of the images in the cube explicitly.
- > **DOWAIT -1 ; GO C_R** to resume normal task functioning and to build the data cube.

Examine the cube to find the “best” plane and use the scaling factor of that plane in later imaging. One of these cubes, testing throw length, is illustrated in Figure 10.6.

BSGRD is in fact a deconvolution algorithm to remove the plus-minus beam from the difference image. An experimental Clean algorithm has been made available in **BSCLN**. Although initial tests seemed promising, it appears to converge to the EKH solution after very many iterations and to have systematic problems before that point. The one-dimensional display task **BSTST** will allow you to evaluate and compare the two algorithms on model (one-dimensional) data.

10.4 Analysis and display of single-dish data

The analysis and display of images produced from single-dish data are not, in general, different from those produced by interferometers. See Chapter 6 for a discussion of display tools, Chapter 7 for a variety of analysis tasks, and § 8.5 and § 8.6 for spectral-line analysis and display. Some matters of particular interest to single-dish users will be discussed below.

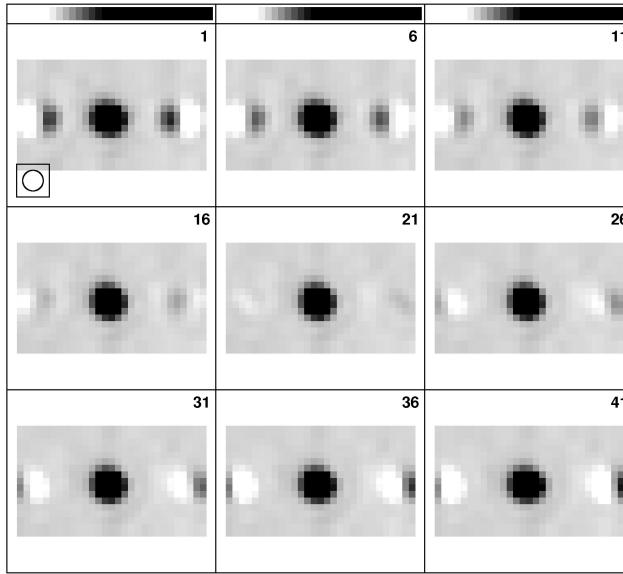


Figure 10.6: Images at selected beam throw corrections from 0.9 to 1.1 by -0.5. Note that rotation errors cause vertical separations of the plus and minus images while throw length errors cause horizontal separations (and hence incomplete cancellation) of the beams.

10.4.1 Spectral baseline removal

As the **SPFLG** display in § 10.2.2 shows, one of the first things most users will want to do is remove a spectral baseline at each pixel in the their image. This is frequently done with **SDLSF** (§ 10.2.4). To do this in the image plane, you must first transpose the data cube to make the frequency axis be first:

- > **TASK 'TRANS'** ; **INP C_R** to review the task's parameters.
 - > **INDI n₁ ; GETN ctn₁ C_R** to select the input image from disk n_1 catalog slot ctn_1 .
 - > **TRANSCOD '312' ; OUTCL 'VXY' C_R** to move the frequency axis from 3rd to 1st.
 - > **GO C_R** to transpose the cube.
- You must also determine, using this input image if needed, which spectral channels are completely free of real emission or absorption. **TMOMOVIE** is often useful; see § 8.5.4. Then:
- > **TASK 'IMLIN'** ; **INP C_R** to review the task's parameters.
 - > **INDI n₂ ; GETN ctn₂ C_R** to select the input image (output from **TRANS**) from disk n_2 catalog slot ctn_2 .
 - > **ORDER 1 C_R** to subtract linear baselines; up to 4th are allowed.
 - > **NBOXES n C_R** to select n contiguous regions along the spectral axis to be used in fitting the channels.
 - > **BOX c₁₁, c₁₂, c₂₁, c₂₂, ... c_{n1}, c_{n2} C_R** to use spectral channels $c_{11} - c_{12}$, $c_{21} - c_{22}$, up to $c_{n1} - c_{n2}$ to fit the baselines at each pixel.
 - > **INP C_R** to review the inputs.
 - > **GO C_R** to fit the baselines, writing a new data cube.

It is sometimes useful to specify **DOOUT TRUE** to obtain images of the fit parameters and of their uncertainties. The uncertainty in the DC offset is a good measure of the uncertainty in the image.

The output from **IMLIN** is the baseline-corrected image in the familiar position-velocity form, with a third axis giving multiple positions on the second celestial coordinate. To go back to sky images as a function of frequency:

```
> TASK 'TRANS' ; INP CR to review the task's parameters.  

> INDI n3 ; GETN ctn3 CR to select the input image (output from IMLIN) from disk n3  

   catalog slot ctn3.  

> TRANSCOD '231' ; OUTCL 'XYV' CR to move the frequency axis from 1st to 3rd.  

> GO CR to transpose the cube back again.
```

10.4.2 Using WTSUM and BSAVG to combine images

To do a weighted average of multiple images of the same field, be sure to make all images with the same geometry type, the same cell size, and the same center coordinate. If you have two images,

```
> TASK 'WTSUM' ; INP CR to review the inputs.  

> INDI n1 ; GETN ctn1 CR to select the first input image from disk n1 catalog slot ctn1.  

> INDI n2 ; GET2N ctn2 CR to select the second input image from disk n1 catalog slot ctn2.  

> INDI n3 ; GET3N ctn3 CR to select the first weight image from disk n3 catalog slot ctn3.  

> INDI n4 ; GET4N ctn4 CR to select the second weight image from disk n4 catalog slot ctn4.  

> DOINVER FALSE CR to state that the weight images are weights rather than rms's.  

> GO CR to compute an averaged image cube and a new weight image.
```

The weight images can be either a single plane or a cube that matches the corresponding image cube. All must be on the same spectral and celestial coordinate system.

If you have more than two images of the same field, then all images must have the same name parameters, differing only by having consecutive sequence numbers. All weight images must have the same name parameters with corresponding consecutive sequence numbers. The verb RENAME may be used to correct problems in naming. Then

```
> TASK 'WTSUM' ; INP CR to review the inputs.  

> INDI n1 ; GETN ctn1 CR to select the first input image from disk n1 catalog slot ctn1.  

> CLR2NAME ; IN2SEQ m2 CR to select the looping mode and set the highest image sequence  

   number.  

> INDI n3 ; GET3N ctn3 CR to select the first weight image from disk n3 catalog slot ctn3.  

> CLR4NAME CR to clear the unused fourth name set.  

> DOINVER FALSE CR to state that the weight images are weights rather than rms's.  

> GO CR to compute an averaged image cube and a new weight image.
```

If m₁ is the sequence number of the first image (in ctn₁) and w₁ is the sequence number of the first weight image (in ctn₃), then images of sequence numbers m₁ through m₂ will be weighted with corresponding weight images of sequence number w₁ through w₁ + m₂ - m₁. All weight images must be a single plane or all weight images must be a full cube matching the images.

BSAVG is a special task written to average beam-switched continuum images. Each image is Fourier transformed and weighted to give no weight to Fourier components at the beam switching spatial frequency and direction (since the images lack any non-noise information at these lines in the Fourier domain). Images made at different parallactic angles (*i.e.*, different hour angles) have these zero-weight lines at different angles while images made with different throw lengths have these zero-weight lines at different spatial frequencies. Thus, averaging images in this way (and Fourier transforming them back) should produce images with less noise and more information content. This algorithm works only on images that are made very quickly. If there is a significant rotation of the parallactic angle during the observation of one image, then the zero-weight "line" is actually curved and smeared away from the center (in Fourier space). The failure of this algorithm when observations are made with constant-elevation throws is one reason why some telescopes are designed to beam-switch in celestial coordinates.

10.4.3 Spectral moment analysis

A data cube may be reduced to a line-sum and a predominant-velocity image when the spectral shape is fairly simple at all points of the image. The simplest task to do this is:

- > **TASK 'XMOM'** ; **INP C_R** to review the inputs.
- > **INDI n** ; **GETN ctn C_R** to select the input image from disk *n* catalog slot *ctn* — use the output from **IMLIN** with velocity as the first axis.
- > **FLUX x C_R** to include only pixels $> x$ in brightness when computing the moments.
- > **GO C_R** to compute images of the *o*th through 3rd moments plus an image of the number of pixels used at each position.

This simple prescription will produce a result which should tell you whether this mode of analysis is interesting. If it is, then the regions of signal should be separated from regions of no signal so that the latter do not contribute to the noise in the moment images. See the discussions in § 7.4 and § 8.6 for methods of doing this. After the non-signal regions are blanked, the moments should be recomputed.

10.4.4 Source modeling and fitting

Gaussian fitting of images is discussed in some detail in § 7.5 while source modeling may be done in the “uv” data domain with **SDMOD** (§ 10.2.5) and in the image domain with **IMMOD**. The task **SAD** will find, and fit Gaussians to, sources in your image. Although it works on a plane of the image at a time, it records the plane number in its output model-fit (MF) table. This will allow you to examine the fits to your sources as a function of frequency. To run **SAD** on a number of image planes:

- > **TASK 'SAD'** ; **INP C_R** to review the inputs.
- > **INDI n** ; **GETN ctn C_R** to select the image cube from disk *n* catalog slot *ctn*
- > **BLC 0** ; **TRC 0 C_R** to search for sources over the full plane.
- > **DORESID FALSE C_R** to delete the residual image after fitting; the fit results are kept in an MF file attached to the input image.
- > **NGAUSS 10 C_R** to allow up to 10 possible sources to be fit; make this enough to allow for a noise spike or two.
- > **CUTOFF x C_R** to fit “islands” of flux $> x$ only — this is probably the most important parameter.
- > **DOCRT 132 C_R** to display results on your workstation rather than the line printer.
- > **DOALL 1** ; **DOWIDTH 1 C_R** to allow the task to fit multiple sources to an island and to fit the source widths.
- > **OUTVERS -1 C_R** to suppress writing of CC files.
- > **INVERS 1 C_R** to use one MF file for all fits.
- > **DOWAIT TRUE C_R** to resume AIPS only when the task finishes; this allows looping without tripping over ourselves.
- > **INP C_R** to recheck the inputs.
- > **FOR BLC(3) = c₁ TO c₂ ; GO; END C_R** to fit channels *c₁* through *c₂*.

SAD will reject dubious solutions for a variety of reasons. The **DPARM** adverb allows you to control these reasons and **PRTLEV** controls how much of an explanation you get.

SAD offers a printer option to provide a detailed account of each execution. To view a simpler summary of the current contents of one or more MF files, use

> TASK 'MFPRT' ; INP C_R	to review the inputs.
> INDI <i>n</i> ; GETN <i>ctn</i> C_R	to select the image cube from disk <i>n</i> catalog slot <i>ctn</i> as input to SAD .
> INVER <i>n</i>₁ ; IN2VER <i>n</i>₂ ; XINC 1 C_R	to view MF file versions <i>n</i> ₁ through <i>n</i> ₂ .
> DOCRT 132 C_R	to see the display on your monitor.
> FLUX 0 ; IMSIZE 0 C_R	to see all components.
> SORT 'C' C_R	to see the file in channel number order.
> GO C_R	to run the task.

Setting **DOCRT FALSE** and specifying **OUTPRINT** will produce a file suitable for some non-AIPS modeling programs.

10.4.5 Image displays

The subject of displays in AIPS has been treated extensively in earlier chapters. To make a printer representation of your image, see § 6.2.2 for a discussion of **PRTIM**. See § 6.3.2 for a discussion of plotter displays of images including tasks **CNTR**, **PCNTR**, **GREYS**, **PLROW**, **PROFL**, **IMVIM**, and **IMEAN**. Spectral-line displays are described in some detail in § 8.5.4 including tasks **KNTR** and **PLCUB** and the TV-movie display verbs **TMOVIE** and **TVCUBE**. The use of the TV for display, image enhancement, parameter setting, data examination, image comparison, and the like is described in detail in §§ 6.4.

For tutorial purposes, we will include one example here. The contouring task of choice is now **KNTR** since it can display images in grey-scales and/or contours with one or more planes per display and with an optional beam display. It also can plot polarization and has several “coloring” options. For example, to display several spectral channels as contours with the 0th-moment (total CO) image as a grey scale on each display, enter

> TASK 'KNTR' ; INP C_R	to review the inputs.
> INDI <i>n</i>₁ ; GETN <i>ctn</i>₁ C_R	to select the image cube from disk <i>n</i> ₁ catalog slot <i>ctn</i> ₁ .
> IN2DI <i>n</i>₂ ; GET2N <i>ctn</i>₂ C_R	to select the 0 th -moment image plane from disk <i>n</i> ₂ catalog slot <i>ctn</i> ₂ .
> DOCONT 1 ; DOGREY 2 ; DOVECT -1 C_R	to have contours drawn of the first image, grey-scale of the second image, and no polarization.
> BLC 0, 0, <i>c</i>₁ ; TRC 0, 0, <i>c</i>₂ C_R	to draw the full plane from channels <i>c</i> ₁ through <i>c</i> ₂ .
> ZINC Δc C_R	to display every Δc th channel.
> PIXRANGE <i>B</i>₁, <i>B</i>₂ C_R	to do grey scales from <i>B</i> ₁ through <i>B</i> ₂ only, clipping the most negative and positive values if desired. The default is the full range of image DOGREY .
> FUNCTYPE 'SQ' C_R	to use a square-root transfer function on the grey scales to emphasize the lower levels.
> OFMFILE '' C_R	to do no pseudo-coloring in KNTR .
> DOWEDGE 1 C_R	to plot a step wedge along the top.
> CLEV 0.1 C_R	to plot 0.1 K as the basic contour level.
> LEVS 2.7, 7.4, 20.1, 54.6, 148.4, 403.4 C_R	to do logarithmic contours, starting at 0.27 K.
> CBPLOT 18 C_R	to plot a half-power beam contour in the upper right corner and fill it in.
> LABEL 1 C_R	to label each pane with its coordinate (velocity usually).
> DOTV -1 ; INP	to make a plot file and to review the inputs.
> GO C_R	to run the task.

The contour lines will be drawn in a contrasting color when the background grey-scale intensity is high. When [KNTR](#) has finished:

- > [PLVER 0 C_R](#) to plot the most recent plot file for the image.
- > [OUTFILE ' ' C_R](#) to print the plot immediately rather than saving it in a file.
- > [GO LWPLA C_R](#) to translate the plot file into PostScript on a suitable printer.

[LWPLA](#) offers additional control over fonts, paper size, line width, the grey-scale plotting (if [PIXRANGE](#) was not quite right), image pseudo-coloring, coloring of lines and backgrounds, and number of copies. It can make an “encapsulated” PostScript file for inclusion in other documents, such as this *CookBook*. See [HELP POSTSCRIPT](#) for information on other things that can be done with PostScript plot files.

10.4.6 Backing up your data

The next chapter describes how to help the *AIPS* programming team (with “[GRIPES](#)”), to exit *AIPS* (with [EXIT](#)), to delete your data (with [ZAP](#) and [ALLDEST](#)), and, most importantly, to back up your data. Do not assume that data on disk is permanent. Disks can fail and users can make mistakes, so it is wise to make backups to some demountable medium.

10.5 Combining single-dish and interferometer data

We add this section to this chapter with some trepidation since the combination of single-dish data into interferometric imaging is still an area more suited to research than to production. In principle, the problem is fairly simple. You begin by observing a region of sky with a single-dish telescope rather larger than the individual telescopes of the interferometer. From these observations, you make an image which you correct if necessary (*e.g.*, by removing spectral baselines). Then you deconvolve the image removing the convolution of the sky with the beam of the large single-dish telescope. The “sky” observed with the interferometer is the product of the real sky (estimated by your deconvolved image) and the beam of the individual telescopes of the interferometer. Therefore, you multiply your deconvolved image with an image of the single-dish beam and Fourier transform the result. Adjusting the flux scales (usually of the single-dish data), you append or “feather in” the “visibilities” produced by the Fourier transform.

This is a lot of steps and contains several dangers, namely pointing, image alignment, the deconvolution, and the flux re-calibration. *AIPS* can provide you with some help. The imaging and image correction software is described earlier in this chapter. The deconvolution is tricky. Try [DBCON](#) first. It attempts an iterative solution of the deconvolution problem in the image plane. If that is not acceptable, try [CONVL](#) with [OPCODE 'DCON'](#) (in 15JAN96 and later releases). This is a brute force deconvolution that will be very noisy at high spatial frequencies, but these frequencies will be tapered or truncated away later. A third approach is to use [PATGN](#) ([OPCODE 'GAUS'](#)) to make an image of the single-dish beam of the large telescope. [APCLN](#) (§ 5.3.7) can then be persuaded to do a Clark image-based Clean; use a small restoring beam. Remember that this image will be tapered in the *uv* plane. It does not have to be beautiful in detail in the image plane.

The next step is to make an image of the interferometer single-dish beam on the same cell size and center as your deconvolved image. Use [PATGN](#) with [OPCODE 'BEAM'](#) for this. Then multiply the result by the deconvolved image with [COMB](#) using [OPCODE 'MULT'](#) (§ 7.1). If this produces an image with any blanked pixels, run [REMAG](#) to convert the blanks to zeros. Then start trying [IM2UV](#) to produce a *uv* data set. Use [UVTAPER](#) to weight down longer spacings, [FLUX](#) to scale the visibilities, and [UVRANGE](#) to omit the outer spacings. (The first two options appear only in 15JAN96 and later releases.) You should use [PRTUV](#), [UVPLT](#), and even [UVFLG](#) on the output of [IM2UV](#) to make sure that the visibility phases and amplitudes of your single-dish and interferometer data are in reasonable agreement. Finally, combine the two data sets with [DBCON](#) and have fun with [IMAGR](#) (Chapter 5).

11 Exiting from, and Solving Problems in, *AIPS*

This chapter contains a grab-bag of miscellaneous advice on exiting from *AIPS* and on solving a variety of common problems that may arise. The latter are also addressed in § [Z.1.5](#).

11.1 Helping the *AIPS* programmers

Comments, suggestions and bug reports about any facet of *AIPS* are very useful to the *AIPS* programming and management group. Note that “gripes” are only useful when they are informative — *e.g.*, giving details of the circumstances under which a task failed with accompanying system error messages (if any). Terse gripes along the lines of “[UVCOP](#) doesn’t work!” whilst perhaps true in some circumstances, are unlikely to arouse the *AIPS* programmers’ enthusiasm. In many cases, it may be necessary for the programmer to use your data to fix the bug. A FITS-disk file read over the net is a common means to this end. The *AIPS* group may often seem unresponsive to your gripe. This is an unavoidable consequence of the breadth of the *AIPS* project combined with the small size of the group. Nonetheless, if you do not tell the programmers that there is a problem or a good idea, then you are almost certain to encounter the same problem years later and never to see your good idea put into practice. The *AIPS* group depends on help from users.

Gripes can be entered into a site-wide “GR” file and automatically mailed to `daip@nrao.edu` by typing:

> [GRIPE](#) c_R

while in *AIPS*. Follow the directions to record your comment. Current gripes in the file may be read via

> [GRINDEX](#) c_R to display an index of all gripes in the file.

> [JOBNUM](#) n ; [GRLIST](#) c_R to list the n^{th} gripe in the file.

and a gripe may be deleted with

> [JOBNUM](#) n ; [GRDROP](#) c_R to delete the n^{th} gripe in the file and notify `daip`.

Note that the deleted gripe has already been e-mailed to the *AIPS* group in Socorro, so dropped ones get sent too. Do not do a [GRDROP](#) unless you realize that the gripe was erroneous. (An explanatory “gripe” would be appreciated.) If you change your mind about a gripe before you finish it, type `_forget` or `_FORGET` (case sensitive!) to stop the gripe before it is mailed and entered in the file. The addition of automatic e-mail gives immediacy to all gripes and provides, for the first time, real access to the gripe system for sites outside of the NRAO.

11.2 Exiting from *AIPS*

Before ending a period of data reduction with *AIPS*, you should back up those data files which you wish to keep, delete all of your disk data files, tidy up your work area, and then issue the [KLEENEX](#) (to stop the TV, tek, and message servers too) or [EXIT](#) (to leave the servers running) command to the *AIPS* program. Of course, if the computer and disks are part of your very own workstation in your office, you may ignore all this advice. The tape back-ups are a very good idea in any case. Disk files are easily deleted due to software or user malfunction, or lost due to disk hardware malfunction.

11.2.1 Backups

While processing and particularly just before exiting from *AIPS*, please delete as many of your own data sets as possible. Images and *uv* data may be backed up on tape in FITS format using the task **FITTP**. This task can write more than one *AIPS* file on tape in a single execution. For example, to backup all sorted *uv* files (class **UVSRT**), type

> TASK 'FITTP'; INP C_R	to review the inputs.
> DOALL TRUE C_R	to specify that all files with the allowed name parameters are to be written.
> CLRNNAME C_R	to allow any name, class sequence number and disk.
> INTYP 'UV' C_R	to restrict to <i>uv</i> files.
> INCLASS 'UVSRT' C_R	to restrict to class UVSRT files.
> INP C_R	to check the inputs.
> GO C_R	to write the tape.

Then, for example, to write all 3C123 files on disk 2 after the sorted *uv* data files, type:

> INTYP '' ; INCLASS '' C_R	to allow any class and type.
> INNA '3C123' ; INDISK 2 C_R	to restrict things to 3C123 files on disk 2.
> WAIT ; GO C_R	to have AIPS wait for the FITTP execution started above to finish and then to run FITTP with the new inputs.

Note that this sequence will write two copies of any 3C123 **UVSRT** files to be found on disk 2.

Task **FITAB** also writes FITS tapes. For *uv* data it has the advantage of being able to write the data in compressed form, saving disk or tape, and of writing the data in multiple “pieces” for increased reliability. Unfortunately, the table form of data used may not be read by older *AIPS* versions and is not understood by other software systems. **FITAB** was revised in October 2007 and subsequent output can only be read by versions of **UVLOD** and **FITLD** revised in a corresponding manner. Note that **FITAB** is used for processed *uv* data by the NRAO archive — it will matter to many users to update to a modern release. **FITAB** may apply a quantization to images on output that allow the FITS files to be compressed very much more efficiently. If the quantization level is set below 1/4 of the image noise, then the noise in the output image will only be 1–2% larger than in the input image.

11.2.2 Deleting your data

Please delete redundant images and data as soon as possible to preserve disk space for yourself and other users. It is tempting to work on many sets of data at the same time, but this generally takes a lot of disk space and users should limit the amount of data resident on disk to that which will be processed during the session. A data set and all extension files can be deleted by:

> IND n ; GETN ctn C_R	where <i>n</i> and <i>ctn</i> select the disk and catalog numbers of the data set to be deleted.
> ZAP C_R	to do the deletion.

To delete data in contiguous slots from *n* to *m* in a catalog, set the **INDISK** and use the loop:

> **FOR I = n TO m ; GETN I ; ZAP ; END C_R**

For massive deletions — the kind we hope you will use when you leave an NRAO site — use:

> **ALLDEST C_R** to destroy all data files consistent with the inputs to **ALLDEST**.

And to compress your message file, after using **PRTMSG** to print any you want to keep, use:

> **PRNUM -1 ; PRTASK '' ; PRTIME 0 C_R** to do all messages.

> **CLRMSG C_R** to do the clear and compress.

DO NOT DELETE OTHER USERS' DATA OR MESSAGES WITHOUT THE EXPLICIT PERMISSION EITHER OF THE OTHER USER OR OF THE AIPS MANAGER.

11.2.3 Exiting

To exit from AIPS type:

- > **EXIT** C_R to leave TV, message, and graphics servers running, or
- > **KLEENEX** C_R to kill server processes as well as AIPS.

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

11.3 Solving problems in using AIPS

On all computer systems things go wrong due to user error, program error, or hardware failure. Unfortunately, AIPS is not immune to this. The section below reviews several general problem areas and their generalized solutions. Refer to § Z.1.5 for the details appropriate to NRAO's computer systems. Some well-known possibilities follow.

11.3.1 “Terminal” problems

If your workstation window is alive, but AIPS has “disappeared” you may have “suspended” it by typing CTRL Z. The AIPS can be left in a suspended state, placed into the “background” with bg, or returned to the “foreground” again with fg after which it will resume accepting terminal input. If your AIPS appears to be “suspended”, try typing jobs to see which jobs are attached to your window and then use fg %n to bring back job n where n is the job number of the suspended AIPS. If no AIPS job is suspended from the current window, check all other windows you have running on the workstation for the missing simian before starting a new AIPS. Otherwise, you may run out of allowed AIPSe and/or encounter mysterious file locking problems.

If your workstation window (or terminal on obsolete systems) is “dead”, *i.e.*, refuses to show signs of talking to your computer, you have a problem. There are numerous possible causes. If typed characters are shown on the screen, but not executed, then

1. Are you executing a long verb, *e.g.*, **REWIND**, **AVFILE**, **RESCALE**? If so, be patient.
2. Are you executing some interactive TV or TEK verb which is waiting for input from the cursor or buttons? If so, provide the input.
3. Have you started a task with **DOWAIT** set to **TRUE** (+1.0)? If so, wait for the task to finish. Most tasks report their progress on the message monitor window (or your input window).
4. Is AIPS waiting while a tape rewinds or skips files or is it waiting to open some disk file currently being used by one of your tasks? Be patient.

If typed characters do not appear, then

1. Have you stopped output to your window accidentally by hitting the appropriate NO SCRL or other XOFF control sequence? If so, hit the XON control sequence. (These are CTRL S and CTRL Q, respectively.)

2. Do other windows connected to the computer appear to be “alive”? If so, use one of them and inquire about the status of your AIPS program and tasks; on Linux and Berkeley Unix try `ps aux CR` and on Linux, Solaris and other Bell Unix try `ps -elf CR`. It might be necessary to stop your old AIPS session from your new window and then use that window to start a new AIPS.
3. Can you abort AIPS at your window using the appropriate system commands (*i.e.*, `CTRL C` on Unix machines)?
4. If all windows appear dead, then your computer or its X-Windows server may have “crashed.” Try a remote login from another computer. If that works, check on your processes and try to kill the server and other tasks. This should return your computer to a login state. Otherwise, report the problem to your *AIPS* Manager or System Administrator. If you feel you must reboot the system, do so *only* after checking that all current users and the System Administrator (if available) agree that that action is required.
5. If even a reboot fails, report the problem to the System Administrator or hardware experts and go do something else. UNDER NO CIRCUMSTANCES SHOULD YOU ATTEMPT TO REPAIR ANY HARDWARE DEVICES. Such repairs must be performed by trained personnel.

11.3.2 Disk data problems

If you encounter the message `CATOPN: ACCESS DENIED TO DISK n FOR USER mmm`, it means that user *mmm* has not been given access to write (or read) on limited-access disk *n*. The access rights for all disks can be checked by typing `FREESPAC` in the AIPS session. In the list of mounted disks, the Access column can say Alluser, Scratch (scratch files only), Resrvd (limited access including you), and Not you (limited access not including you). If you feel that you should have access to that particular disk, resume using your correct user number or see your *AIPS* Manager about enabling your user number.

If your data set seems to have disappeared, consider

1. Have you set `INDISK` *et al.* (especially `INTYPE`) correctly before running CAT? Type `INP CAT CR` to check. Is `USERID` not set to 0 or your user number?
2. Are you connected to the right *AIPS* computer, if your site has more than one?
3. Are the desired disks mounted for your AIPS session? Type `FREE CR` to see which disks are currently running and which numbers they are assigned in this session. When you attach disks from other computers (using the `da=` option of the `aips` command — § 2.2.3), they are assigned numbers which depend on the list of computers and which may thus vary from session to session.
4. Did you leave your file untouched for a “long” time on a public disk? System managers may have had to delete “old” files to make room for new ones. In this case your data are gone and we hope you made a backup on tape.

The message `write failed, file system is full` will appear when the search for scratch space encounters a disk or disks without enough space. (*AIPS* usually emits messages at this time as well.) This is only a problem when none of the disks available for scratch files has enough space, at which point the task will “die of unnatural causes.” Run the verb `FREESPAC` to see how much disk is available and then review the inputs to the task to make sure that `OUTDISK` and `BADDISK` are set properly. Change them to include disks with space. Check the other adverbs to make sure that you have not requested something silly, such as a 2000-channel cube 8096 on a side. Then try again.

If there simply is not enough space, try some of the things suggested in § 3.6, such as `SCRDEST` to delete orphan scratch files, `DISKU` to find the disk hogs, and, if all else fails, `ZAP` to delete some of your own files.

DISKU may be run with **DOALL** = *n* to list catalog entries that occupy more than *n* Megabytes. This will help identify those files which will yield the most new space when deleted. Your *AIPS* Manager may help you by removing non-*AIPS* files from the *AIPS* data disks. Do not do this yourself unless they are your files.

11.3.3 Printer problems

All *AIPS* print operations now function by writing the output to a disk text file, then queuing the file to a printer, and then sometime later, deleting the file. After the job is queued, the *AIPS* task or verb will display information about the state of the queue. Read this carefully to be sure that the operation was successful and to find out the job number assigned to your print out. If you are concerned that your print job may be lengthy, or expect that you will only need a few numbers from the job, please consider using the **DOCRT** option to look at the display on your terminal or the **OUTPRINT** option to send the display to a file of your choosing without the automatic printing. See § [Z.1.5.3](#) for information about printing such files later.

To find out what jobs are in the spooling queue for the relevant printer, type, at the monitor level:

\$ lpq -P*ppp* C_R

to show printer *ppp*.

\$ lpstat *ppp* C_R

to show printer *ppp* under Solaris, HP, SGI (Sys V systems).

where *ppp* is the name of the printer assigned to you when you began *AIPS*. If the file is still in the queue as job number *nn*, you can type simply

\$ lprm -P*ppp nn* C_R

to remove the job.

\$ cancel *nn* C_R

to remove the job under Sys V systems.

lprm and cancel will announce the names of any files that they remove and are silent if there are no jobs in the queue which match the request.

Since modern printers are capable of swallowing large amounts of input, your job may still be printing even though it is no longer visible in the queue. If you turn off the printer at this stage, you are likely to kill the remainder of your print job and quite possibly one or more other print jobs that followed yours. Use discretion. Do not turn the printer back on if the job is still in the queue. Most systems will start the print job over again after you turn the power back on without doing a lprm or cancel.

If your printout fails to appear

1. Did the print queuing actually work? Review the messages at the end of the verb or task.
2. Did the printout go to a printer other than the one you expected? Was it diverted to a printer used for especially long print jobs or one used for color plots? The messages at the end of the verb or task should show this.
3. Was the printer not working or backed up for so long that the file was deleted before it could be printed? The delay time for deletion is shown at the end of the verb or task. It can be changed by your *AIPS* Manager for future jobs.
4. Was your print job, or that of a user in the queue ahead of you, a large plot? These can take a long time in some PostScript printers (usually indicated by a blinking green light), so be patient.

11.3.4 Tape problems

When *AIPS* does a software **MOUNT** of a magnetic tape, it actually reads the device on most systems. An error messages along the lines of ZMOUN2: Couldn't open tape device ... usually means that you have attempted the **MOUNT** before the device was ready. Wait for all whirring noises and blinking lights to subside

and try again. Remote tape mounts are more fragile. If you get a message such as ZVTP02 connect (INET) : Connection refused, then the tape daemon **TPMON** is probably not running on the remote host. **EXIT** and restart AIPS, specifying the remote host in the **tp=** option (see § 2.2.3). If you are told AMOUNT: **TAPE IS ALREADY MOUNTED BY TPMON**, then there is a chance that you are trying to mount the wrong tape or that someone left the tape device in a mounted state. See § Z.1.5.7 for advice on curing this stand-off between AIPS, which knows that the tape is not mounted, and **TPMON** which knows that it is.

If you are having problems reading and writing a tape, consider

1. Did you actually mount the tape in software from the *AIPS* level with the **MOUNT** verb. A message like ZTPOPN: NO SUCH LOGICAL DEVICE = AMTO_n: indicates that you have not.
2. Have you specified the **INTAPE** or **OUTTAPE** number to correspond with the drive you mounted the tape on?
3. Does your computer have access to tapes on the remote host? The message AIPS **TAPE PERMISSION DENIED ON REMOTE HOST** suggests not. See the *AIPS* Manager for the remote host.
4. Is the tape correctly loaded in the drive and is the drive “on line” (check the **ON LINE** light)?
5. Have you set the density correctly? Some drives need the density to be set by a switch, others have software control. Some try to read the tape and sense the density automatically. Be aware that some drives do not set the density until you actually read or write the tape. Under these circumstances, the density indication on the drive can be misleading. If in doubt, consult your local *AIPS* Manager about the meaning of the tape density indicator lights on the drive you are using.
6. Are you using the correct program to read the tape? If you are unsure of the format of a tape, use the task **PRTTP** to diagnose it for you. It will recognize any format that *AIPS* is able to read.
7. Are you writing to a completely blank tape? This fails sometimes. Or are you writing to an old tape which is new to you? In both cases, try specifying **DOEOT FALSE C_R** and then rerunning the tape-writing program.
8. Has the drive been cleaned recently? Do *not* attempt to clean a drive yourself. Using the wrong cleaning fluid or cleaning the wrong parts of a drive can do serious damage. If you have any doubts, use another drive.
9. Is your tape defective? Tapes can lose oxide or become stretched, creased, or dirty, all of which will cause problems. Try using another tape, if possible.

11.4 Additional recipe

11.4.1 Banana coffeelate

1. Peel and mash 2 ripe **bananas**.
2. Blend in 1/2 teaspoon **vanilla extract**, a few grains **salt**, 1/4 cup **chocolate syrup**, 2 teaspoons **sugar**, and 2 teaspoons instant powdered **coffee**.
3. Add 1½ cups **milk**.
4. Beat with rotary beater or electric mixer until smooth and creamy. Chill.

12 AIPS for the More Sophisticated User

The program AIPS uses the *POPS* language to communicate with you. *POPS* has many capabilities that have been hidden or taken for granted in the previous sections. Once you start to become familiar with *AIPS*, you will need to know more about *POPS* to take full advantage of the powerful features which are available. The first section below describes some of the shortcuts and conventions of *AIPS* while the second section describes program flow control options.. The third section describes multiple features of the *POPS* language including the constructions of procedures. The fourth section addresses the needs of “remote” users of *AIPS*, while the last section discusses how to begin writing your own *AIPS* tasks.

12.1 AIPS conventions

12.1.1 AIPS shortcuts

Some niceties of using *POPS* syntax in *AIPS* are:

1. More than one expression can be put on a line. These expressions must be separated by a semicolon (`;`). Exceptions are `RESTORE`, `GET`, `RUN` and a few other “pseudoverbs” which, with their arguments, must stand alone. For example, `GET MYIMAGE ; INDISK = 1 CR` will ignore the `INDISK = 1`. When in doubt, see the `HELP` files for the pseudoverb to find the restrictions on its use.
2. As in many other systems, recognized keywords in AIPS do not need to be typed in full. You must type only enough of the leading characters to get a unique match. This “minimum-matching” has been exploited throughout this *CookBook*. If in doubt, hit the Tab key. The keyword will be completed if it is unique or the screen will blink. Hit the Tab key a second time to see your choices.
3. The parameter variables in AIPS are called “adverbs.” They are assigned values by the equals verb (`=`), e.g., `INTAPE = 2 CR`. The equals sign may usually be replaced by a space. The exception arises when the variable on the left is a subscripted array element and the expression on the right involves a unary minus or other function reference (e.g., `APARM(3) = -1 ; APARM(4) = SIN(X) CR`).
4. Array adverbs are set to a constant value by putting a single value on the right hand side of the equals sign, e.g., `CELLSIZE = 1.5 CR`. A list of values may be put in the array by putting the list on the right hand side of the `=` sign separated by commas (`,`), e.g., `LEVS = -1, 1, 2, 3, 6, 9 CR`. The commas may be replaced by spaces in most cases. An exception occurs if an element is negative or some other arithmetic expression. Thus, `SHIFT = -19 -2 CR` will produce `SHIFT = -21, -21`.
5. A list of values may also be put in a sequence of scalar adverbs with the `~` (tilde sign) adverb. The main use of this is to overcome the 80-character limit for input lines to *POPS* in assigning values to an array. Thus, for example,
`RASHIFT = -3000, -2500, -2000, -1500, -1000, -500, 0, 500, 1000 CR`
`RASHIFT(10) ~ 1500, 2000, 2500, 3000 CR`
6. Adverbs can be used in arithmetical expressions or set equal to other adverbs, e.g., `OUTNAME = INNAME ; OUTSEQ = 2.5 * INSEQ + 3.`
7. Both upper and lower case letters may be entered by the user; with one exception, *AIPS* is case insensitive. That exception is in adverb character string values, which are converted to upper case by the trailing quote sign. If you omit the trailing quote — and make that the last command on the input line — then case is preserved (and used) in the string.

As an example, these shortcuts allow the following AIPS command sequence:

```
> INNAME = '3C138' CR
> INCLASS = 'IMAGE' CR
> INSEQ = 0 CR
> BLC = 200 , 200 CR
> TRC = 300 , 300 CR
> OUTNAME = '3C138' CR
> OUTSEQ = 5 CR
> GO PRTIM CR
```

to be shortened to:

```
> Inn '3c138' ; INC 'image' ; blc 200 ; Trc 300 CR (Note use of upper and lower case.)
> OUTN INN ; OUTS INS + 5 CR
> go prt1 CR (Task name can be in either case, too.)
```

12.1.2 Data-file names and formats

The physical name of the data file is generated internally, depends on the type of computer, and will not often concern you as a user. You will refer to an image by specifying its disk number, the type of image ('MA' for images, 'UV' for *uv* data), and the following three parts of the image designation:

1. Name — A string of up to 12 legal characters.
2. Class — A string of up to 6 legal characters.
3. Seq — A number between 0 and 46655.

Each of these parts corresponds to separate input adverbs called **INNAME**, **INCLASS**, and **INSEQ** (and their variations). You can choose the image name arbitrarily and sensible choices will reduce other book-keeping. Many programs will choose a reasonable image name if you do not specify one.

A common set of conventions for the name adverbs is used throughout *AIPS*. **INNAME** '' C_R means "accept any image name with the specified class and sequence." **INSEQ** 0 C_R means "accept any image with the specified name and class" or, if only one image is to be used, "accept the image with the specified name and class having the highest sequence number." **OUTNAME** '' C_R means "use the actual **INNAME**." **OUTCLASS** '' C_R means "use the task name," except for tasks that write more than one output image, in which case task-based defaults will be used. **OUTSEQ** 0 C_R means "use a sequence number that is one higher than that of any files currently on disk with the same name and class as the requested output file." The name and class strings also support "wild-card" characters for input and output. This feature is especially powerful in tasks, such as **FITTP**, that can be told to operate on *all* images that match the specified name parameters. Type **HELP INNAME** C_R and **HELP OUTNAME** C_R for details. The verb **NAMEGET**, new in 31DEC12, will convert the **INNAME** set of adverbs to the values that would be used by a task. This is especially useful in complicated procedures.

Only the array data, in the form of 4-byte floating-point numbers, are stored in the image or *uv* data file. The header information is stored separately for each image or *uv* data set. Directory information is stored in a special file, called the Catalog File. Each disk has one such file for each user and it contains directory information for all images and *uv* data sets belonging to that user.

Extension files may be associated with any image data file. Each image can have (in principle) up to fifty types of extension files and up to 46655 "versions" of each type. These subordinate files contain additional information associated with the image and are designated by a two-letter type code. 'HI' is a history file,

‘CC’ is a Clean components file, ‘PL’ is a plot file, ‘AN’ is an antennas file, ‘SL’ is a slice file. In AIPS, an extension file associated with an image is uniquely specified by the usual file-naming adverbs plus the extension file type (adverb **INEXT**) and the version number (adverb **INVERS**). The default convention for **INVERS** is reasonable — on input, zero means the highest (*i.e.*, most recent) version and, on output, the highest plus one.

Array elements in an image are designated by their pixel coordinates (counts). If $M(i, j, k, l, m, n, o)$ is a seven-dimensional array, the (1,1,1,1,1,1) pixel will be associated with the lower left hand corner of the image. The i^{th} (first) coordinate increments fastest and is associated with a column in each plane of the image. The j^{th} (second) coordinate is associated with a row in each plane. The other coordinates allow the image to be generalized to cover up to seven dimensions, *i.e.*, “cubes” and the like. The two adverbs **BLC** for bottom left corner and **TRC** for top right corner let you specify the desired sub-array in up to seven dimensions. When a sub-image is taken from an image, the pixel designation of any image element will usually change.

12.2 Process control features of AIPS

There are a number of tools available to assist you in scheduling and controlling the execution of AIPS tasks. They include taking the input from a text file rather than an interactive terminal, special adverbs to the verb **GO**, and even one or more detached, batch processing queues.

12.2.1 RUN files

If you have some lengthy sequence of commands to give to AIPS, especially if you will have to give essentially the same sequence more than once, you may find it helpful to prepare the sequence in a text file using your favorite and flexible text editor. In particular, it is cumbersome to write and edit procedures more than about five lines long in AIPS because of its primitive internal editor. Long procedures are best written as text files at your computer’s monitor level, where the editing facilities will usually be much better. These text files can be transferred to AIPS easily using the **RUN** file facility.

Any commands that can be typed on the terminal to AIPS can be stored in your text file. The text files may be stored in a disk area of your choosing or in a public area identified by the logical name **\$RUNFIL**. The name of the file must be all upper case letters, followed by a period, followed by your user number as a three-digit “extended-hexadecimal” number with leading zeros. (To translate between decimal and extended hexadecimal, use the AIPS procedures or the AIPS verbs called **EHEX** and **REHEX**.) To use the **RUN** file from your own disk area, define a logical name before starting AIPS to point to the area (see § 3.10.1). Then start up AIPS under your user number and enter

```
> VERSION = 'MYAREA' CR where MYAREA is your disk area, or
> VERSION = ' ' CR if $RUNFIL is to be used
> RUN FILE CR to execute the file named FILE.uuu
```

where *uuu* is your user number in extended hexadecimal with leading zeros to make three digits. The file *FILE.uuu* or, if it does not exist, *FILE.001* will be executed by the above command. Note that minimum match also applies to **RUN** file names.

The first line of a **RUN** file is ignored by AIPS. You should type comments into your **RUN** files to remind you what they are doing. The first line, any line which begins with an * in column one, and all text following a \$ sign (in any line) are treated as comments which will not be compiled or executed by AIPS. All facilities in AIPS such as **GET**, **SAVE**, and **TGET** can be used in **RUN** files. You may even nest **RUN** files within **RUN** files up

to 20 deep, but remember that the pseudoverb **COMPRESS** makes use of the **RUN** process. The full contents of a **SAVE** area may be put in a **RUN** file using **SG2RUN**.

AIPS programmers also provide a few **RUN** files for general use in each release *e.g.*, **VLACALIB**, **VLAACLAL** and **VLARESET**. They are normally used to create procedures helpful to some, but not all, *AIPS* users. These are stored under user number 1, but are available automatically to everyone. To use a procedure from one of them, type, for example:

> **RUN VLAPROCS C_R** to read and execute file **VLAPROCS** in AIPS. The text will be listed as it is read.

This file contains a number of procedure definitions including those named above. To execute the procedure, prepare the input adverbs as needed, and then type

> **VLACALIB C_R** to execute the calibration sequence of the simplified procedure.

AIPS programmers will provide a file with the inputs and help information for all canned procedures. Thus you can do

> **HELP VLACAL C_R** to see help information on the procedure.

> **INP VLACAL C_R** to see the current inputs to the procedure.

The help and inputs functions may require the **RUN** if special adverbs are used by the procedure, but you must do the **RUN** to compile the procedure before you can use it.

12.2.2 More about GO

The verb **GO** is shown in examples throughout this *CookBook*. Don't overlook the fact that **GO**, like all other *AIPS* verbs, has its own inputs. You will be familiar with the ability to specify which task you want to execute either by an immediate argument, *e.g.*, **GO UVSRT C_R**, or by the parameter **TASK**, *e.g.*, **TASK 'UVSRT' ; GO C_R**. **GO** has three further parameters, **DOWAIT**, **VERSION**, and **QCREATE**. The value of **DOWAIT** is passed to the task and instructs it to resume AIPS as soon as possible (**DOWAIT FALSE C_R**) or to resume AIPS only after completing its operations (**DOWAIT TRUE C_R**). The latter option lets the task return a meaningful error code to which AIPS may respond by aborting the current input line, procedure, **FOR** loop, etc. Note that the verb **WAIT 'taskname'** also forces AIPS to wait for a task to complete, but it cannot respond to some failure in that task. For example, the line:

> **GO UVSRT ; WAIT UVSRT ; GO UVMAP C_R**

may cause unwanted images to be generated by **UVMAP** if **UVSRT** fails for lack of disk space or some other reason. However, the line:

> **DOWAIT TRUE ; GO UVSRT ; GO UVMAP C_R**

will not attempt to execute **UVMAP** if **UVSRT** fails. Note that AIPS will not get hung up when a task aborts even if **DOWAIT** is true. *This consideration is particularly important when you do multiple runs of FITTP to write your output tape. If one fails, the tape may even be rewound. If the next scheduled FITTP actually runs, it may write at the beginning of tape, destroying all previous data on the tape!*

VERSION is used to specify which version of the program you wish to execute. You might use this to select the **TST** or **OLD** versions of a task from the **NEW** version of AIPS, for example. **VERSION** also allows you to execute private versions of programs and even to check a list of areas for versions of your program. Type **HELP VERSION C_R** for details. The last adverb, **QCREATE = 1**, instructs tasks to create files quickly. This can save a lot of time, but does not guarantee that the disp space will be available when the task gets around to needing it.

GO has another useful capability. Normally, in order to invoke a verb, you simply type its name, *e.g.*:

> **PRTMSG C_R** to print the message file contents.

However, if you type, instead:

> **GO PRTMSG C_R**

having forgotten that **PRTMSG** is a verb, then AIPS will actually execute:

> **TPUT PRTMSG ; PRTMSG C_R** to save the current **PRTMSG** inputs and then print the contents of the message file.

You can recover those inputs at a later time with **TGET PRTMSG C_R**.

12.2.3 Batch jobs

The **AIPS** batch processor can be used to run jobs outside interactive **AIPS**. The job consists of a set of **AIPS** instructions which do not need user interaction. This excludes the TV, the Tektronix, and the tape-drive oriented tasks and verbs. **RUN** files may be used in batch jobs — as the batch editor facility is also primitive, they are particularly attractive to batch users. Older restrictions on which tasks may be run in queue 1 have been removed in the 15JAN96 release.

The instructions to be executed in the batch processor are prepared in a “workfile.” The workfile can be made while in **AIPS** and detailed instructions are given by typing:

> **HELP BATCHJOB C_R**

A simple example is given here:

> BATQUE = 2 ; BATCLEAR C_R	to select queue 2 and clear its workfile.
> BATCH C_R	to enter batch preparation mode.
< TASK = 'UVSRT' C_R	(Notice < prompt. Begin typing as in AIPS .)
< INN = '3C16' ; INCL = 'UVDATA' C_R	
< INSEQ 1 ; OUTN INN ; OUTCL INCL C_R	
< OUTSEQ = 0 ; SORT = 'XY' C_R	
< GO C_R	(Batch AIPS always waits for a task to finish before continuing.)
< RUN XXXXX C_R	(RUN files are good to use in a batch job.)
< GO C_R	
< ENDBATCH C_R	to leave batch preparation mode type in ENDBATCH spelled out in full.
>	(Resume normal interactive processing.)

To list a batch file, type:

> **BATFLINE = 0 ; BATLIST C_R**

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

> **BATEDIT n C_R**

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

As with procedures (§ 12.3.2), if *n* is an integer, the existing line *n* is overwritten with the line or lines typed before **ENDBATCH**. If *n* is not an integer, the new lines are simply inserted between lines *n* and *n+1*. **BAMODIFY** provides, for workfiles, the same functions as **MODIFY** does for procedures.

Finally, the workfile can be submitted to the batch processor by typing:

> **SUBMIT C_R**

The instructions are sent to a checking program which checks that the input is free of obvious errors. All **RUN** files are expanded and checked. If Checker (the task **AIPSC_m** where *m* is some extended hexadecimal number < Z) approves, the job goes into the **AIPS** job queue, which is managed by **QMNGR*n***. If you change your mind, the job can be removed from the queue and returned to the workfile with the verb **UNQUE**. The batch job may be submitted to any local computer, not just the one on which the current **AIPS** session is running. Adverbs **REMHOST** and **REMQUE** control this option. The remote host must be one sharing the same **AIPS** installation as the current session. Note that the data must also be available to the remote host, but verb **ADDDISK** may be useful either within the batch job or in preparing the data files for the job. **REMHOST** is supported by all relevant batch-related verbs.

Batch has several limitations. First, devices which require interactive use (TV device, Tektronix device, and the tape drives) cannot be used in batch. Also, batch uses a different set of **TPUT** and **TGET** files. Thus, a **TGET** in batch does not get the adverbs from your last interactive use of the specified task. However, the AIPS facilities **GET** and **SAVE** are particularly useful for batch. You can use interactive AIPS to set up and test set(s) of procedures and adverb values and **SAVE** them in named files. These files may then be recovered by batch for the routine processing of large sets of data. This is considerably more convenient than using the batch editor. Note that **SAVE / GET** files may become obsolete with new **AIPS** releases, but that improvements to the **POPS** language have made this quite unlikely.

At present, batch jobs are run after a short delay, on a first-come, first-served basis. After your job has been submitted successfully, type:

> **QUEUES C_R** to list jobs in the queue.

Note the **SUBMIT TIME** for your job. It will not start before that time. The messages generated by your batch job will be printed automatically into a text file. They are kept in your message file, however, and can be reprinted or examined later via **PRTMSG** with **PRNUMB** set to the **AIPS** number of the batch queue. Printer output for batch jobs is concatenated into a file either specified by the user with **OUTPRINT** or a file named **PRTFIL:BATCH_{jjj}.nnn**, where *jjj* is the job number and *nnn* the user number both in extended hexadecimal. Note, this means batch job printouts are concatenated in (normally) one file and are not automatically printed. An interactive AIPS can interact with a batch job via **TELL**; see § 5.3.1.

12.3 AIPS language

AIPS contains a basic set of symbols and keywords which are needed to construct a computer language, as well as the symbols needed by the application code. A list of the basic symbols is given in the help file called **POPSYM**, reproduced below:

Type: Symbols used in the POPS interpretive language

VERB	USE	COMMENTS
----Arithmetic expressions		
+	A + B	Add the expression A to B
-	A - B	Subtract the expression B from A
*	A * B	Multiply the expression A with B
/	A / B	Divide the expression A by B
**	A ** B	Calculate A to the power B
()	(A+B)*C	Grouping expressions as desired
=	A = B	Store the value of B into A
,	A = 3,5,4	Separator of elements in an array
~	A(i) ~ 1,2,3	Store values in A(i),A(i+1)... (change only as many as on RHS)

: TO Equivalent to the verb TO
 ; Separator between AIPS statements

----Logical expressions

>	A > B	A greater than B
<	A < B	A less than B
=	A = B	A equal B (numeric or string)
>=	A >= B	A equal to or greater than B
<=	A <= B	A equal to or less than B
<>	A <> B	A not equal to B (numeric or string)
!	A ! B	A or B
&	A & B	A and B
~	~ A	not A

----String expressions

!!	A !! B	string = string A followed by string B
SUBSTR	SUBSTR(A,i,j)	string = chars i through j of string A
LENGTH	LENGTH(A)	position last non-blank in A
CHAR	CHAR(A)	convert number A to string
VALUE	VALUE(A)	convert string A to number

----Looping constructions

(FOR-TO-BY-END) FOR I=1 TO 7 BY 2
 <any valid set of AIPS syntax>
 END

(WHILE-END) WHILE <any logical expression>
 <any valid set of AIPS syntax>
 END

(IF-THEN-ELSE-END) IF <any logical expression>
 THEN <any valid set of AIPS syntax>
 ELSE <any valid set of AIPS syntax>
 END

----Built-in functions

ACOS	Arc-cosine - output in degrees
ASIN	Arc-sine - output in degrees
ATAN	Arc-tangent (one argument) - output in degrees
ATAN2	Arc-tangent (two arguments) - output in degrees
COS	Cosine (degrees)
SIN	Sine (degrees)
TAN	Tangent (degrees)
EXP	Exponential
LN	Log base e
LOG	Log base 10
SQRT	Square-root
MAX	Maximum i.e. X = MAX (A, B)

MIN	Minimum i.e. X = MIN (A, B)
MODULUS	Root-square sum of two arguments
MOD(A,B)	A - (A/B) * B i.e. remainder of A/B
CEIL(A)	Lowest integer >= A
FLOOR(A)	Highest integer <= A

----Procedure building verbs

PROC	PV	Begin building a procedure
PROCEDUR	PV	Begin building a procedure
LIST	pV	List a procedure
EDIT	PV	Edit a procedure
ENDEDIT	PV	End editing a procedure
ERASE	PV	Delete line(s) of a procedure
MODIFY	PV	Modify a line in a procedure
RETURN	V	Last statement in a procedure
FINISH	PV	End procedure building

----Variable declarations

SCALAR	pV	Declare scalars
ARRAY	pV	Declare arrays
STRING	pV	Declare strings

----Input/Output functions

PRINT	V	Print the following keyword value(s)
TYPE	V	Print the following keyword value(s)
READ	V	Read value(s) from terminal after # prompt

----Other information

CORE	pV	Amount of core left in POPS
COMPRESS	PV	Compress the core area, recovering lost space and acquiring any new vocabulary
CLRTEMP	V	Clear the temp data array
DEBUG	pV	Debug: turns on compiler debug information
DUMP	V	Dump K array on terminal screen
SCRATCH	PV	Remove procedures in POPS
\$	PV	Makes rest of input line a comment

12.3.1 Using *POPS* outside of procedures

POPS variables are either numeric or character valued and may be multi-dimensional arrays. Once created, all variables are available everywhere, *i.e.*, they are global. You may manipulate these variables on the command line using most of the symbols listed above. In fact, you have been doing this while setting the adverbs for all the tasks and verbs described in preceding chapters. The more advanced user may wish to use some of the language features in order to simplify repetitive data processing. Here are some simple examples of uses of the AIPS language:

> TYPE (2 + 5 * 6) C_R
 > TYPE 'X =' ATAN (1.0) C_R

32 is written on the terminal.
 X = 45 is written on the terminal.

> TYPE 'MAPNAME ', INNAME, INCLASS, INSEQ C_R

MAPNAME 3C138 IMAGE 1 is written on the terminal.

The simplest loop capability in AIPS uses the pseudoverbs FOR, TO, and BY for repetitive operations. Such loops are primarily intended for use in “procedures” (see § 12.3.2). If a FOR loop can be typed fully on one input line, it will also work outside the procedure mode. The following example shows how to delete a series of images with the same name and class and with consecutive sequence numbers 1 through 10:

> INNA 'TEST' ; INCL 'IMAGE' C_R

to set (fixed) name parts.

> INDI 1 C_R

to set (fixed) disk number.

> FOR INSEQ = 1 TO 10 ; ZAP ; END C_R

to delete the files.

FOR loops must be terminated with an END. The following example shows how to delete every other file in a catalog with 20 entries:

> FOR INSEQ = 1 TO 20 BY 2 ; GETN(I) ; ZAP ; END C_R

More extensive examples are shown in the sections below on procedures.

In some cases, you may wish to manipulate character strings to give your files meaningful names — particularly if your RUN file or procedure operates repetitively on many similar files. The verbs for character manipulation are listed above. As an example,

> OUTNAME = 'CLEAN' !! CHAR(BLC(3)) C_R

to name each output file after the input image plane.

Note that trailing blanks are ignored. If you wanted a space after CLEAN before the plane number, use

> OUTNAME = 'CLEAN' C_R

to set the basic form.

> SUBSTR (OUTNAME , 7 , 12) = CHAR (BLC(3)) C_R

to alter only the last six characters of OUTNAME.

12.3.2 Procedures

Procedure building is a way to combine keywords in AIPS in any convenient way to obtain useful constructs. For complicated sequences, it is easier to prepare and debug procedures in RUN files (§ 12.2.1) than to prepare them in interactive AIPS. A procedure is given a name, with or without arguments, and then can be treated as an AIPS verb. As an example, consider a procedure to load an image on the TV, set the cursor, and fit for the maximum intensity. You could type the following on your terminal:

> PROC MFIT (I) C_R

to define procedure MFIT with one argument I. (I and J are two dummy adverbs which are already defined in AIPS.)

: GETNAME(I) C_R

(Notice the prompt symbol : . This means that we are in the procedure-building mode.)

: TVLOAD ; IMXY ; MAXFIT C_R

to load the image, produce and read the cursor, and fit the maximum near the cursor position when a TV button is pressed

: RETURN C_R

to designate a return point in the procedure — normally not required at the end of a procedure unless a value is to be left on the stack, i.e., a function.

: FINISH C_R

to designate the end of the procedure-building mode and to get back into the normal (prompt >) mode.

>

Notice the prompt symbol, you are back to interactive input mode.

When you type such a procedure into AIPS, the code is compiled as you type. Most syntax errors are spotted immediately and will unceremoniously dump you out of procedure mode. However, all lines written before the detected error are kept and the procedure editor can be used to continue.

The AIPS procedure editing capabilities are quite primitive. If you want to build procedures longer than about five lines, we therefore recommend using permanent storage files in the “RUN” area, as discussed in § 12.2.1 above.

To list the procedure **MFIT**, type:

> **LIST MFIT C_R**

This will produce the following:

```
1PROC MFIT(I)
2GETNAME(I)
3TVLOD ; IMXY ; MAXFIT
4RETURN
5FINISH
```

The procedure is identical to what you typed, with line numbers added.

Procedures are edited line by line. To edit line 2 in the above procedure, type:

> EDIT MFIT 2 C_R	to enter Procedure editing mode.
; GETNAME(I) ; TVLOD C_R	(Notice prompt symbol ; for procedure-editing mode.) This change replaces the old line 2 adding a TVLOD .
; IMXY ; MAXFIT C_R	to add a line between the changed line 2 and old line 3.
; GETNAME(I+1) C_R	to add yet another line after 2.
> ENDEDIT C_R	to terminate procedure editing.

Listing the modified procedure will give:

```
1PROC MFIT(I)
2GETNAME(I) ; TVLOD
3IMXY ; MAXFIT
4GETNAME(I+1)
5TVLOD ; IMXY; MAXFIT
6RETURN
7FINISH
```

To delete lines *n* through *m* from a procedure, type:

> **ERASE xxxxxxxx n : m C_R** where *xxxxxxxx* is the name of the procedure.

To insert one or more lines between lines 3 and 4 of a procedure, type:

> **EDIT xxxxxxxx 3.5 C_R**
 ; (*Type additional lines as needed.*)
 ; **ENDEDIT C_R**

Notice that the lines are renumbered after any **EDIT** or **ERASE**. Use **LIST** to determine the new line numbers.

The pseudoverb **MODIFY** lets you modify characters within a line of a procedure to correct the line or change its meaning. The grammar is:

> **MODIFY proc-name line-number** where *proc-name* is the name of the procedure and *line-number* is the line number in the procedure as shown by **LIST**.

MODIFY begins by showing the existing line with a ? as a prefix. Then it prompts for input with a ? To keep the character of the original line immediately above the cursor, type a blank (space-bar). To delete that character, type a \$ (dollar-sign). To replace that character, type the new character (to get a new blank

character, type an @ sign). Insertions complicate things. To insert text prior to the character immediately above the cursor, type a \ followed by the desired text followed by another \. You may continue to **MODIFY** the remainder of the line, but you must remember that the current character position in the old line is to the left of the current cursor position by the number of inserted characters (including the 2 \’s). **MODIFY** will display the resulting line of code after you hit a carriage return (C_R) and does not change the line number. Example:

```
> MODIFY ED 2 |CR
?TYPE 'THIS IS EDS PROC'
?           MY@\NEW\      @FOR@EXAMPLE' |CR
TYPE 'THIS IS MY NEW PROC FOR EXAMPLE'
> MODIFY ED 2 |CR
?TYPE 'THIS IS MY NEW PROC FOR EXAMPLE'
?           $$$$      \EDURE,\ |CR
TYPE 'THIS IS MY PROCEDURE, FOR EXAMPLE'
```

More information about procedure building and editing can be found by typing:

```
> HELP PROCEDUR CR
```

Procedure creation and editing uses up the limited memory of the *POPS* processor. When the memory is gone, the message BLEW **CORE!** will appear and you can do no more procedure writing without starting over (*i.e.*, **RESTORE** 0 C_R). **CORE** C_R will tell you how much memory is left. If the memory remaining appears small, try **COMPRESS** to recover the lost memory (in 15JAN96 and later releases). **COMPRESS** might even work after a BLEW **CORE!** if you are lucky.

The procedure **MFIT** can be executed by:

```
> MFIT(n) CR                                where n is the slot number of the appropriate image.
```

(It is assumed that the correct disk unit number has already been set.) This procedure can also be part of another procedure or put in a loop. For example:

```
> FOR I= 1 TO 10 BY 2; MFIT(I) ; END CR
```

will load the TV and fit the maximum for the first ten images on the appropriate disk.

All the syntax available in AIPS is available for use inside procedures *except for certain pseudoverbs*. The “prohibited” pseudoverbs include **SAVE**, **GET**, **STORE**, **RESTORE**, **PROCEDURE**, **EDIT**, **ENDEDIT**, **MODIFY**, **LIST**, **CORE**, **SCRATCH** and **COMPRESS**. Others do not make much sense in procedures, including **MSGKILL**, **DEBUG**, and **ABORTASK**. Other pseudoverbs are, however, particularly useful in procedures. These include **TGET**, **TPUT**, and **GO**.

Several verbs are extremely useful in procedures. To set the image name adverbs to those visible on the TV, use **TVNAME**. When **GETN** accesses an empty slot, an error condition is raised and the procedure dies. To handle this error condition in your procedure, use **EGETN** *n* instead and test the adverb **ERROR** which will be “true” if the slot is empty. **CHKNAME** may be used similarly to check on the existence of files with computed names. Some tasks require image-data dependent inputs. To help handle this in general procedures, the verb **GETHEAD** allows all header parameters to be fetched into adverbs. Type **EXPLAIN GETHEAD** C_R for details. Verbs **GETTHEAD** and **TARGET** perform similar functions for data in table extension files. The verb **GETPOPSN** fetches the current *POPS* number for use in procedures, for example, to allow the same procedure to run concurrently in multiple *AIPS* sessions. The verb **DELAY** will cause AIPS to pause for a specified period of time — **MFIT** above would benefit by pausing to allow the user to see his images. There are numerous arithmetic functions, useful looping constructions, and powerful methods of building arithmetic, logical, and string expressions in *POPS*. See § 12.3 above for a list of these. **CLRTEMP** may be used in procedures which do a lot of looping. It clears the temporary space used to hold substrings and other temporary constants. A procedure that does much string manipulation is likely to overflow this area after a number of iterations. The message BLEW TEMP C! usually accompanies the overflow.

Once a procedure is written and edited, it can be stored in a **SAVE** file for later use. Procedures are lost when

another **GET** file is obtained. Procedures can be stored more permanently in **RUN** files which are described in § 12.2.1 above. To list the names of all procedures currently in your AIPS environment, type:

```
> HELP PROCS CR
```

This will list internal *AIPS* procedures as well as your own.

Several procedures have been built into *AIPS*. In particular, some procedures are defined in the system **RUN** file **VLAPROCS** to aid *routine* calibration of **VLA** data. Currently, these are **VLACALIB**, **VLACLCAL** and **VLARESET**. Similarly, VLBA reductions are aided by the procedures in the file named **VLBAUTIL**. They may be useful templates for your own procedures. If you are unfamiliar with the use of *AIPS* procedures, looking at these system-supplied ones will help you to understand, and see the power of, this feature of *AIPS*.

12.3.3 Writing your own programs with $\mathcal{P}\mathcal{O}\mathcal{P}\mathcal{S}$

You may want to add your own programs to *AIPS*. It is not a trivial matter to generate an *AIPS*-standard FORTRAN program (see **HELP NEWTASK** C_R, the *Going AIPS* manuals, and § 12.6 below). Simple but powerful programs may however be built as procedures that use existing verbs and tasks.

Consider a the following example. (This example is presented as if it were typed into an interactive AIPS. In practice, you will probably prefer to prepare such a complicated procedure as a **RUN** file.) We wish to determine the average value and rms scatter at any pixel location in a set of n images. We shall demand that the n images all have the same **INNAME** and **INCLASS** with sequence numbers between 1 and n . The **RENAME** verb can be used to name the images appropriately. We could call this procedure:

```
AVGRMS (PIXXY, N, AVG, RMS)
```

where

PIXXY	is the pixel location in the images,
N	is the number of images,
AVG	is the average value at the pixel location, and
RMS	is the rms value at the pixel location.

The array adverb **PIXXY** is a standard AIPS adverb, but the variables **N**, **AVG**, and **RMS** are unknown to AIPS. These must be defined before we can write the procedure AVGRMS. This is done by a short dummy procedure which we will call DAVGRMS:

> PROC DAVGRMS C _R	to define dummy procedure.
: SCALAR N, AVG, RMS C _R	to define scalar adverbs.
: FINISH C _R	to exit from dummy procedure.

Now begin the procedure AVGRMS:

> PROC AVGRMS (PIXXY, N, AVG, RMS) C _R	to enter procedure building mode.
: SCALAR SUM, SUM2 C _R	to define more variables.
: ARRAY VAL(20) C _R	to define an array.
: RMS = 0 ; SUM = 0 ; SUM2 = 0 C _R	to zero some variables.
: FOR INSEQ =1 TO N C _R	to begin summing loop.
: QIMVAL C _R	to get pixel value at PIXXY in image INNAME INCLASS INSEQ .
: VAL(INSEQ) = PIXVAL C _R	to save pixel value (placed in PIXVAL by IMVAL) in our array.
: SUM = SUM + PIXVAL C _R	to sum for averaging.
: SUM2 = SUM2 + PIXVAL * PIXVAL C _R	to sum for rms.

```

:   END CR                                to mark end of FOR loop.
:   AVG = SUM / N CR                      to get average value.
:   IF N > 1.5 THEN CR                  to check if N > 1.
:     RMS = SQRT((SUM2 - N*AVG*AVG) / (N * (N-1))) CR to calculate rms if N > 1.
:   ELSE ; TYPE 'N TOO SMALL', N CR      to warn the user.
:     END CR                            to mark end of IF clause.
:   TYPE 'AVG=',AVG,'RMS=',RMS,'AT PIXEL',PIXXY CR
:   TYPE '# ',VAL ','ERROR' CR          to print a heading.
:   FOR INSEQ = 1 TO N CR                to begin another loop.
:     SUM = AVG - VAL(INSEQ) CR          to get residual.
:     TYPE INSEQ, VAL(INSEQ), SUM CR    to print data and residual.
:   END CR                            to mark end of FOR loop.
:   FINISH CR                          to return to regular AIPS mode.
>

```

The above procedure could be run as follows. First fill in the adverbs **INNAME**, **INCLASS** and **PIXXY** with the desired values. Then type:

```
> AVGRMS (PIXXY, n, AVG, RMS) CR
```

where *n* is the number of images to average. The average and rms will be calculated and written on the terminal and in the message file. This procedure could be used by another procedure. Suppose we wanted to determine the average and rms of the pixels within a rectangular area. If we set **BLC** and **TRC** in the usual way to define the rectangular boundary, then the procedure:

```

> PROC AVGARRAY (BLC, TRC) CR          to define new proc.
:   FOR I = BLC(1) TO TRC(1) CR        to loop over x-coordinate
:     FOR J = BLC(2) TO TRC(2) CR      to loop over y-coordinate.
:       PIXXY = I , J CR              to set pixel coordinates for AVGRMS.
:       AVGRMS (PIXXY, N, AVG, RMS) CR
:     END ; END CR                  to end y loop, then x loop.
:   FINISH CR                        to end the proc., RETURN not needed.
>

```

will calculate the average value and rms at this array of pixel locations. Please note that this is just an example. The verb **IMSTAT** performs this function much more efficiently.

12.3.3.1 Special facilities for use in procedures

When a task or verb encounters an error condition, it sets an error flag which normally causes **POPS** to terminate the line or procedure it was executing. You can avoid the termination due to task failure by setting **DOWAIT FALSE** and then doing a **WAIT** on the task. This does not let your procedure know that an error in the task occurred, but it does let you ignore any possible error. Certain verbs have a second version which sets the **ERROR** adverb, rather than the automatic termination flag. These include **CHKNAME** which checks for the existence of an **AIPS** file with specified name parameters, **EGETHEAD** which returns a keyword value from a header or **ERROR** if the keyword is not present, **EGETNAME** which returns **ERROR** if the catalog slot is empty, **COPIXEL** which returns **ERROR** if the coordinate is not in the image, and **SYSTEM** which returns **ERROR** if the function fails. Several fitting verbs, **IMVAL**, **MAXFIT**, **QIMVAL**, **TVFLUX**, and **TVMAXFIT**, return **ERROR** if the fit fails. Your procedure can test **ERROR** after these verbs and take appropriate action.

Numerous verbs return adverb values which may be used in your procedures. These include **COPIXEL**, **COWINDOW**, **EGETHEAD**, **EGETNAME**, **EPOCONV**, **GAMMASET**, the various **GETNAMEs**, **GETHEAD**, **GETHEAD**, **GREAD**,

`IN2TV`, `IMDIST`, `IMPOS`, `IMSTAT`, `IMVAL`, `IMMXY`, `MAXFIT`, `QIMVAL`, `REBOX`, `SET1DG`, `SETSLICE`, `SETXWIN`, `TABGET`, `TK1SET`, `TKBOX`, `TKNBOXS`, `TKSET`, `TKVAL`, `TKWIN`, `TV1SET`, `TVBOX`, `TCOLORS`, `TVDIST`, `TVFLUX`, `TVMAXFIT`, `TVNAME`, `TVPOS`, `TVSET`, `TVSTAT`, and `TVWINDOW`. `GETHEAD` and the table functions `GETTHEAD` and `TABGET` are particularly useful in procedures. The verb `NAMEGET`, new in 31DEC12, will convert the `INNAME` set of adverbs to the values that would be used by a task.

The `RUN` file `VLBAUTIL` provides interesting procedures such as `MAXTAB` which returns the maximum table number of a specified table type. Other procedures in that `RUN` file include ones that return antenna number corresponding to a specific name and ones that test whether a data set appears to be new and appears to contain only VLBA antennas. `MAXTAB` has been made obsolete by the verbs `GETVERS` and `QGETVERS`, new in 31DEC12, which perform the same function, returning adverb `MAXVERS` either with or without messages.

Several tasks now return adverb values. These values can then be used for later computation. These tasks include `CUBIT`, `GAL`, `FINDR`, `IMEAN`, `IMFIT`, `JMFIT`, `RLDIF`, and `SETFC`. The first two of these return image fit parameters, the next two return *uv* and image statistics, the next two return the results of Gaussian fitting, `RLDIF` returns the phase difference between RL and LR polarizations for polarization calibration, and `SETFC` returns imaging parameters including number of facets, cell size, and image size.

There is also a verb named `SYSTEM` which allows the *AIPS* user to fork a command to the host operating system from within *AIPS* including from within procedures. This may be used to delete unwanted text or FITS files, to run `ftp` to fetch data files from the web, to run another copy of *AIPS*, or any of a very large number of other commands. The verb `FILEZAP` will allow you to delete non-*AIPS* files without use of the `SYSTEM` verb.

12.4 Remote use of *AIPS*

AIPS users do not always find themselves seated in front of the main display screen of the computer which they intend to use for their data analysis. *AIPS* provides facilities for such users which depend to some extent on the nature and location of the workstation or terminal at which the user is seated. Nearly seamless function is provided to a user seated at a workstation on the local Ethernet well known to the *AIPS* installation. Substantial capabilities are still available to the user at a more distant workstation capable of X-Windows display, especially if that workstation can also run *AIPS* programs such as the TV server and remote tape server. Even the user at a simple terminal or workstation, capable of emulating a Tektronix 4010, can still get some interactive displays. It is only the users at very simple (and now antique) terminals who will be rather limited in their interactive use of *AIPS*.

12.4.1 Connections via X-Windows

In the following discussion, we will assume that you need to do your computing on a computer called *Server* and that you are sitting in front of a workstation called *MyHost*.

If *Server* and *MyHost* are both on the same local area network and both have the same byte ordering, then they should have *AIPS* installed with both of them shown as being at the same *AIPS* “site.” In this case, you simply `slogin`, `rlogin`, or `telnet` into *Server* from a window on *MyHost* and issue the `aips` command as described in § 2.2.3. The `aips` command will recognize that you are coming in from *MyHost* only if the `$DISPLAY` environment variable is correct (*MyHost*:0). In that case, or if you add `tv=MyHost` to the `aips` command line, the procedure will start the message, graphics, and TV servers on *MyHost* if needed. If you want to share data areas between the two computers, you may add a `da=MyHost` on the command line and *AIPS* will run with all data areas from both machines. (The disk systems must be auto-mountable between

the two computers.) There are two verbs within AIPS which will allow you the equivalent of da=. They are [ADDDISK](#) which will add the disks of a selected computer to the current list of data disks and [REMDISK](#) which will remove the disks of a selected computer from the current list. The aips command also lets you select the most convenient printer within your local area network for use in your *AIPS* session. Other forms of data transfer, including magnetic tapes, will be discussed later.

If *Server* and *MyHost* have different byte orders or are not both on the same local area network, then they cannot be at the same *AIPS* site. If both machines have *AIPS* installed and the versions are compatible, then you may run on *Server* with *MyHost* treated as a “guest.” Again, slogin, rlogin, or telnet into *Server* from a window on *MyHost*. Make sure that the environment variable DISPLAY on *Server* is set to *MyHost*:0 and that *Server* is mentioned in your .rhosts file on *MyHost*. Then issue the usual aips command. You do not have to give the tv= option, but you may give tv=*MyHost* if you wish.. This will start the message, graphics, and TV servers on *MyHost* if needed. If the servers fail to start and messages such as “Cannot start remote TV servers...” appear, then you must start the servers using the aips command on *MyHost*. The displays will work without restarting AIPS on *Server*. Thereafter, give the aips option tvok to *Server* (rather than tv=) to suppress the annoying messages. There is no drawback to being a guest TV; all catalogs and device information are now maintained by [XAS](#) itself. Since the display refreshing is handled locally by programs running in *MyHost*, this level of connection supports nearly full interactivity including such demanding displays as [TVLINK](#) and [TVMOVIE](#).

If *MyHost* does not have *AIPS* installed, then you may run the message, graphics, and TV servers on *Server*, with the X-Windows \$DISPLAY set to *MyHost*:0. You may do this with internet sockets, but this ties up the one instance of the servers allowed to use such sockets. The socially acceptable method uses local Unix sockets so that only current *AIPS* session(s) on *Server* may talk to the windows in *MyHost*. You must ask for this explicitly, setting tv=local. This mode of operation is not encouraged since the display refreshing has to be transmitted over the network, making some of the interactive displays too slow to be useful. Nonetheless, there are circumstances in which this mode of operation is the only one available. You will have to add *Server* as an allowed X host (xhost +*Server*) to use this option. See [HELP AIPS C_R](#) for more information on “local” TVs which, among other things, allow for multiple TVs on a single display screen.

12.4.2 Connections to a terminal

You may do some interactive *AIPS*ing if your workstation window is able to emulate a Tektronix 4010 terminal, or you are at a terminal capable of this emulation. (Note that most xterm displays may be switched between a “Tek mode” and the normal “VT mode” by pressing the Control key and the middle mouse button.) To operate in this mode, log in to *Server* from your terminal or workstation window and start AIPS with the command-line option REMOTE. This option will disable all TV functions, will cause all task messages to come to your terminal or window, and will cause any graphics (“TK”) displays to be sent to your terminal or window. You may display an “image” by creating a contour drawing with [CNTR](#) and then displaying the plot file with [TKPL](#) Interactive cursor verbs and procedures such as [TKXY](#), [TKPOS](#), and [TKWIN](#) may then be used. The old 4010 Tektronix display mixed plotting and text in a less than elegant fashion which is slavishly honored by most emulations. This is unfortunate, but usually does not prevent the display from being used for simple position selection and the like.

The *AIPS* task [TXPL](#) is a powerful tool for remote users without any, or correct, Tektronix 4010 emulation. It reads an extension file of type PL and translates the graphics commands in that file to an alphanumeric display for a “dumb” terminal. [TXPL](#) may be exactly what you need for *AIPS* applications that depend on scanning the *shape* of a plot rather than its fine detail. Common examples are viewing the shape of visibility functions produced by [UVPLT](#) (to guide self-calibration or to diagnose interference) or examining calibration solution plots from [SNPLT](#). [TXPL](#) can also usefully interpret simple contour plots or even grey scales(!) for a remote user. It is often much faster to use [TXPL](#) to diagnose the state of your *AIPS* data processing over

a low-bandwidth link than to use [TKPL](#) to execute a stream of Tektronix graphics instructions (even if you have full Tektronix 4010 emulation).

12.4.3 Remote data connections

Like the message, graphics, and TV servers, there is also a remote data server in *AIPS* in the form of a remote “tape” daemon called [TPMON](#). All computers that run *AIPS* run a copy of [TPMON](#) to serve FITS-disk files plus a copy of [TPMON](#) for each real *AIPS* tape device on the computer. Some computers start these [TPMONs](#) when they are booted while others wait until someone runs *AIPS* on them or orders the tape servers to run by using the `tp=` option for the computer from some other computer (§ 2.2.3). If you are computing on the remote *Server* and wish to use an *AIPS* tape drive on your computer, type on *Server*:

% `aips tp=MyHost CR` to start *AIPS* on *Server* and to start the [TPMONs](#) on *MyHost*.

Then, within *AIPS*, enter:

> <code>REMHOST 'MyHost' C_R</code>	to specify your computer as the tape server.
> <code>REMTAPE n C_R</code>	to specify the <i>n</i> th <i>AIPS</i> tape device on <i>MyHost</i> .
> <code>INTAPE m C_R</code>	to specify a <i>Server</i> tape number one or two higher than the number of real tape devices on <i>Server</i> . These match those shown for <i>REMOTE</i> as you started <i>AIPS</i> ; see § 3.9 for an example.
> <code>MOUNT C_R</code>	to mount the remote tape.

You may then use the remote tape as you would any other tape in *AIPS*. See § 3.9, § 4.1, § 5.1, and § 6.1 for examples of normal tape usage. The verb [TAPES](#) is even able to use [TPMON](#) to tell you what tape devices are available on the remote host.

If you have a FITS-disk file (§ 3.10.3) on *MyHost* which you wish to read into *Server*, then you may set [DATAIN](#) to *MyHost*:*:logical:filename*. Note that double colons connect the host name to the logical name for the disk area and a single colon connects the logical name to the name of the file within that disk area. Note also that the logical name must be one known to [TPMON](#) when it is started. Put your file in a standard area such as FITS, which is known to all of *AIPS*, or create the logical variable in a window on *MyHost* (*e.g.*, with `setenv` or `export` — see § 3.10.1) and then start the [TPMONs](#) from that window with the `aips` command. Similarly, you may use [FITTP](#) to write a FITS-disk file onto *MyHost* of an *AIPS* image or *uv* data set. Enter [DATAOUT](#) with the form shown above for [DATAIN](#).

Unlike the display servers, [TPMON](#) can read and write disk files and magnetic tape devices. Given the hostile environment now found on the Internet, this poses a security problem. Therefore, [TPMON](#) checks every connection request to see if the remote computer has permission to use its services. To do the remote tape operations described above, you must have the *AIPS* Manager for *MyHost* alter the appropriate files to give *Server* permission to use the [TPMONs](#) on *MyHost*. This should have been done already if *Server* and *MyHost* are on the same *AIPS* “site” or are routinely used together.

12.4.4 File transfer connections

The techniques discussed above apply to many computer configurations and to most tools within *AIPS*. They do not, however, handle the outputs of printing and plotting tasks. Nor do they provide much support for small computer systems that have no *AIPS* capability of their own. In such cases, the results of your computations will need to be written onto disk on *Server* and then transferred over the network to *MyHost*.

AIPS does not support remote printers explicitly. However, all *AIPS* tasks and verbs which generate printer output support the **OUTPRINT** adverb. With this adverb, you may specify a disk text file to receive the printer text. If you specify the same file for successive printer verbs or tasks, the outputs will be concatenated. (*AIPS* batch jobs do this automatically to concatenate all printer displays for the job.) You may then copy the text file to *MyHost* for editing, printing, or whatever. If you wish to print the whole file on a PostScript printer on *MyHost*, you may wish to run F2PS on the text file on *Server* and copy the result to *MyHost*. *AIPS* provides a “filter” program to convert plain (or Fortran) text files to PostScript for printing on PostScript printers. The command

```
$ F2PS -nn < file > outfile
```

will convert text file *file* to PostScript format file *outfile*. The parameter *nn* is the number of lines per page used inside *AIPS*; use 97 for a small font in “portrait” form or 61 for a larger font in “landscape” form.

The *AIPS* task **LWPLA** also has an option to write its output to a disk file in encapsulated PostScript form using the **OUTFILE** parameter. Similarly, **TVRGB** and **TVCPS** use the same adverb to write PostScript text files containing their three-color displays. Numerous other *AIPS* tasks offer the option to write details of the operation to a text file specified with **OUTTEXT**. These include **SLICE** (slice), **IMEAN** (histogram), **GAL** (fit results), **POSSM** (spectrum), **FRPLT** (spectrum), **HITEXT** (history), **UVCRS** (*uv*-plane crossings), **CONPL** (convolving functions), etc.

FITS-disk files may be written with **FITTP** and **FITAB** and read with **FITLD**, **UVLOD** and **IMLOD**. For efficiency reasons, these are binary files rather than the text files produced by everything else. *AIPS* table files will have their contents transferred by these tasks along with the main data files. To put an *AIPS* table in a disk file in text form, use **TBOUT** to write a simple text file or **EXTAB** to write a file suitable for database and spreadsheet programs. Files in the form written by **TBOUT** may be read back into *AIPS* with **TBIN**.

To transfer the FITS-disk files and text files between *MyHost* and *Server*, some standard network file transfer must be used. For example, use **ftp** on *MyHost* with

% cd MyArea C _R	to switch to the disk area on <i>MyHost</i> used for your files.
% ftp Server C _R	to start ftp to the remote system.
Name (Server...): loginame C _R	to log in to account <i>loginame</i> .
Password: password C _R	to give the account’s password.
ftp> cd directory C _R	to change to the <i>directory</i> name containing the file on <i>Server</i> .
ftp> binary C _R	to allow reading of a binary file — required for FITS-disk files, okay for text files.
ftp> hash C _R	to get progress symbols as the copy proceeds — a good idea for large files.
ftp> put filename C _R	to send <i>filename</i> from <i>MyHost</i> to <i>Server</i> .
ftp> get anothername C _R	to send <i>anothername</i> from <i>Server</i> to <i>MyHost</i> .
ftp> quit C _R	to exit from ftp .

The files should then be in the desired directories. You may have to rename them, however, to a name in all upper-case letters unless you use the “trick” mentioned in § 3.10.1. The secure copy (**scp**) is preferable if you have a secure connection set up. The files may be compressed with **gzip** before copying and then uncompressed with **gunzip** at the other end. This is particularly effective on text files and images written by **FITAB** with quantization.

12.5 Moving data to a new computer

Lucky users who get a new computer frequently ask “how do I move my data to my new, better machine?” The easiest thing to do is first to install *AIPS* on the new computer. Create at least as many data areas on the new machine as there were on the old. If you are moving between machines of the same byte order (same architecture or Solaris to Mac PPC, Linux to Mac Intel, or vice versa), then do a network `scp` copy (or easier yet a `cp` from cross-mounted disks) to move all data files from *AIPS* disk 1 on the old machine to *AIPS* disk 1 on the new machine. Repeat for the other *AIPS* disks. Note that *AIPS* disk 1 is different; it contains message, `SAVE/GET`, and other special files as well as normal data files. The other data areas may be rearranged if you want, but do not copy two old data areas to one new data area. That will cause extreme confusion and loss of data. Now you are ready to resume *AIPSing*.

But if you are moving from Solaris or Mac PPC to Linux or Mac Intel, or vice versa, then you have more to do. Mount the *AIPS* disks of the old machine from the new machine if possible. Otherwise copy the data from the old machine to new, temporary data areas on the new machine (not the new *AIPS* data areas) being careful to keep each data area separate. Outside of *AIPS*, setup the *AIPS* environment:

% <code>cd \$AIPS_ROOT C_R</code>	to move to the root of all <i>AIPS</i> .
% <code>source LOGIN.CSH C_R</code>	to set the basic environment under C-shell, or
% <code>. LOGIN.SH C_R</code>	to set the basic environment under bash — note the dot.
% <code>\$CDTST C_R</code>	to set the full environment.
% <code>RUN REBYTE C_R</code>	to run a format converter once for each <i>AIPS</i> data area.

Answer the question about range of user number (0 0 `CR` will get you all possible numbers) and then enter the full pathname of the input (old) data area and the corresponding new *AIPS* data area. The program will convert the format of all files. The program — barring software error — will produce correct output for almost all files. There is a small danger that the `SAVE/GET` and `TPUT/TGET` files may have some errors since it is way beyond the scope of `REBYTE` to understand the format of these files.

The traditional way to move between computer architectures is more time consuming. On the old computer, use procedure `WRDISK` to write a data area out as FITS files. The FITS file names encode the old *AIPS* disk and catalog numbers so that more than one disk can be written into the same FITS area. Copy the files to the new computer (or better yet, cross mount the disks) and then use procedure `READISK` on the new computer to read them into the new data area(s). These procedures are obtained via a `RUN WRTPROCS`. Plot and slice files are lost in this process. You can move the `SAVE/GET` files with the verb `SG2RUN` which converts an SG file to a text file which can be `RUN` on the new computer and then `SAVED`. This traditional method works fairly well and can deal with the new computer having fewer *AIPS* data areas than the old one. It also offers the opportunity to do a probably overdue backup of your data. However, the loss of message, plot, and slice files and the added overhead in disk space may matter to you.

12.6 Adding your own tasks to *AIPS*

This Section is a brief guide for the user who wants to modify an existing task in *AIPS* or to write a new task. While it is difficult to write an *AIPS* task from scratch, *AIPS* contains several template tasks that are designed to hide most of the work from the “occasional” programmer. Anyone familiar with FORTRAN and a little of the system services of their local computer should be able to add convenient tasks to their local version of *AIPS* with a little practice and patience. The growing use of binary installations makes this more difficult since local sites are unlikely to have the relevant compilers.

12.6.1 Initial choices to make

The simplest way to write an *AIPS* task is to modify one of the four template tasks, [TAFFY](#), [CANDY](#), [FUDGE](#) or [UVFIL](#). These tasks handle the *AIPS* I/O, contain extensive documentation, need to be modified in only a few well-defined places, and can be easily interfaced to user subroutines. They are limited in their ability to handle many input and output images, however, and generally operate on one image row or one visibility point at a time. Softening of these limitations will be discussed in § [12.6.5](#).

TAFFY — This template task reads an existing image in *AIPS* line by line, modifies the line of data as desired and then creates and writes the modified image in *AIPS*. This template task might, for example, be used to blank pixels in an input image in some specified manner.

CANDY — This task is similar to [TAFFY](#) except the input data are contained in an auxiliary file which is FORTRAN readable and outside *AIPS*. The task can transfer an image in any reasonable format outside *AIPS* into the *AIPS* data structure.

FUDGE — This template task reads an existing *AIPS uv* database optionally applying calibration, editing, and data selection, modifies the results point by point, and then creates and writes this modified output *uv* database in the *AIPS* catalog. This template task might, for example, be used to modify a *uv* data base for which the time parameter has been incorrectly written.

UVFIL — This task is similar to [FUDGE](#) without calibration and editing, except that the input *uv* data are contained in one or two auxiliary files, which are FORTRAN readable and outside *AIPS*. The task is useful for translating visibility data in an arbitrary format into an *AIPS* catalogued *uv* data set, or for computing a *uv* data set from model sources.

If you wish to make a minor change in an existing *AIPS* task, it will be simpler to copy the *AIPS* task itself and modify it as needed. For example, if you wanted to add a option in [COMB](#) to combine two input images in a new way to obtain a resultant image, it would be better to start with [COMB](#) itself than with one of the templates. However, non-trivial changes to major tasks will require a careful look at the code, which is generally well documented and segmented.

Changes to tasks that make plots and/or use the TV and other graphics devices can be tricky, but are useful sometimes. Changes to imaging and deconvolution software should not be attempted unless the changes are almost trivial. Modifying existing verbs or creating new ones require you to find which AU_{xx}.FOR routine is involved and then require the entire AIPS program to be relinked. *Going AIPS* describes some of the considerations. The new pseudoverbs VERB and PSEUDOVB allow you to update your AIPS vocabulary using your own procedure rather than by changing the supplied vocabulary files and executing POPSGN as described in *Going AIPS*. In all cases, do not put your modified code into the standard *AIPS* code areas unless your local *AIPS* Manager agrees and has made a backup copy of the original code and executables. Such modified files may well disturb the *AIPS* updates done by the so-called “Midnight Job.”

12.6.2 Getting started

Even if you have special privileges in *AIPS*, it is wise to generate and link new code in your own user directory rather than in your installation’s designated local *AIPS* directory. After the code has been written and tested, check with your local *AIPS* Manager about installing it in a public area. Decide which template task or other *AIPS* task you want to modify, then follow these instructions after logging into your private area.

Under Unix

```
% source /AIPS/LOGIN.CSH C_R
```

to get the basic *AIPS* system logicals under a c-shell, or

% . /AIPS/LOGIN.SH C _R	to get the basic <i>AIPS</i> system logicals under a korn, bourne, or bash shell. (These assume that your <i>AIPS</i> system home directory is called /AIPS).
% \$CDTST (or \$CDNEW) C _R	to set up the logical assignments for programming.
% PROG <i>TASK</i> C _R	to locate the task called <i>TASK</i> .
% LIBS \$AREA > NTASK.OPT C _R	to create a file of linking information for a task found in area <i>AREA</i> , e.g., APLPGM for <i>TAFFY</i> , APGNOT for <i>FUDGE</i> , <i>CANDY</i> and <i>UVFIL</i> , YPGM for <i>IMEAN</i> , etc.
% cp \$AREA/ <i>TASK.FOR</i> NTASK.FOR C _R	to copy code for <i>TASK</i> into your area and to give it a new name (≤ 5 letters).
% cp \$HLPFIL/ <i>TASK.HLP</i> NTASK.HLP C _R	to copy and rename the inputs/help file for the task.
% setenv MYAIPS ‘pwd’ C _R	to define MYAIPS (must be uppercase) as your disk area for <i>AIPS</i> programs, or
% export MYAIPS=‘pwd’ C _R	to define the MYAIPS environment variable under korn, bash, and bourne shells.

12.6.3 Initial check of code and procedures

Before modifying the task in any way, it is wise to compile, link and execute the unchanged task to check that the original code is sound and that all of the *AIPS* logical assignments have been properly set. The duplicated *AIPS* task should run identically to the original; the template tasks are set up to duplicate the input data set with no changes and default parameters. In this way errors or other problems associated with the generation of new tasks can be found and corrected before getting into the quagmire of bugs that you are about to add. Note that *CANDY* and *UVFIL* require external data files and are therefore not so easily checked.

One change is needed in the FORTRAN program before checking. In the main program, about 60 lines down, there is a data statement which identifies the task.

DATA PRGM /'*TASK*' /

Change this to

DATA PRGM /'NTASK'/

The maximum number of characters in a task name is five and the DATA statement must be in the above form. There is no need to change the help file yet, as long as it is named NTASK.HLP.

To compile and link the task, type:

% COMLINK NTASK NTASK.OPT C_R

To compile the task for debugging, add the DEBUG NOOPT options to the COMLINK line. There should be no error messages and no significant warnings.

After successful compilation and linking, try executing the task. Under Unix, stay logged in to your area. To initiate the AIPS program, type:

% aips C_R

or

% aips debug C_R

to use the debugger (e.g., dbx) on tasks compiled with the DEBUG option. You will have to tell the procedure which debugger you want and that you do not want to start AIPS itself under that debugger.

Your *AIPS* Manager may have to make some arrangements for you to activate *AIPS* from your login. Once you have started *AIPS*, type

> VERSION 'MYAIPS' C _R	to specify the location of inputs and help information and of the task executable.
Set up the input parameters, then	
> INP C _R	to review the input parameters. You should be told AIPS 1: Found in Version=MYAIPS at the start of the inputs display. If not, there is something wrong with the logical or task name or the location of the inputs/help file.
> GO C _R	to run the task. The Found ... line should appear again.

12.6.4 Modifying an AIPS task

If you are modifying an existing task, the only ground rules are to read the task carefully and note the locations where changes must be made. Unlike the template tasks which are organized so that additional code need be inserted in only one or two places, an *AIPS* task may need revisions in a variety of places. Some guidelines are:

1. Change the data statement **DATA** PRGM ... near the beginning of the program and the **PROGRAM** statement at the very beginning, if you have not already done this.
2. Make changes in the introductory text which describes the task. This is particularly important if you are adding or removing adverbs.
3. The subroutine GTPARM obtains the adverb values from the input table in order. If you add or subtract adverbs, change INPRMS, the amount of input information. Be careful in the translation of the adverbs, which are usually listed in a COMMON named /INPARM/.
4. Change code as desired. Try to modify HIstory file entries as well.
5. Liberally sprinkle **PRINT** statements in crucial places to help debug the program. If much of the new code can be put in a subroutine, write and debug the subroutine outside *AIPS*. Then, add it to the task.
6. Revise the file NTASK.HLP. At least, change all references to **TASK** to NTASK! If there are any changes in the adverb list, look at these changes carefully. Further changes in the **HELP** and **EXPLAIN** portions of the file may be needed to document your work for others and for yourself at a later date.
7. Compile and link the modified task following the instructions in the previous section. When it compiles and links without errors, try it out in *AIPS*.

12.6.5 Modifying an AIPS template task.

The template tasks are:

1. **TAFFY** — modifies an existing image file and writes a new image.
2. **CANDY** — writes a new image. Input data from outside *AIPS*.
3. **FUDGE** — modifies an existing *uv* database and writes a new database.
4. **UVFIL** — writes a new *uv* database taking input data from outside *AIPS*.

These tasks are described in detail in Chapter 2 of *Going AIPS*. The template tasks and the code for each are extensively documented there, so only the major points are discussed here.

TAFFY reads a selected subset of an image, one row at a time, to a user interface subroutine DIDDLE. An output image is created, catalogued and filled with values calculated in DIDDLE (or attached subroutines). The dimensionality of the output image need not be the same as the input image.

1. If the output image has identical dimensions to the input image, and an output image row can be generated from each input image row, then **TAFFY** is relatively straight-forward to use and the accompanying documentation should suffice.
2. **TAFFY** has the option of deferring the writing of an output row until some number of input rows have been read. This would allow the output to depend on some function of several, or even all, input rows. Examples would include smoothing in x and y and other sorts of spatial filtering.
3. It is possible to handle several input images and several output images with **TAFFY** by using multi-dimensional images and the axis transposing task, **TRANS**. Suppose you want to calculate the spectral index from a set of four images. First, use FQCUBE to put the four images into one data cube. (**FQCUBE** lets you avoid re-defining the frequency axis as **MCUBE** would require.) Then, transpose the cube so that the frequency axis is first with right ascension and declination as the second and third axes. Use this image as the input to **TAFFY** and use **CPARM** to specify the frequency values. Each input row will then be the intensity at a pixel for the four frequencies — from which the spectral index and other parameters can be calculated. The output dimension can be specified arbitrarily. For example, you might want to write out the spectral index, the error, the curvature and the flux intercept at some fiducial frequency. When the task has completed, transpose the output cube so that the celestial coordinates are again the first two axes and the spectral index and friends are the third axis.
4. Subroutine NEWHED may be modified to require certain axis types, operation codes, etc. It must be used to change the output image dimensions if they are not to be the same as the input. An option to omit from the output image the first input axis is available and must be selected in NEWHED.
5. If you can write and debug outside *AIPS* any subroutines that DIDDLE will call in calculating the output image, you may speed up the debugging of your algorithms.

CANDY lets you create an image one row at a time. Input information is obtained through a file which is external to *AIPS*.

1. Subroutine NEWHED shows an example of how the external file may be defined and how it can be used. The first few records of the external file should contain information for defining the header of the output image and updating the appropriate catalog blocks. The pointers to the *AIPS* catalog are given in Chapter 5 of *Going AIPS*.
2. The subroutine MAKMAP reads further records from the external file until a full row of the output image is obtained.
3. Many adverbs are built into **CANDY**. If you need more or wish to change them, read the information at the beginning of subroutine CANIN.

FUDGE reads an existing uv database point by point and creates a new uv database with the modified data. It can easily be used to make simple changes in a uv database; for example, if the u , v , or w terms are to be recalculated, if all phases are to be changed in sign, etc.

1. If the output data set has different dimensions than the input data set (e.g., by combining several spectral-line channels), changes must be made in the subroutine FUDGIN.

2. To combine several input data points, use the task [UVSRT](#) to put the data points adjacent in the file.
3. If you wish to calculate quantities from an input *uv* data set without creating an output file, use this task. In the subroutine DIDDLE set IRET = -1 to avoid writing an the output *uv* data set.

[UVFIL](#) reads input from one or two FORTRAN-readable files which are outside *AIPS* to create, catalog, and fill a *uv* database into *AIPS*. This task is useful for transcribing *uv* data in an arbitrary format into an *AIPS* database.

1. The task code has copious notes to help the user. The first auxiliary file should contain header information about the observations and the telescopes. This file is read in the subroutine NEWHED.
2. The second auxiliary file contains the actual *uv* data in some format and it is converted into an *AIPS uv* database, point by point, in the subroutine FIDDLE. The comments are extensive and an example is given in the code.

12.6.6 Further remarks

1. Try to use the *AIPS* coding standards as described in *Going AIPS*. This will make your code more readable and more portable. It will also save other *AIPS* programmers lots of work if your new task comes into general use.
2. Declare all variables that you use.
3. While debugging, use the FORTRAN [PRINT](#) *, ... statement in your code to obtain temporary output on your terminal during execution. If you want more permanent output, use the *AIPS* message file facilities with the appropriate message level (4 is recommended for information, 6 for warnings, 8 for errors).
4. You may use a debugger (such as dbx) on *AIPS* tasks by specifying [DEBUG](#) to the COMLINK and aips procedures. Inside AIPS, set the “hidden” adverb [SETDEBUG](#) to 0 for normal operation and to 20 to run tasks with your specified debugger.

12.7 Additional recipes

12.7.1 Going bananas with bananas

1. Garnish a baked ham or ham steak with bananas.
2. Make a quick, rich desert with bananas and cream.
3. Bananas are perfect for lunch boxes. They come in their own wrapper, are easy to eat and mess-less.
4. Slice a banana in half lengthwise, brush with melted butter and bake it until tender; serve it as a “vegetable” with roasted meats or fish. Very Caribbean.
5. Don’t forget old favorites like bananas sliced over cereal, diced in pancake batter, or buried midst the ice cream in a banana split.
6. Slice and stir-fry bananas with carrots, tomatoes and ground beef for a super-quick main dish.

12.7.2 Banana-chocolate tea bread

1. Cream 1/2 cup softened **butter**, gradually add 1 cup **sugar**, beating until light and fluffy. Add 2 **eggs**, one at a time, beating well after each addition.
2. Combine 1 1/2 cups all-purpose **flour**, 2 tablespoons **cocoa**, 1 teaspoon **baking soda**, 1 teaspoon **salt**, and 1/2 teaspoon **cinnamon**; sift together.
3. Stir flour mixture into egg mixture, blending well.
4. Add 1 teaspoon **vanilla extract**; stir in 1 cup mashed **banana**, 1/2 cup **sour cream**, 1/2 cup chopped **walnuts**, and 1/3 cup miniature **semi-sweet chocolate chips**.
5. Spoon batter into two greased and floured 7-1/2 x 3 x 2-inch loaf pans. Bake at 350° F for 55 minutes or until a wooden pick inserted in center comes out clean. Cool in pans 10 minutes, remove from pans and cool completely on a wire rack.

Thanks to Tim D. Culey, Baton Rouge, La. (tsculey@bigfoot.com).

12.7.3 Banana caramel pie

1. Mix 1 cup **dark brown sugar**, 1/4 cup **all-purpose flour**, 1/4 teaspoon **salt** in a saucepan. Stir in 1/4 cup **cold water** and 2 **egg yolks**. Beat until smooth.
2. Gradually stir in 1 cup **boiling water**. Then cook, stirring constantly, about 3 minutes until smoothly thickened.
3. Stir in 1 tablespoon **butter**, 1/2 teaspoon **vanilla**, and 1/4 cup **evaporated milk**. Cool slightly.
4. Pour into pre-baked, cooled 8 or 9 inch **pastry pie shell**.
5. Slightly before serving, slice 4 ripe **bananas** and arrange over filling. Top with whipped cream or with a meringue made out of the 2 left-over egg whites.

12.7.4 Curried shrimp

1. Cook 2 1/2 pounds **shrimp** for 3 minutes. Peel and devein.
2. Heat 1/3 pound **butter** or margarine in large saucepan. Saute 4 chopped **scallions** and 2 cups chopped, peeled **apples** until tender. Stir in 2 tablespoons **curry powder**, 1 tablespoon **ground ginger**, and 1/3 cup **flour**. Stir for 2 minutes. Remove from heat and blend in 3 cups **chicken broth**. Return to heat, cook stirring until mixture boils and thickens.
3. Add 1 pound roasted **cashews**, 1 pound **Turkish apricots**, and, if desired, 2 ounces diced **crystallized ginger** and **raisins**. Cook over low heat for 15 minutes.
4. Add shrimp and mix in.
5. Cut 3 **bananas** into thick slices and add to mixture. Serve over cooked white or curried rice.

Thanks to Chiquita Bananas. See <http://www.jaetzel.de/tim/chiquit.htm>.

13 Current *AIPS* Software

The complete lists of software in *AIPS* are kept up-to-date in certain special files which may then be accessed with the AIPS verb [ABOUT](#). Semi-automatic software makes these listing files using the primary and secondary keywords entered by *AIPS* programmers in the numerous help files. The explanations given for each symbol are also those entered by the programmers as the “one-liner” descriptions of the symbol for which the help file is written. The lists of primary and secondary keywords may be viewed directly by typing:

```
> HELP CATEGORY CR          to view primary keywords  
> HELP SECONDARY CR        to view secondary keywords
```

The help file for [ABOUT](#) is more explanatory, however. The general help file (for [HELP](#) itself) lists a number of general help files which will be of interest. These are also mentioned in the section called [INFORMATION](#) below.

The following sections are verbatim reproductions of the various listing files used by [ABOUT](#) and are roughly current to the 15JAN95 release of *AIPS*. Each section title is the name of the keyword which is used as the [ABOUT](#) topic. Each line within a section lists a task, verb, pseudoverb, procedure, adverb, or [RUN](#) file with a very brief description of its function. Pseudoverbs come in two flavors: those that act roughly like verbs and those that must be treated specially, *i.e.*, that must appear alone on a line or only in certain contexts. The help file for each pseudoverb should clarify the its grammatical limits. Typing

```
> HELP name CR
```

where *name* is one of the entries in the left-hand column, will give more useful information about that *AIPS* symbol.

13.1 ADVERB

ADVERB

Type: General type of POPS symbol

Use: Adverbs are the symbols used to address values. They may be REAL (single-precision floating point), \zxhlp{ARRAY} (multiply-dimensioned REALs), or \zxhlp{STRING} (character strings with or without subscripts). The user may create new adverbs by defining them while typing or editing procedures.

Grammar: Adverb names may be used either in compile mode or in regular execute mode. In the former, their pointers are compiled with the procedure and their values, at the time the procedure is invoked, are used during the execution of the procedure.

Usage examples:

```
\zxhlp{ARRAY2} = \zxhlp{ARRAY1}  
\zxhlp{ARRAY3}(I,J) = 23.6  
\zxhlp{CHAN} = 4  
STRARRY = 'STR1', 'STR2', 'STR3', STRAR2  
DUM3 (24, I, STRARRY)  
===== Character string data must be enclosed in quotes!  
===== This allows the compiler to tell data from adverb  
names and allows embedded special characters.  
*****
```

List of ADVERBs

\zxhlp{ALIAS}	adverb to alias antenna numbers to one another
\zxhlp{ALLOKAY}	specifies that initial conditions have been met.
\zxhlp{ANTENNAS}	Antennas to include/exclude from the task or verb
\zxhlp{ANTNAME}	A list of antenna (station) names
\zxhlp{ANTUSE}	Antennas to include/exclude from the task or verb
\zxhlp{ANTWT}	Antenna Weights for UV data correction in Calibration
\zxhlp{APARM}	General numeric array adverb used many places
\zxhlp{ARRAY1}	General scratch array adverb
\zxhlp{ARRAY2}	General scratch array adverb
\zxhlp{ARRAY3}	General scratch array adverb
\zxhlp{ASDMFILE}	Full path to \zxhlp{EVLA} \zxhlp{ASDM}/BDF directory
\zxhlp{ASPMM}	Plot scaling parameter - arc seconds per millimeter on plot
\zxhlp{AVGCHAN}	Controls averaging of spectral channels
\zxhlp{AVGIF}	Controls averaging of IF channels
\zxhlp{AVOPTION}	Controls type or range of averaging done by a task
\zxhlp{AX2REF}	Second reference pixel number
\zxhlp{AXINC}	Axis increment - change in coordinate between pixels
\zxhlp{AXREF}	Reference pixel number
\zxhlp{AXTYPE}	Type of coordinate axis
\zxhlp{AXVAL}	Value of axis coordinate at reference pixel
\zxhlp{BADDISK}	specifies which disks are to be avoided for scratch files
\zxhlp{BAND}	specifies the approximate frequency of UV data to be selected
\zxhlp{BANDPOL}	specifies polarizations of individual IFs
\zxhlp{BASELINE}	specifies which antenna pairs are to be selected/deselected
\zxhlp{BATFLINE}	specifies starting line in a batch work file
\zxhlp{BATNLINE}	specifies the number of lines to process in a batch work file
\zxhlp{BATQUE}	specifies the desired batch queue
\zxhlp{BCHAN}	sets the beginning channel number
\zxhlp{BCOMP}	gives beginning component number for multiple fields
\zxhlp{BCOUNT}	gives beginning location for start of a process
\zxhlp{BDROP}	gives number of points dropped at the beginning
\zxhlp{BIF}	gives first IF to be included
\zxhlp{BITER}	gives beginning point for some iterative process
\zxhlp{BLC}	gives lower-left-corner of selected subimage
\zxhlp{BLOCKING}	specifies blocking factor to use on e.g. tape records
\zxhlp{BLVER}	specifies the version of the baseline-calibration table used
\zxhlp{BMAJ}	gives major axis size of beam or component
\zxhlp{BMIN}	gives minor axis size of beam or component
\zxhlp{BOXFILE}	specifies name of Clean box text file
\zxhlp{BOX}	specifies pixel coordinates of subarrays of an image
\zxhlp{BPA}	gives position angle of major axis of beam or component
\zxhlp{BPARM}	general numeric array adverb used too many places
\zxhlp{BPASSPRM}	Control adverb array for bandpass calibration
\zxhlp{BPRINT}	gives beginning location for start of a printing process
\zxhlp{BPVER}	specifies the version of the bandpass table to be applied
\zxhlp{BWSMEAR}	amount of bandwidth smearing correction to use
\zxhlp{CALCODE}	specifies the type of calibrator to be selected
\zxhlp{CALIN}	specifies name of input disk file usually with calibration data
\zxhlp{CALSOUR}	specifies source names to be included in calibration
\zxhlp{CATNO}	Specifies AIPS catalog slot number range
\zxhlp{CBPLOT}	selects a display of a Clean beam full width at half maximum
\zxhlp{CCBOX}	specifies pixel coordinates of subarrays of an image
\zxhlp{CELLSIZE}	gives the pixel size in physical coordinates
\zxhlp{CHANNEL}	sets the spectral channel number
\zxhlp{CHANSEL}	Array of start, stop, increment channel numbers to average the increment between selected channels
\zxhlp{CHINC}	specifies subarrays of an image for Clean to search
\zxhlp{CLBOX}	

\zxhlp{CLCORPRM}	Parameter adverb array for task \zxhlp{CLCOR}
\zxhlp{CLEV}	Contour level multiplier in physical units
\zxhlp{CLINT}	CL table entry interval
\zxhlp{CMETHOD}	specifies the method by which the uv model is computed
\zxhlp{CMODEL}	specifies the method by which the uv model is computed
\zxhlp{CODETYPE}	specifies the desired operation type
\zxhlp{COLORS}	specifies the desired TV colors
\zxhlp{COMMENT}	64-character comment string
\zxhlp{CON3COL}	Controls use of full 3-color graphics for contouring
\zxhlp{CONFIG}	Configuration ID number within an \zxhlp{EVLA} \zxhlp{ASDM}/BDF data set
\zxhlp{COOINC}	Celestial axes increment: change in coordinate between pixels
\zxhlp{COORDINA}	Array to hold coordinate values
\zxhlp{COOREF}	Reference pixel number for two coordinate axes
\zxhlp{COOTYPE}	Celestial axes projection type
\zxhlp{COPIES}	sets the number of copies to be made
\zxhlp{CPARM}	general numeric array adverb used many places
\zxhlp{CROWDED}	allows a task to perform its function in a crowded fashion
\zxhlp{CTYPE}	specifies type of component
\zxhlp{CUTOFF}	specifies a limit below or above which the operation ends
\zxhlp{DARKLINE}	The level at which vectors are switched from light to dark
\zxhlp{DATA2IN}	specifies name of input FITS disk file
\zxhlp{DATAIN}	specifies name of input FITS disk file
\zxhlp{DATAOUT}	specifies name of output FITS disk file
\zxhlp{DCHANSEL}	Array of start, stop, increment channel #S + IF to avoid
\zxhlp{DCODE}	General string adverb
\zxhlp{DDISK}	Determines where input \zxhlp{DDT} data is found
\zxhlp{DDTSIZE}	Determines which type of \zxhlp{DDT} is \zxhlp{RUN}.
\zxhlp{DECIMAL}	specifies if something is in decimal format
\zxhlp{DECSHIFT}	gives Y-coordinate shift of an image center from reference
\zxhlp{DEFER}	Controls when file creation takes place
\zxhlp{DELCORR}	specifies whether VLBA delay corrections are to be used
\zxhlp{DELTAX}	Increment or size in X direction
\zxhlp{DELTAY}	Increment or size in Y direction
\zxhlp{DENSITY}	gives the desired tape density
\zxhlp{DENUMB}	a scalar decimal number
\zxhlp{DETIME}	specifies a time interval for an operation (destroy, batch)
\zxhlp{DIGICOR}	specifies whether VLBA digital corrections are to be applied
\zxhlp{DIST}	gives a distance - \zxhlp{PROFL} uses as distance to observer
\zxhlp{DO3COL}	Controls whether full 3-color graphics are used in a plot
\zxhlp{DO3DIMAG}	specifies whether uvw's are reprojected to each field center
\zxhlp{DOACOR}	specifies whether autocorrelation data are included
\zxhlp{DOALIGN}	specifies how two or more images are aligned in computations
\zxhlp{DOALL}	specifies if an operation is done once or for all matching
\zxhlp{DOALPHA}	specifies whether some list is alphabetized
\zxhlp{DOAPPLY}	Flag to indicate whether an operation is applied to the data
\zxhlp{DOARRAY}	specifies if subarrays are ignored or the information used
\zxhlp{DOBAND}	specifies if/how bandpass calibration is applied
\zxhlp{DOBLANK}	controls handling of blanking
\zxhlp{DOBTWEEN}	Controls smoothing between sources in calibration tables
\zxhlp{DOCALIB}	specifies whether a gain table is to be applied or not
\zxhlp{DOCAT}	specifies whether the output is saved (cataloged) or not
\zxhlp{DOCELL}	selects units of cells over angular unit
\zxhlp{DOCENTER}	selects something related to centering
\zxhlp{DOCIRCLE}	select a "circular" display (i.e. trace coordinates, ...)
\zxhlp{DOCOLOR}	specifies whether coloring is done
\zxhlp{DOCONCAT}	selects concatenated or individual output files
\zxhlp{DOCONFIRM}	selects user confirmation modes of repetitive operation
\zxhlp{DOCONT}	selects a display of contour lines

\zxhlp{DOCRT}	selects printer display or CRT display (giving width)
\zxhlp{DODARK}	specifies whether "dark" vectors are plotted dark or light
\zxhlp{DODELAY}	selects solution for phase/amplitude or delay rate/phase
\zxhlp{DOEBAR}	Controls display of estimates of the uncertainty in the data
\zxhlp{DOEOF}	selects end-of-file writing or reading until
\zxhlp{DOEOT}	selects tape positioning before operation: present or EOI
\zxhlp{DOFIT}	Controls which antennas are fit by what methods
\zxhlp{DOFLAG}	Controls closure cutoff in gain solutions and flagging
\zxhlp{DOFRACT}	Tells whether to compute a fraction or ratio
\zxhlp{DOGREY}	selects a display of a grey-scale image
\zxhlp{DOGRIDCR}	selects correction for gridding convolution function
\zxhlp{DOHIST}	selects a histogram display
\zxhlp{DOHMS}	selects sexagesimal (hours-mins-secs) display format
\zxhlp{DOIIFS}	controls functions done across IFs
\zxhlp{DOINVERS}	selects opposite of normal function
\zxhlp{DOKEEP}	specifies if something is kept or deleted
\zxhlp{DOMAX}	selects solutions for maxima of models
\zxhlp{DOMODEL}	selects display of model function
\zxhlp{DONEWTAB}	do we make new tables, use a new table format, etc.
\zxhlp{DOOOUTPUT}	selects whether output image or whatever is saved / discarded
\zxhlp{DOPLOT}	Controls plotting of something
\zxhlp{DOPOL}	selects application of any polarization calibration
\zxhlp{DOPOS}	selects solutions for positions of model components
\zxhlp{DOPRINT}	selects printer display or CRT display (giving width)
\zxhlp{DORESID}	selects display of differences between model and data
\zxhlp{DOROBUST}	Controls method of averaging - simple mean/rms or robust
\zxhlp{DOSCALE}	specifies if a scaling operation of some sort is to be performed
\zxhlp{DOSCAN}	specifies if a scan-related operation is to be done
\zxhlp{DOSLICE}	selects display of slice data
\zxhlp{DOPIX}	selects solutions for spectral index of model components
\zxhlp{DOSTOKES}	selects options related to polarizations
\zxhlp{DOTABLE}	selects use of table-format for data
\zxhlp{DOTV}	selects use of TV display option in operation
\zxhlp{DOTWO}	do we make two of something
\zxhlp{DOUVCOMP}	selects use of compression in writing UV data to disk
\zxhlp{DOVECT}	selects display of polarization vectors
\zxhlp{DOWAIT}	selects wait-for-completion mode for running tasks
\zxhlp{DOWEDGE}	selects display of intensity step wedge
\zxhlp{DOWEIGHT}	selects operations with data weights
\zxhlp{DOWIDTH}	selects solution for widths of model components
\zxhlp{DPARM}	General numeric array adverb used many places
\zxhlp{ECHAN}	define an end for a range of channel numbers
\zxhlp{ECOUNT}	give the highest count or iteration for some process
\zxhlp{EDGSKP}	Determines border excluded from comparison or use
\zxhlp{EDROP}	number of points/iterations to be omitted from end of process
\zxhlp{EFACTOR}	scales some error analysis process
\zxhlp{EHNUMB}	an extended hexadecimal "number"
\zxhlp{EIF}	last IF number to be included in operation
\zxhlp{EPRINT}	gives location for end of a printing process
\zxhlp{ERROR}	was there an error
\zxhlp{EXPERT}	specifies an user experience level or mode
\zxhlp{FACTOR}	scales some display or CLEANing process
\zxhlp{FGAUSS}	Minimum flux to Clean to by widths of Gaussian models
\zxhlp{FITOUT}	specifies name of output text file for results of fitting
\zxhlp{FLAGVER}	selects version of the flagging table to be applied
\zxhlp{FLDSIZE}	specifies size(s) of images to be processed
\zxhlp{FLMCOMM}	Comment for film recorder image.
\zxhlp{FLUX}	gives a total intensity value for image/component or to limit

\zxhlp{FMAX}	specifies peak values of model components - results of fits
\zxhlp{FORMAT}	gives a format code number: e.g. FITS accuracy required
\zxhlp{FOV}	Specifies the field of view
\zxhlp{FPARM}	General numeric array adverb used in modeling
\zxhlp{FPOS}	specifies pixel positions of fit model components
\zxhlp{FQCENTER}	specifies that the frequency axis should be centered
\zxhlp{FQTOL}	Frequency tolerance with which FQ entries are accepted.
\zxhlp{FREQID}	Frequency Identifier for frequency, bandwidth combination
\zxhlp{FSHIFT}	specifies a position shift - output from fitting routines
\zxhlp{FSIZE}	file size in Megabytes
\zxhlp{FUNCTYPE}	specifies type of intensity transfer function
\zxhlp{FWIDTH}	gives widths of model components - results of fitting
\zxhlp{GAINERR}	gives estimate of gain uncertainty for each antenna
\zxhlp{GAIN}	specifies loop gain for deconvolutions
\zxhlp{GAINUSE}	specifies output gain table or gain table applied to data
\zxhlp{GAINVER}	specifies the input gain table
\zxhlp{GCVER}	specifies the version of the gain curve table used
\zxhlp{GG}	spare scalar adverb for use in procedures
\zxhlp{GMAX}	specifies peak values of model components
\zxhlp{GPOS}	specifies pixel positions of model components
\zxhlp{GRADDRES}	specifies user's home address for replies to gripes
\zxhlp{GRCHAN}	specifies the TV graphics channel to be used
\zxhlp{GREMAIL}	gives user's e-mail address name for reply to gripe entry
\zxhlp{GRNAME}	gives user's name for reply to gripe entry
\zxhlp{GRPHONE}	specifies phone number to call for questions about a gripe
\zxhlp{GUARD}	portion of UV plane to receive no data in gridding
\zxhlp{GWIDTH}	gives widths of model components
\zxhlp{HIEND}	End record number in a history-file operation
\zxhlp{HISTART}	Start record number in a history-file operation
\zxhlp{ICHANSEL}	Array of start, stop, increment channel #S + IF to average
\zxhlp{ICUT}	specifies a cutoff level in units of the image
I	spare scalar adverb for use in procedures
\zxhlp{IM2PARM}	Specifies enhancement parameters for OOP-based imaging: 2nd set
\zxhlp{IMAGRPRM}	Specifies enhancement parameters for OOP-based imaging
\zxhlp{IMSIZE}	specifies number of pixels on X and Y axis of an image
\zxhlp{IN2CLASS}	specifies the "class" of the 2nd input image or data base
\zxhlp{IN2DISK}	specifies the disk drive of the 2nd input image or data base
\zxhlp{IN2EXT}	specifies the type of the 2nd input extension file
\zxhlp{IN2FILE}	specifies name of a disk file, outside the regular catalog
\zxhlp{IN2NAME}	specifies the "name" of the 2nd input image or data base
\zxhlp{IN2SEQ}	specifies the sequence # of the 2nd input image or data base
\zxhlp{IN2TYPE}	specifies the type of the 2nd input image or data base
\zxhlp{IN2VERS}	specifies the version number of the 2nd input extension file
\zxhlp{IN3CLASS}	specifies the "class" of the 3rd input image or data base
\zxhlp{IN3DISK}	specifies the disk drive of the 3rd input image or data base
\zxhlp{IN3EXT}	specifies the type of the 3rd input extension file
\zxhlp{IN3NAME}	specifies the "name" of the 3rd input image or data base
\zxhlp{IN3SEQ}	specifies the sequence # of the 3rd input image or data base
\zxhlp{IN3TYPE}	specifies the type of the 3rd input image or data base
\zxhlp{IN3VERS}	specifies the version number of the 3rd input extension file
\zxhlp{IN4CLASS}	specifies the "class" of the 4th input image or data base
\zxhlp{IN4DISK}	specifies the disk drive of the 4th input image or data base
\zxhlp{IN4NAME}	specifies the "name" of the 4th input image or data base
\zxhlp{IN4SEQ}	specifies the sequence # of the 4th input image or data base
\zxhlp{IN4TYPE}	specifies the type of the 4th input image or data base
\zxhlp{IN5CLASS}	specifies the "class" of the 5th input image or data base
\zxhlp{IN5DISK}	specifies the disk drive of the 5th input image or data base
\zxhlp{IN5NAME}	specifies the "name" of the 5th input image or data base

\zxhlp{IN5SEQ}	specifies the sequence # of the 5th input image or data base
\zxhlp{IN5TYPE}	specifies the type of the 5th input image or data base
\zxhlp{INCLASS}	specifies the "class" of the 1st input image or data base
\zxhlp{INDISK}	specifies the disk drive of the 1st input image or data base
\zxhlp{INEXT}	specifies the type of the 1st input extension file
\zxhlp{INFILE}	specifies name of a disk file, outside the regular catalog
\zxhlp{INLIST}	specifies name of input disk file, usually a source list
\zxhlp{INNAME}	specifies the "name" of the 1st input image or data base
\zxhlp{INSEQ}	specifies the sequence # of the 1st input image or data base
\zxhlp{INTAPE}	specifies the input tape drive number
\zxhlp{INTERPOL}	specifies the type of averaging done on the complex gains
\zxhlp{INTEXT}	specifies name of input text file, not in regular catalog
\zxhlp{INTPARM}	specifies the parameters of the frequency interpolation function
\zxhlp{INTYPE}	specifies the type of the 1st input image or data base
\zxhlp{INVERS}	specifies the version number of the 1st input extension file
\zxhlp{IOTAPE}	Determines which tape drive is used during a \zxhlp{DDT} \zxhlp{RUN}
\zxhlp{ISCALIB}	states that the current source is a point-source calibrator
J	spare scalar adverb for use in procedures
\zxhlp{JOBNUM}	specifies the batch job number
\zxhlp{KEYSTRNG}	gives contents of character-valued keyword parameter
\zxhlp{KEYTYPE}	Adverb giving the keyword data type code
\zxhlp{KEYVALUE}	gives contents of numeric-valued keyword parameter
\zxhlp{KEYWORD}	gives name of keyword parameter - i.e. name of header field
\zxhlp{LABEL}	selects a type of extra labeling for a plot
\zxhlp{LEVS}	list of multiples of the basic level to be contoured
\zxhlp{LPEN}	specifies the "pen width" code # => width of plotted lines
\zxhlp{LTYPE}	specifies the type and degree of axis labels on plots
\zxhlp{MAPDIF}	Records differences between \zxhlp{DDT} test results and standards
\zxhlp{MAXPIXEL}	maximum pixels searched for components in Clark CLEAN
\zxhlp{MDISK}	Determines where input \zxhlp{DDT} data is found
\zxhlp{MINAMPER}	specifies the minimum amplitude error prior to some action
\zxhlp{MINANTEN}	states minimum number of antennas for a solution
\zxhlp{MINPATCH}	specifies the minimum size allowed for the center of the beam
\zxhlp{MINPHSER}	specifies the minimum phase error prior to some action
\zxhlp{NAXIS}	Axis number
\zxhlp{NBOXES}	Number of boxes
\zxhlp{NCCBOX}	Number of clean component boxes
\zxhlp{NCHAN}	Number of spectral channels in each spectral window
\zxhlp{NCHAV}	Number of channels averaged in an operation
\zxhlp{NCOMP}	Number of CLEAN components
\zxhlp{NCOUNT}	General adverb, usually a count of something
\zxhlp{NDIG}	Number of digits to display
\zxhlp{NFIELD}	The number of fields imaged
\zxhlp{NFILES}	The number of files to skip, usually on a tape.
\zxhlp{NGAUSS}	Number of Gaussians to fit
\zxhlp{NIF}	Number of IFs (spectral windows) in a data set
\zxhlp{NITER}	The number of iterations of a procedure
\zxhlp{NMAPS}	Number of maps (images) in an operation
\zxhlp{NOISE}	estimates the noise in images, noise level cutoff
\zxhlp{NORMALIZ}	specifies the type of gain normalization if any
\zxhlp{NPICE}	The number of pieces to make
\zxhlp{NPLOTS}	gives number of plots per page or per job
\zxhlp{NPOINTS}	General adverb giving the number of something
\zxhlp{NPRINT}	gives number of items to be printed
\zxhlp{NTHREAD}	Controls number of threads used by multi-threaded processes in \zxhlp{OBIT}
\zxhlp{NUMTELL}	selects POPS number of task which is the target of a \zxhlp{TELL}
\zxhlp{NX}	General adverb referring to a number of things in the Y direction
\zxhlp{NY}	General adverb referring to a number of things in the Y direction

\zxhlp{OBJECT}	The name of an object
\zxhlp{OBOXFILE}	specifies name of output Clean box text file
\zxhlp{OFFSET}	General adverb, the offset of something.
\zxhlp{OFMFILE}	specifies the name of a text file containing OFM values
\zxhlp{ONEBEAM}	specifies whether one beam is made for all facets or one for each
\zxhlp{ONEFREQ}	states that the current CC model was made with one frequency
\zxhlp{OPCODE}	General adverb, defines an operation
\zxhlp{OPTELL}	The operation to be passed to a task by \zxhlp{TELL}
\zxhlp{OPTYPE}	General adverb, defines a type of operation.
\zxhlp{ORDER}	Adverb used usually to specify the order of polynomial fit
\zxhlp{OUT2CLAS}	The class of a secondary output file
\zxhlp{OUT2DISK}	The disk number of a secondary output file.
\zxhlp{OUT2NAME}	The name of a secondary output file.
\zxhlp{OUT2SEQ}	The sequence of a secondary output file.
\zxhlp{OUTCLASS}	The class of an output file
\zxhlp{OUTDISK}	The disk number of an output file.
\zxhlp{OUTFGVER}	selects version of the flagging table to be written
\zxhlp{OUTFILE}	specifies name of output disk file, not in regular catalog
\zxhlp{OUTNAME}	The name of an output file.
\zxhlp{OUTPRINT}	specifies name of disk file to keep the printer output
\zxhlp{OUTSEQ}	The sequence of an output file.
\zxhlp{OUTTAPE}	The output tape drive number.
\zxhlp{OUTTEXT}	specifies name of output text file, not in regular catalog
\zxhlp{OUTVERS}	The output version number of an table or extension file.
\zxhlp{OVERLAP}	specifies how overlaps are to be handled
\zxhlp{OVRSWTCH}	specifies when \zxhlp{IMAGR} switches from \zxhlp{OVERLAP} >= 2 to \zxhlp{OVERLAP} = 1 mode
\zxhlp{PBPARM}	Primary beam parameters
\zxhlp{PBSIZE}	estimates the primary beam size in interferometer images
\zxhlp{PCUT}	Cutoff in polarized intensity
\zxhlp{PDVER}	specifies the version of the spetral polarization table to use
\zxhlp{PHASPRM}	Phase data array, by antenna number.
\zxhlp{PHAT}	Prussian hat size
\zxhlp{PHSLIMIT}	gives a phase value in degrees
\zxhlp{PIX2VAL}	An image value in the units specified in the header.
\zxhlp{PIX2XY}	Specifies a pixel in an image
\zxhlp{PIXAVG}	Average image value
\zxhlp{PIXRANGE}	Range of pixel values to display
\zxhlp{PIXSTD}	RMS pixel deviation
\zxhlp{PIXVAL}	Value of a pixel
\zxhlp{PIXXY}	Specifies a pixel location.
\zxhlp{PLCOLORS}	specifies the colors to be used
\zxhlp{PLEV}	Percentage of peak to use for contour levels
\zxhlp{PLVER}	specifies the version number of a PL extension file
\zxhlp{PMODEL}	Polarization model parameters
\zxhlp{POL3COL}	Controls use of full 3-color graphics for polarization lines
\zxhlp{POLANGLE}	Intrinsic polarization angles for up to 30 sources
\zxhlp{POLPLOT}	specifies the desired polarization ratio before plotting.
\zxhlp{PRIORITY}	Limits prioroty of messages printed
\zxhlp{PRNUMBER}	POPS number of messages
\zxhlp{PRSTART}	First record number in a print operation
\zxhlp{PRTASK}	Task name selected for printed information
\zxhlp{PRTIME}	Time limit
\zxhlp{PRTLEV}	Specified the amount of information requested.
\zxhlp{PRTLIMIT}	specifies limits to printing functions
\zxhlp{QCREATE}	adverb controlling the way large files are created
\zxhlp{QUAL}	Source qualifer
\zxhlp{QUANTIZE}	Quantization level to use
\zxhlp{RADIUS}	Specify a radius in an image

\zxhlp{RASHIFT}	Shift in RA
\zxhlp{REASON}	The reason for an operation
\zxhlp{REFANT}	Reference antenna
\zxhlp{REFDATE}	To specify the initial or reference date of a data set
\zxhlp{REMHOST}	gives the name of another computer which will provide service
\zxhlp{REMQUE}	specifies the desired batch queue on a remote computer
\zxhlp{REMTAPE}	gives the number of another computer's tape device
\zxhlp{RESTFREQ}	Rest frequency of a transition
\zxhlp{REWEIGHT}	Reweighting factors for UV data weights.
\zxhlp{RGBCOLOR}	specifies the desired TV graphics color
\zxhlp{RGBGAMMA}	specifies the desired color gamma corrections
\zxhlp{RGBLEVS}	colors to be applied to the contour levels
\zxhlp{RMSLIMIT}	selects things with RMS above this limit
\zxhlp{ROBUST}	Uniform weighting "robustness" parameter
\zxhlp{ROMODE}	Specified roam mode
\zxhlp{ROTATE}	Specifies a rotation
\zxhlp{RPARM}	General numeric array adverb used in modeling
\zxhlp{SAMPTYPE}	Specifies sampling type
\zxhlp{SCALR1}	General adverb
\zxhlp{SCALR2}	General adverb
\zxhlp{SCALR3}	General adverb
\zxhlp{SCANLENG}	specify length of "scan"
\zxhlp{SCUTOFF}	noise level cutoff
\zxhlp{SEARCH}	Ordered list of antennas for fring searches
\zxhlp{SELBAND}	Specified bandwidth
\zxhlp{SELFREQ}	Specified frequency
\zxhlp{SHIFT}	specifies a position shift
\zxhlp{SKEW}	Specifies a skew angle
\zxhlp{SLOT}	Specifies AIPS catalog slot number
\zxhlp{SMODEL}	Source model
\zxhlp{SMOOTH}	Specifies spectral smoothing
\zxhlp{SMOTYPE}	Specifies smoothing
\zxhlp{SNCORPRM}	Task-specific parameters for \zxhlp{SNCOR}.
\zxhlp{SNCUT}	Specifies minimum signal-to-noise ratio
\zxhlp{SNVER}	specifies the output solution table
\zxhlp{SOLCON}	Gain solution constraint factor
\zxhlp{SOLINT}	Solution interval
\zxhlp{SOLMIN}	Minimum number of solution sub-intervals in a solution
\zxhlp{SOLMODE}	Solution mode
\zxhlp{SOLSUB}	Solution sub-interval
\zxhlp{SOLTYP}	Solution type
\zxhlp{SORT}	Specified desired sort order
\zxhlp{SOUCODE}	Calibrator code for source, not calibrator, selection
\zxhlp{SOURCES}	A list of source names
\zxhlp{SPARM}	General string array adverb
\zxhlp{SPECIDX}	Spectral index used to correct calibrations
\zxhlp{SPECPARM}	Spectral index per polarization per source
\zxhlp{SPECTRAL}	Flag to indicate whether an operation is spectral or continuum
\zxhlp{SPECURVE}	Spectral index curvature used to correct calibrations
\zxhlp{STFACTOR}	scales star display or SDI CLEANING process
\zxhlp{STOKES}	Stokes parameter
\zxhlp{STORE}	Store current POPS environment
\zxhlp{STRA1}	General string adverb
\zxhlp{STRA2}	General string adverb
\zxhlp{STRA3}	General string adverb
\zxhlp{STRB1}	General string adverb
\zxhlp{STRB2}	General string adverb
\zxhlp{STRB3}	General string adverb

\zxhlp{STRC1}	General string adverb
\zxhlp{STRC2}	General string adverb
\zxhlp{STRC3}	General string adverb
\zxhlp{STVERS}	star display table version number
\zxhlp{SUBARRAY}	Subarray number
\zxhlp{SYMBOL}	General adverb, probably defines a plotting symbol type
\zxhlp{SYS2COM}	specifies a command to be sent to the operating system
\zxhlp{SYSCOM}	specifies a command to be sent to the operating system
\zxhlp{SYSOUT}	specifies the output device used by the system
\zxhlp{SYSVEL}	Systemic velocity
\zxhlp{TASK}	Name of a task
\zxhlp{TAU0}	Opacities by antenna number
\zxhlp{TBLC}	Gives the bottom left corner of an image to be displayed
\zxhlp{TCODE}	Determines which type of \zxhlp{DDT} is \zxhlp{RUN}.
\zxhlp{TDISK}	Determines where output \zxhlp{DDT} data is placed
\zxhlp{THEDATE}	contains the date and time in a string form
\zxhlp{TIMERANG}	Specifies a timerange
\zxhlp{TIMSMO}	Specified smoothing times
\zxhlp{TMASK}	Determines which tasks are executed when a \zxhlp{DDT} is \zxhlp{RUN}.
\zxhlp{TMODE}	Determines which input is used when a \zxhlp{DDT} is \zxhlp{RUN}.
\zxhlp{TNAMF}	Determines which files are input to \zxhlp{DDT}.
\zxhlp{TRANSCOD}	Specified desired transposition of an image
\zxhlp{TRC}	Specified the top right corner of a subimage
\zxhlp{TRECVR}	Receiver temperatures by polarization and antenna
\zxhlp{TRIANGLE}	specifies closure triangles to be selected/deselected
\zxhlp{TTRC}	Specifies the top right corner of a subimage to be displayed
\zxhlp{TVBUT}	Tells which AIPS TV button was pushed
\zxhlp{TVCHAN}	Specified a TV channel (plane)
\zxhlp{TVCORN}	Specified the TV pixel for the bottom left corner of an image
\zxhlp{TVLEVS}	Gives the peak intensity to be displayed in levels
\zxhlp{TVXY}	Pixel position on the TV screen
\zxhlp{TXINC}	TV X coordinate increment
\zxhlp{TYINC}	TV Y coordinate increment
\zxhlp{TYVER}	specifies the version of the system temperature table used
\zxhlp{TZINC}	TV Z coordinate increment
\zxhlp{USERID}	User number
\zxhlp{UVBOX}	radius of the smoothing box used for uniform weighting
\zxhlp{UVBXFN}	type of function used when counting for uniform weighting
\zxhlp{UVCOPPRM}	Parameter adverb array for task \zxhlp{UVCOP}
\zxhlp{UVFIXPRM}	Parameter adverb array for task \zxhlp{UVFIX}
\zxhlp{UVRANGE}	Specify range of projected baselines
\zxhlp{UVSIZE}	specifies number of pixels on X and Y axes of a UV image
\zxhlp{UVTAPER}	Widths in U and V of gaussian weighting taper function
\zxhlp{UVWTFN}	Specify weighting function, Uniform or Natural
\zxhlp{VCODE}	General string adverb
\zxhlp{VECTOR}	selects method of averaging UV data
\zxhlp{VELDEF}	Specifies velocity definition
\zxhlp{VELTYP}	Velocity frame of reference
\zxhlp{VERSION}	Specify AIPS version or local task area
\zxhlp{VLAMODE}	\zxhlp{VLA} observing mode
\zxhlp{VLAOBS}	Observing program or part of observer's name
\zxhlp{VLBINPRM}	Control parameters to read data from NRAO/MPI MkII correlators
\zxhlp{VNUMBER}	Specifies the task parameter (\zxhlp{VGET}/\zxhlp{VPUT}) save area
\zxhlp{VPARM}	General numeric array adverb used in modeling
\zxhlp{WEIGHTIT}	Controls modification of weights before gain/fringe solutions
\zxhlp{WGAUSS}	Widths of Gaussian models (FWHM)
\zxhlp{WTTHRESH}	defines the weight threshold for data acceptance
\zxhlp{WTUV}	Specifies the weight to use for UV data outside \zxhlp{UVRANGE}

```
\zxhlp{XAXIS}      Which parameter is plotted on the horizontal axis.
X           spare scalar adverb for use in procedures
\zxhlp{XINC}      increment associated with an array of numbers
\zxhlp{XPARM}     General adverb for up to 10 parameters, may refer to X coord
\zxhlp{XTYPE}     Specify type of process, often the X axis type of an image
\zxhlp{XYRATIO}   Ratio of X to Y units per pixel
Y           spare scalar adverb for use in procedures
\zxhlp{YINC}      Y axis increment
\zxhlp{YPARM}     Specifies Y axis convolving function
\zxhlp{YTYPE}     Y axis (V) convolving function type
\zxhlp{ZEROSP}   Specify how to include zero spacing fluxes in FT of UV data
\zxhlp{ZINC}      Set the increment of the third axis
\zxhlp{ZXRATIO}  Ratio between Z axis (pixel value) and X axis
```

13.2 ANALYSIS

```
\zxhlp{ACTNOISE}  puts estimate of actual image uncertainty and zero in header
\zxhlp{AFARS}    Is used after \zxhlp{FARS} to determine Position and Value of the maximum
\zxhlp{AGAUS}    Fits 1-dimensional Gaussians to absorption-line spectra
\zxhlp{AHIST}    Task to convert image intensities by adaptive histogram
\zxhlp{AVOPTION} Controls type or range of averaging done by a task
\zxhlp{BDEPO}    computes depolarization due to rotation measure gradients
\zxhlp{BLANK}    blanks out selected, e.g. non-signal, portions of an image
\zxhlp{BLSUM}    sums images over irregular sub-images, displays spectra
\zxhlp{BSCOR}    Combines two beam-switched images
\zxhlp{BSTST}    Graphical display of solutions to frequency-switched data
\zxhlp{BWSMEAR}  amount of bandwidth smearing correction to use
\zxhlp{CC2IM}    Make model image from a CC file
\zxhlp{COMB}     combines two images by a variety of mathematical methods
\zxhlp{CTYPE}    specifies type of component
\zxhlp{CUBIT}    Model a galaxy's density and velocity distribution from full cube
\zxhlp{DFTIM}    Makes image of DFT at arbitrary point showing time vs frequency
\zxhlp{DOALIGN}  specifies how two or more images are aligned in computations
\zxhlp{DOFARS}   Procedure to aid in Faraday rotation synthesis using the \zxhlp{FARS} task
\zxhlp{DOINVERS} selects opposite of normal function
\zxhlp{DOMAX}   selects solutions for maxima of models
\zxhlp{DOOUTPUT} selects whether output image or whatever is saved / discarded
\zxhlp{DOPOS}    selects solutions for positions of model components
\zxhlp{DOSPIX}  selects solutions for spectral index of model components
\zxhlp{DOWIDTH}  selects solution for widths of model components
\zxhlp{ECOUNT}   give the highest count or iteration for some process
\zxhlp{FARS}     Faraday rotation synthesys based on the brightness vs wavelength
\zxhlp{FLUX}    gives a total intensity value for image/component or to limit
\zxhlp{FMAX}    specifies peak values of model components - results of fits
\zxhlp{FPOS}    specifies pixel positions of fit model components
\zxhlp{FQUBE}   collects n-dimensional images into n+1-dimensional \zxhlp{FREQID} image
\zxhlp{FSHIFT}  specifies a position shift - output from fitting routines
\zxhlp{FWIDTH}  gives widths of model components - results of fitting
\zxhlp{GAL}     Determine parameters from a velocity field
\zxhlp{GMAX}    specifies peak values of model components
\zxhlp{GPOS}    specifies pixel positions of model components
\zxhlp{GRBLINK} Verb which blinks 2 TV graphics planes
\zxhlp{GWIDTH}  gives widths of model components
\zxhlp{HGEOM}   interpolates image to different gridding and/or geometry
HLPTVHLD  Interactive image display with histogram equalization - run-time help
HLPTVSAD  Find & fit Gaussians to an image with interaction - run-time help
HLPTVSPC  Interactive display of spectra from a cube - run-time help
```

13.2. ANALYSIS

13. Current AIPS Software

```

\zxhlp{HOLGR}          Read & process holography visibility data to telescope images
\zxhlp{HOLOG}          Read & process holography visibility data to telescope images
\zxhlp{IMCENTER}        returns pixel position of sub-image centroid
\zxhlp{IMDIST}          determines spherical distance between two pixels
\zxhlp{IMEAN}           displays the mean & extrema and plots histogram of an image
\zxhlp{IMERG}           merges images of different spatial resolutions
\zxhlp{IMFIT}           Fits Gaussians to portions of an image
\zxhlp{IMLIN}           Fits and removes continuum emission from cube
\zxhlp{IMMOD}           adds images of model objects to an image
\zxhlp{IMSTAT}          returns statistics of a sub-image
\zxhlp{IMVAL}           returns image intensity and coordinate at specified pixel
\zxhlp{IMVIM}           plots one image's values against another's
\zxhlp{IRING}           integrates intensity / flux in rings / ellipses
\zxhlp{JMFIT}           Fits Gaussians to portions of an image
\zxhlp{LAYER}            Task to create an RGB image from multiple images
\zxhlp{LGEOM}            regrids images with rotation, shift using interpolation
\zxhlp{MATHS}            operates on an image with a choice of mathematical functions
\zxhlp{MAXFIT}          returns pixel position and image intensity at a maximum
\zxhlp{MCUBE}            collects n-dimensional images into n+1-dimensional image
\zxhlp{MEDI}              combines four images by a variety of mathematical methods
\zxhlp{MFITSET}          gets adverbs for running \zxhlp{IMFIT} and \zxhlp{JMFIT}
\zxhlp{MFPRD}            prints MF tables in a format needed by modelling software
\zxhlp{MINPATCH}         specifies the minimum size allowed for the center of the beam
\zxhlp{MODAB}            Makes simple absorption/emission spectral-line image in I/V
\zxhlp{MODIM}            adds images of model objects to image cubes in IQU polarization
\zxhlp{MODSP}            adds images of model objects to image cubes in I/V polarization
\zxhlp{MOMFT}            calculates images of moments of a sub-image
\zxhlp{MOMNT}            calculates images of moments along x-axis (vel, freq, ch)
\zxhlp{MWFLT}            applies linear & non-linear filters to images
\zxhlp{NGAUSS}           Number of Gaussians to fit
\zxhlp{NINER}             Applies various 3x3 area operators to an image.
\zxhlp{NNLSQ}            Non-Negative-Least-Squares decomposition of spectrum
\zxhlp{OMFIT}            Fits sources and, optionally, a self-cal model to uv data
\zxhlp{OUTTEXT}          specifies name of output text file, not in regular catalog
\zxhlp{PANEL}             Convert \zxhlp{HOLGR} output to panel adjustment table
\zxhlp{PBCOR}             Task to apply or correct an image for a primary beam
\zxhlp{PRTIM}             prints image intensities from an MA catalog entry
\zxhlp{QIMVAL}            returns image intensity and coordinate at specified pixel
\zxhlp{QUXTR}             extracts text files from Q,U cubes for input to \zxhlp{TARS}
\zxhlp{RFARS}             Correct Q/U cubes for Faraday rotation synthesis results
\zxhlp{RM2PL}             Plots spectrum of a pixel with \zxhlp{RMFIT} fit
\zxhlp{RMFIT}             Fits 1-dimensional polarization spectrum to Q/U cube
\zxhlp{RM}                 Task to calculate rotation measure and magnetic field
\zxhlp{RMSD}               Calculate rms for each pixel using data at the box around the pixel
\zxhlp{RMSLIMIT}          selects things with RMS above this limit
\zxhlp{SAD}                 Finds and fits Gaussians to portions of an image
\zxhlp{SCLIM}              operates on an image with a choice of mathematical functions
\zxhlp{SERCH}              Finds line signals in transposed data cube
\zxhlp{SET1DG}             Verb to set 1D gaussian fitting initial guesses.
\zxhlp{SHADO}              Calculate the shadowing of antennas at the array
\zxhlp{SLCOL}              Task to collate slice data and models.
\zxhlp{SLFIT}              Task to fit gaussians to slice data.
\zxhlp{SLICE}              Task to make a slice file from an image
\zxhlp{SMOTH}              Task to smooth a subimage from upto a 7-dim. image
\zxhlp{SPCOR}              Task to correct an image for a primary beam and spectral index
\zxhlp{SPIXR}              Fits spectral indexes to each row of an image incl curvature
\zxhlp{SPMOD}              Modify UV database by adding a model with spectral lines
\zxhlp{STFUN}              Task to calculate a structure function image

```

```

\zxhlp{STVERS}      star display table version number
\zxhlp{SUMSQ}       Task to sum the squared pixel values of overlapping,
\zxhlp{TABGET}      returns table entry for specified row, column and subscript.
\zxhlp{TAPUT}       replaces table entry for specified row, column and subscript.
\zxhlp{TARPL}       Plot output of \zxhlp{TARS} task
\zxhlp{TARS}        Simulation of Faraday rotation synthesis (mainly task \zxhlp{FARS})
\zxhlp{TK1SET}      Verb to reset 1D gaussian fitting initial guess.
\zxhlp{TKAGUESS}    Verb to re-plot slice model guess directly on TEK
\zxhlp{TKAMODEL}    Verb to add slice model display directly on TEK
\zxhlp{TKASLICE}    Verb to add a slice display on TEK from slice file
\zxhlp{TKGUESS}     Verb to display slice model guess directly on TEK
\zxhlp{TKMODEL}     Verb to display slice model directly on TEK
\zxhlp{TKSET}       Verb to set 1D gaussian fitting initial guesses.
\zxhlp{TKSLICE}     Verb to display slice file directly on TEK
\zxhlp{TKVAL}       Verb to obtain value under cursor from a slice
\zxhlp{TKXY}        Verb to obtain pixel value under cursor
\zxhlp{TV1SET}      Verb to reset 1D gaussian fitting initial guess on TV plot.
\zxhlp{TVACOMPS}   Verb to add slice model components directly on TV graphics
\zxhlp{TVAGUESS}   Verb to re-plot slice model guess directly on TV graphics
\zxhlp{TVAMODEL}   Verb to add slice model display directly on TV graphics
\zxhlp{TVARESID}  Verb to add slice model residuals directly on TV graphics
\zxhlp{TVASLICE}   Verb to add a slice display on TV graphics from slice file
\zxhlp{TVBLINK}    Verb which blinks 2 TV planes, can do enhancement also
\zxhlp{TVCOMPS}    Verb to display slice model components directly on TV graphics
\zxhlp{TCUBE}       Verb to load a cube into tv channel(s) & run a movie
\zxhlp{TVDIST}     determines spherical distance between two pixels on TV screen
\zxhlp{TVGUESS}    Verb to display slice model guess directly on TV graphics
\zxhlp{TVHLD}      Task to load an image to the TV with histogram equalization
\zxhlp{TVMAXFIT}  displays fit pixel positions and intensity at maxima on TV
\zxhlp{TVMODEL}   Verb to display slice model directly on TV graphics
\zxhlp{TVRESID}   Verb to display slice model residuals directly on TV graphics
\zxhlp{TVSAD}      Finds and fits Gaussians to portions of an image with interaction
\zxhlp{TVSET}      Verb to set slice Gaussian fitting initial guesses from TV plot
\zxhlp{TVSLICE}   Verb to display slice file directly on TV
\zxhlp{TVSPC}     Display images and spectra from a cube
\zxhlp{UVADC}     Fourier transforms and corrects a model and adds to uv data.
\zxhlp{UVCON}     Generate sample UV coverage given a user defined array layout
\zxhlp{UVFIT}     Fits source models to uv data.
\zxhlp{UVGIT}     Fits source models to uv data.
\zxhlp{UVHIM}     Makes image of the histogram on two user-chosen axes
\zxhlp{UVMOD}     Modify UV database by adding a model incl spectral index
\zxhlp{UVSEN}     Determine RMS sidelobe level and brightness sensitivity
\zxhlp{UVSIM}     Generate sample UV coverage given a user defined array layout
\zxhlp{WARP}       Model warps in Galaxies
\zxhlp{XBASL}     Fits and subtracts nth-order baselines from cube (x axis)
\zxhlp{XG2PL}     Plots spectrum of a pixel with \zxhlp{XGAUS}/\zxhlp{AGAUS} and \zxhlp{ZEMAN}/\zxhlp{ZAMAN} fits
\zxhlp{XGAUS}     Fits 1-dimensional Gaussians to images: restartable
\zxhlp{XMOM}      Fits one-dimensional moments to each row of an image
\zxhlp{ZAMAN}     Fits 1-dimensional Zeeman model to absorption-line data
\zxhlp{ZEMAN}     Fits 1-dimensional Zeeman model to data

```

13.3 AP

```

\zxhlp{APCLN} Deconvolves images with CLEAN algorithm
\zxhlp{APGS} deconvolves image with Gerchberg-Saxton algorithm
\zxhlp{APVC} Deconvolves images with van Cittert algorithm
\zxhlp{BLING} find residual rate and delay on individual baselines
\zxhlp{BPASS} computes spectral bandpass correction table
\zxhlp{BSGRD} Task to image beam-switched single-dish data
\zxhlp{CALIB} determines antenna calibration: complex gain
\zxhlp{CCRES} Removes or restores a CC file to a map with a gaussian beam.
\zxhlp{COMAP} Procedure to MAP and Self-Calibrate a UVDATA set
COMAP_NA Procedure to MAP and Self-Calibrate a UVDATA set
COMAP_UV Procedure to MAP and Self-Calibrate a UVDATA set
\zxhlp{CONPL} Plots AIPS gridding convolution functions
\zxhlp{CONVL} convolves an image with a gaussian or another image
\zxhlp{CPASS} computes polynomial spectral bandpass correction table
\zxhlp{EDITA} Interactive TV task to edit uv data based on TY/SY/SN/CL tables
\zxhlp{EDITR} Interactive baseline-oriented visibility editor using the TV
\zxhlp{FFT} takes Fourier Transform of an image or images
\zxhlp{FRCAL} Faraday rotation self calibration task
\zxhlp{FRING} fringe fit data to determine antenna calibration, delay, rate
\zxhlp{GUARD} portion of UV plane to receive no data in gridding
HLPCLEAN Cleaning tasks - run-time help
HLPSCIMG Full-featured image plus self-cal loops, editing - run-time help
HLPSCMAP Imaging plus self-cal and editing \zxhlp{SCMAP} - run-time help
\zxhlp{HYB} \zxhlp{RUN} to set parameters for HYBRID (\zxhlp{CALIB}/\zxhlp{MX}) self-cal imaging
\zxhlp{IM2UV} converts an image to a visibility data set
\zxhlp{IMAGR} Wide-field and/or wide-frequency Cleaning / imaging task.
\zxhlp{IMFRING} large image delay fitting with \zxhlp{IM2CC} and \zxhlp{OOFRING}
\zxhlp{IMSCAL} large image self-cal with \zxhlp{IM2CC} and \zxhlp{OOCAL}
\zxhlp{KRING} fringe fit data to determine antenna calibration, delay, rate
\zxhlp{MAXPIXEL} maximum pixels searched for components in Clark CLEAN
\zxhlp{NOBAT} Task to lock lower priority users out of the AP
\zxhlp{OOCAL} determines antenna complex gain with frequency-dependent models
\zxhlp{OOFRING} fringe fit data to determine antenna calibration, delay, rate
\zxhlp{RLDLY} fringe fit data to determine antenna R-L delay difference
\zxhlp{RSTOR} Restores a CC file to a map with a gaussian beam.
\zxhlp{SCIMG} Full-featured imaging plus self-calibration loop with editing
\zxhlp{SCMAP} Imaging plus self-calibration loop with editing
\zxhlp{SDGRD} Task to select and image random-position single-dish data
\zxhlp{SDIMG} Task to select and image random-position single-dish data
\zxhlp{SNEDT} Interactive SN/CL/TY/SY table editor using the TV
\zxhlp{TDEPEND} Time-dependent imaging procedure sequence
TD_SCANS Time-dependent imaging procedure sequence: find intervals
TD_SSCAN Time-dependent imaging procedure sequence: find intervals
TD_STEP3 Time-dependent imaging procedure "step 3"
TD_STEP5 Time-dependent imaging procedure sequence: later steps
\zxhlp{UVADC} Fourier transforms and corrects a model and adds to uv data.
\zxhlp{UVMAP} makes images from calibrated UV data.
\zxhlp{VLABPSS} computes spectral bandpass correction table
\zxhlp{WFCLN} Wide field and/or widefrequency CLEANing/imaging task.

```

13.4 ASTROMET

```
\zxhlp{ASTROMET} Describes the process of astrometric/geodetic reduction in AIPS
\zxhlp{FRMAP} Task to build a map using fringe rate spectra
\zxhlp{HF2SV} convert HF tables from \zxhlp{FRINGE}/\zxhlp{MBDLY} to form used by Calc/Solve
\zxhlp{HFPRT} write HF tables from \zxhlp{CL2HF}
\zxhlp{XTRAN} Create an image with transformed coordinates
```

13.5 BATCH

Type: Operations to prepare, submit, and monitor batch jobs

Use: There are two batch streams of AIPS, each capable of processing a queue of jobs. To run a batch job, one must first prepare the text of the job in a work file. This text may contain any normal AIPS/POPS statement including \zxhlp{RUN}, except for verbs and tasks related to batch preparation, the TV, the TEK4012 green screen, and the tape drives. When the text is ready, it may be submitted to the batch AIPS. On the way, it is tested for errors and is submitted only if none are found. After successful submission, the work file and any \zxhlp{RUN} files involved may be altered without affecting the job. Array processor tasks are allowed only in queue #2 and only at night. They may be submitted at any time, however. Line printer output should be directed to a user chosen file (via adverb \zxhlp{OUTPRINT}). If \zxhlp{OUTPRINT} = ' ', all tasks and AIPS itself will write to a file named PRTFIL:BATCHjjj.nnn, where jjj is the job number in hex and nnn is the user number in hex. Note that all print jobs are concatenated into the specified file(s).

Adverbs:

```
\zxhlp{BATQUE} Number of queue to be used ( 1 or 2 or more )
\zxhlp{JOBNUM} Job number involved (101 - 164, 201 -264, ...)
\zxhlp{BATFLINE} First line number to be editted or listed
\zxhlp{BATNLINE} Number of lines to be listed
```

Verbs:

```
\zxhlp{BATCH} Add text to \zxhlp{BATQUE} work file
\zxhlp{BATCLEAR} Initiate and clear \zxhlp{BATQUE} work file
\zxhlp{BATLIST} List \zxhlp{BATNLINE} starting with \zxhlp{BATFLINE} from \zxhlp{BATQUE} work file
\zxhlp{BATEDIT} Edit text in \zxhlp{BATQUE} work file starting with line \zxhlp{BATFLINE} (or immediate argument)
\zxhlp{BAMODIFY} Edit text in \zxhlp{BATQUE} work file in line \zxhlp{BATFLINE} (or immediate argument), character-mode editing.
\zxhlp{SUBMIT} Submit text in \zxhlp{BATQUE} work file as job for queue \zxhlp{BATQUE}
\zxhlp{JOBLIST} List \zxhlp{BATNLINE} starting with \zxhlp{BATFLINE} from text file of job \zxhlp{JOBNUM}
\zxhlp{QUEUEUS} List jobs submitted, running, and completed in queue \zxhlp{BATQUE}
\zxhlp{UNQUEUE} Remove \zxhlp{JOBNUM} from queue, copy text of job to work file \zxhlp{BATQUE}
```

Batch jobs may also be prepared and submitted outside of AIPS, using the program \zxhlp{BATER}. See \zxhlp{HELP} \zxhlp{BATER}.

```
\zxhlp{AIPSB} AIPS main program for executing batch jobs
\zxhlp{AIPSC} AIPS main program for testing and queuing batch jobs
```

\zxhlp{BAMODIFY}	edits characters in a line of a batch work file
\zxhlp{BATCH}	starts entry of commands into batch-job work file
\zxhlp{BATCLEAR}	removes all text from a batch work file
\zxhlp{BATEDIT}	starts an edit (replace, insert) session on a batch work file
\zxhlp{BATER}	stand-alone program to prepare and submit batch jobs
\zxhlp{BATFLINE}	specifies starting line in a batch work file
\zxhlp{BATLIST}	lists the contents of a batch work file
\zxhlp{BATNLINE}	specifies the number of lines to process in a batch work file
\zxhlp{BATQUE}	specifies the desired batch queue
\zxhlp{ENDBATCH}	terminates input to batch work file
\zxhlp{JOBLIST}	lists contents of a submitted and pending batch job
\zxhlp{JOBNUM}	specifies the batch job number
\zxhlp{QUEUES}	Verb to list all submitted jobs in the job queue
\zxhlp{REMQUE}	specifies the desired batch queue on a remote computer
\zxhlp{UNQUE}	remove a given job from the job queue

13.6 CALIBRAT

For a lengthy description of the calibration of interferometric data (\zxhlp{VLA} and VLB line and continuum) enter:

```
\zxhlp{HELP} \zxhlp{CALIBRAT}
```

```
*****
```

\zxhlp{ACCOR}	Corrects cross amplitudes using auto correlation measurements
\zxhlp{ACFIT}	Determine antenna gains from autocorrelations
\zxhlp{ACLIP}	edits suto-corr data for amplitudes, phases, and weights out of range
\zxhlp{ACSCL}	Corrects cross amplitudes using auto correlation measurements
\zxhlp{ANCAL}	Places antenna-based Tsyst and gain corrections in CL table
\zxhlp{ANCHECK}	Checks By sign in Antenna files
\zxhlp{ANTAB}	Read amplitude calibration information into AIPS
\zxhlp{ANTENNAS}	Antennas to include/exclude from the task or verb
\zxhlp{ANTUSE}	Antennas to include/exclude from the task or verb
\zxhlp{ANTWT}	Antenna Weights for UV data correction in Calibration
\zxhlp{APCAL}	Apply TY and GC tables to generate an SN table
\zxhlp{APGPS}	Apply GPS-derived ionospheric corrections
\zxhlp{ATMCA}	Determines delay/phase gradient from calibrator observations
\zxhlp{BASELINE}	specifies which antenna pairs are to be selected/deselected
\zxhlp{BASFIT}	fits antenna locations from SN-table data
\zxhlp{BDAPL}	Applies a BD table to another data set
\zxhlp{BLAPP}	applies baseline-based fringe solutions a la \zxhlp{BLAPP}
\zxhlp{BLAVG}	Average cross-polarized UV data over baselines.
\zxhlp{BLCAL}	Compute closure offset corrections
\zxhlp{BLCHN}	Compute closure offset corrections on a channel-by-channel basis
\zxhlp{BLING}	find residual rate and delay on individual baselines
\zxhlp{BLVER}	specifies the version of the baseline-calibration table used
\zxhlp{BPASS}	computes spectral bandpass correction table
\zxhlp{BPASSPRM}	Control adverb array for bandpass calibration
\zxhlp{BPCOR}	Correct BP table.
\zxhlp{BPEDT}	Interactive TV task to edit uv data based on BP tables
\zxhlp{BPERR}	Print and plot \zxhlp{BPASS} closure outputs
\zxhlp{BPLOT}	Plots bandpass tables in 2 dimensions as function of time
\zxhlp{BPSMO}	Smooths or interpolates bandpass tables to regular times
\zxhlp{BPVER}	specifies the version of the bandpass table to be applied
\zxhlp{BPWAY}	Determines channel-dependent relative weights
\zxhlp{BPWGT}	Calibrates data and scales weights by bandpass correction
\zxhlp{BSPRT}	print BS tables
\zxhlp{BSROT}	modifies SD beam-switch continuum data for error in throw

\zxhlp{CALCODE}	specifies the type of calibrator to be selected
\zxhlp{CALDIR}	lists calibrator source models available as AIPS FITS files
\zxhlp{CALIB}	determines antenna calibration: complex gain
\zxhlp{CALBRAT}	describes the process of data calibration in AIPS
\zxhlp{CALIN}	specifies name of input disk file usually with calibration data
\zxhlp{CALRD}	Reads calibrator source model-image FITS file
\zxhlp{CALSOR}	specifies source names to be included in calibration
\zxhlp{CC2IM}	Make model image from a CC file
\zxhlp{CENTR}	modifies UV data to center the reference channel
\zxhlp{CHANSEL}	Array of start, stop, increment channel numbers to average
\zxhlp{CLCAL}	merges and smooths SN tables, applies them to CL tables
\zxhlp{CLCOP}	copy CL/SN file calibration between polarizations or IFs
\zxhlp{CLCOR}	applies user-selected corrections to the calibration CL table
\zxhlp{CLCORPRM}	Parameter adverb array for task \zxhlp{CLCOR}
\zxhlp{CLINT}	CL table entry interval
\zxhlp{CLINV}	copy CL/SN file inverting the calibration
\zxhlp{CLIP}	edits data based on amplitudes, phases, and weights out of range
\zxhlp{CLSMO}	smooths a calibration CL table
\zxhlp{CLVLB}	Corrects CL table gains for pointing offsets in VLBI data
\zxhlp{CMETHOD}	specifies the method by which the uv model is computed
\zxhlp{CMODEL}	specifies the method by which the uv model is computed
\zxhlp{CONF1}	Optimize array configuration by minimum side lobes
\zxhlp{CPASS}	computes polynomial spectral bandpass correction table
\zxhlp{CSCOR}	applies specified corrections to CS tables
\zxhlp{CVEL}	shifts spectral-line UV data to a given velocity
\zxhlp{DCHANSEL}	Array of start, stop, increment channel #S + IF to avoid
\zxhlp{DECOR}	Measures the decorrelation between channels and IF of uv data
\zxhlp{DEFLG}	edits data based on decorrelation over channels and time
\zxhlp{DELCORR}	specifies whether VLBA delay corrections are to be used
\zxhlp{DELZN}	Determines residual atmosphere depth at zenith and clock errors
\zxhlp{DFCOR}	applies user-selected corrections to the calibration CL table
\zxhlp{DIGICOR}	specifies whether VLBA digital corrections are to be applied
\zxhlp{DOACOR}	specifies whether autocorrelation data are included
\zxhlp{DOAPPLY}	Flag to indicate whether an operation is applied to the data
\zxhlp{DOBAND}	specifies if/how bandpass calibration is applied
\zxhlp{DOBWEEN}	Controls smoothing between sources in calibration tables
\zxhlp{DOCALIB}	specifies whether a gain table is to be applied or not
\zxhlp{DODELAY}	selects solution for phase/amplitude or delay rate/phase
\zxhlp{DOFIT}	Controls which antennas are fit by what methods
\zxhlp{DOFLAG}	Controls closure cutoff in gain solutions and flagging
\zxhlp{DOOSRO}	calibrating amplitude and phase, and imaging \zxhlp{VLA} data
\zxhlp{DOPOL}	selects application of any polarization calibration
\zxhlp{DOVLAMP}	Produces amp calibration file for phased-\zxhlp{VLA} VLBI data
\zxhlp{DTSIM}	Generate fake UV data
\zxhlp{EDITA}	Interactive TV task to edit uv data based on TY/SY/SN/CL tables
\zxhlp{EDITR}	Interactive baseline-oriented visibility editor using the TV
\zxhlp{ELFIT}	Plots/fits selected contents of SN, TY, SY, PC or CL files
\zxhlp{ELINT}	Determines and removes gain dependence on elevation
\zxhlp{EVASN}	Evaluates statistics in SN/CL tables
\zxhlp{EVAUV}	Subtracts & divides a model into UV data, does statistics on results
\zxhlp{FACES}	makes images of catalog sources for initial calibration
\zxhlp{FARAD}	add ionospheric Faraday rotation to CL table
\zxhlp{FGPLT}	Plots selected contents of FG table
\zxhlp{FGTAB}	Plots selected contents of FG table
\zxhlp{FINDR}	Find normal values for a uv data set
\zxhlp{FIXWT}	Modify weights to reflect amplitude scatter of data
\zxhlp{FLAGR}	Edit data based on internal RMS, amplitudes, weights
\zxhlp{FLAGVER}	selects version of the flagging table to be applied

```

\zxhlp{FLGIT}           flags data based on the rms of the spectrum
\zxhlp{FQTOL}           Frequency tolerance with which FQ entries are accepted.
\zxhlp{FRCAL}           Faraday rotation self calibration task
\zxhlp{FREQID}          Frequency Identifier for frequency, bandwidth combination
\zxhlp{FRING}            fringe fit data to determine antenna calibration, delay, rate
\zxhlp{FTFLG}            interactive flagging of UV data in channel-time using the TV
\zxhlp{GAINERR}          gives estimate of gain uncertainty for each antenna
\zxhlp{GAINUSE}          specifies output gain table or gain table applied to data
\zxhlp{GAINVER}          specifies the input gain table
\zxhlp{GCPLT}            Plots gain curves from text files
\zxhlp{GCVER}            specifies the version of the gain curve table used
\zxhlp{GETJY}             determines calibrator flux densities
\zxhlp{GPSDL}            Calculate ionospheric delay and Faraday rotation corrections

HLPCLEAN   Cleaning tasks - run-time help
HLPEDIBP   Interactive BP table uv-data editor \zxhlp{BPEDT} - run-time help
HLPEDICL   Interactive SN/CL table uv-data editor - run-time help
HLPEDIPC   Interactive PC table editor \zxhlp{PCEDT} - run-time help
HLPEDISN   Interactive SN/CL table (not UV) editor - run-time help
HLPEDISS   Interactive SY table (not UV) editor - run-time help
HLPEDISY   Interactive SY table uv-data editor \zxhlp{EDITA} - run-time help
HLPEDITS   Interactive TY table (not UV) editor - run-time help
HLPEDITTY   Interactive TY table uv-data editor \zxhlp{EDITA} - run-time help
HLPEDIUV   Interactive uv-data editor \zxhlp{EDITR} - run-time help
HLPFTFLG   Interactive time-channel visibility Editor - run-time help
HLPIBLED   Interactive Baseline based visibility Editor - run-time help
HLPPCFLG   Interactive time-channel PC table Editor \zxhlp{PCFLG} - run-time help
HLPSCIMG   Full-featured image plus self-cal loops, editing - run-time help
HLPSCMAP   Imaging plus self-cal and editing \zxhlp{SCMAP} - run-time help
HLPSPFLG   Interactive time-channel visibility Editor \zxhlp{SPFLG} - run-time help
HLPTVFLG   Interactive time-baseline visibility Editor \zxhlp{TVFLG} - run-time help

\zxhlp{HYB}           \zxhlp{RUN} to set parameters for HYBRID (\zxhlp{CALIB}/\zxhlp{MX}) self-cal imaging
\zxhlp{IBLED}          Interactive BaseLine based visibility EDitor
\zxhlp{ICHANSEL}       Array of start, stop, increment channel #S + IF to average
\zxhlp{IM2CC}          Task to convert an image to multi-facet Clean Components
\zxhlp{IMAGR}           Wide-field and/or wide-frequency Cleaning / imaging task.
\zxhlp{IMFRING}         large image delay fitting with \zxhlp{IM2CC} and \zxhlp{OOFRING}
\zxhlp{IMSCAL}          large image self-cal with \zxhlp{IM2CC} and \zxhlp{OOCAL}
\zxhlp{INDXH}           writes index file describing contents of UV data base
\zxhlp{INDXR}           writes index file describing contents of UV data base
\zxhlp{INTERPOL}        specifies the type of averaging done on the complex gains
\zxhlp{INTPARM}         specifies the parameters of the frequency interpolation function
\zxhlp{KRING}           fringe fit data to determine antenna calibration, delay, rate
\zxhlp{LDGPS}            load GPS data from an ASCII file
\zxhlp{LISTR}            prints contents of UV data sets and assoc. calibration tables
\zxhlp{LOCIT}            fits antenna locations from SN-table data
\zxhlp{LPCAL}            Determines instrumental polarization for UV data
\zxhlp{MAPBM}            Map \zxhlp{VLA} beam polarization
\zxhlp{MBDLY}            Fits multiband delays from IF phases, updates SN table
\zxhlp{MINAMPER}         specifies the minimum amplitude error prior to some action
\zxhlp{MINANTEN}         states minimum number of antennas for a solution
\zxhlp{MINPHSER}         specifies the minimum phase error prior to some action
\zxhlp{MSORT}             Sort a UV dataset into a specified order
\zxhlp{MULTI}             Task to convert single-source to multi-source UV data
\zxhlp{NORMALIZ}         specifies the type of gain normalization if any
\zxhlp{OMFIT}             Fits sources and, optionally, a self-cal model to uv data
\zxhlp{ONEFREQ}           states that the current CC model was made with one frequency
\zxhlp{OOCAL}             determines antenna complex gain with frequency-dependent models
\zxhlp{OOFRING}           fringe fit data to determine antenna calibration, delay, rate

```

```

\zxhlp{OOSRT} Sort a UV dataset into a specified order
\zxhlp{OOSUB} Subtracts/divides a model from/into a uv data base
\zxhlp{PBEAM} Fits the analytic function to the measured values of the beam
\zxhlp{PCAL} Determines instrumental polarization for UV data
\zxhlp{PCASS} Finds amplitude bandpass shape from pulse-cal table data
\zxhlp{PCCOR} Corrects phases using \zxhlp{PCAL} tones data from PC table
\zxhlp{PCEDT} Interactive TV task to edit PC tables
\zxhlp{PCFIT} Finds delays and phases using a PC table
\zxhlp{PCFLG} interactive flagging of Pulse-cal data in channel-TB using the TV
\zxhlp{PCLOD} Reads ascii file containing pulse-cal info to PC table.
\zxhlp{PCPLT} Plots pulse-cal tables in 2 dimensions as function of time
\zxhlp{PCVEL} shifts spectral-line UV data to a given velocity: planet version
\zxhlp{PDVER} specifies the version of the spetral polarization table to use
\zxhlp{PEELR} calibrates interfering sources in multi-facet imges
\zxhlp{PHASPRM} Phase data array, by antenna number.
\zxhlp{PIPEAIPS} calibrating amplitude and phase, and imaging \zxhlp{VLA} data
\zxhlp{POLANGLE} Intrinsic polarization angles for up to 30 sources
\zxhlp{POLSN} Make a SN table from cross polarized fringe fit
\zxhlp{PRTSY} Task to print statistics from the SY table
\zxhlp{QUFIX} determines Right minus Left phase difference, corrects cal files
\zxhlp{QUOUT} writes text file of Q, U versus frequency to be used by \zxhlp{RLDIF}
\zxhlp{REAMP} modifies UV data re-scaling the amplitudes
\zxhlp{REFANT} Reference antenna
\zxhlp{RESEQ} Renumber antennas
\zxhlp{REWGT} modifies UV data re-scaling the weights only
\zxhlp{RFI} Look for \zxhlp{RFI} in uv data
\zxhlp{RLCAL} Determines instrumental right-left phase versus time (a self-cal)
\zxhlp{RLDIF} determines Right minus Left phase difference, corrects cal files
\zxhlp{RLDLY} fringe fit data to determine antenna R-L delay difference
\zxhlp{SCIMG} Full-featured imaging plus self-calibration loop with editing
\zxhlp{SCMAP} Imaging plus self-calibration loop with editing
\zxhlp{SDCAL} Task to apply single dish calibration
\zxhlp{SDVEL} shifts spectral-line single-dish data to a given velocity
\zxhlp{SEARCH} Ordered list of antennas for fring searches
\zxhlp{SELBAND} Specified bandwidth
\zxhlp{SELFREQ} Specified frequency
\zxhlp{SETJY} Task to enter source info into source (SU) table.
\zxhlp{SHOUV} displays uv data in various ways.
\zxhlp{SMODEL} Source model
\zxhlp{SMOTYPE} Specifies smoothing
\zxhlp{SNCOR} applies user-selected corrections to the calibration SN table
\zxhlp{SNCORPRM} Task-specific parameters for \zxhlp{SNCOR}.
\zxhlp{SNDUP} copies and duplicates SN table from single pol file to dual pol
\zxhlp{SNEDT} Interactive SN/CL/TY/SY table editor using the TV
\zxhlp{SNFIT} Fits parabola to SN amplitudes and plots result
\zxhlp{SNFLG} Writes flagging info based on the contents of SN files
\zxhlp{SNIFS} Plots selected contents of SN, TY, SY, PC or CL files
\zxhlp{SNP2D} Task to convert SN table single-channel phase to delay
\zxhlp{SNPLT} Plots selected contents of SN, SY, TY, PC or CL files
\zxhlp{SNREF} Chooses best reference antenna to minimize R-L differences
\zxhlp{SNSMO} smooths and filters a calibration SN table
\zxhlp{SNVER} specifies the output solution table
\zxhlp{SOLCL} adjust gains for solar data according to nominal sensitivity
\zxhlp{SOLCON} Gain solution constraint factor
\zxhlp{SOLINT} Solution interval
\zxhlp{SOLMIN} Minimum number of solution sub-intervals in a solution
\zxhlp{SOLMODE} Solution mode
\zxhlp{SOLSUB} Solution sub-interval

```

13.6. CALIBRAT

13. Current AIPS Software

```

\zxhlp{SOLTYPE} Solution type
\zxhlp{SOUCODE} Calibrator code for source, not calibrator, selection
\zxhlp{SOUSP} fits source spectral index from SU table or adverbs
\zxhlp{SPCAL} Determines instrumental polzn. for spec. line UV data
\zxhlp{SPECINDX} Spectral index used to correct calibrations
\zxhlp{SPECPARM} Spectral index per polarization per source
\zxhlp{SPECTRAL} Flag to indicate whether an operation is spectral or continuum
\zxhlp{SPECURVE} Spectral index survature used to correct calibrations
\zxhlp{SPFLG} interactive flagging of UV data in channel-TB using the TV
\zxhlp{SPLAT} Applies calibration and splits or assemble selected sources.
\zxhlp{SPLIT} converts multi-source to single-source UV files w calibration
\zxhlp{SY2TY} Task to generate a TY extension file from an \zxhlp{EVLA} SY table
\zxhlp{SYSOL} undoes and re-does nominal sensitivity application for Solar data
\zxhlp{TASAV} Task to copy all extension tables to a dummy uv or map file
\zxhlp{TAU0} Opacities by antenna number
\zxhlp{TDEPEND} Time-dependent imaging procedure sequence
\zxhlp{TECOR} Calculate ionospheric delay and Faraday rotation corrections
\zxhlp{TIMSMO} Specified smoothing times
\zxhlp{TRECVR} Receiver temperatures by polarization and antenna
\zxhlp{TRIANGLE} specifies closure triangles to be selected/deselected
\zxhlp{TRUEP} determines true antenna polarization from special data sets
\zxhlp{TVFLG} interactive flagging of UV data using the TV
\zxhlp{TYAPL} undoes and re-does nominal sensitivity application
\zxhlp{TYCOP} copy TY or SY table calibration between IFs
\zxhlp{TYSMO} smooths and filters a calibration TY or SY table
\zxhlp{TYVER} specifies the version of the system temperature table used
\zxhlp{UNCAL} sets up tables for uncalibrating Australia Telescope data
\zxhlp{USUBA} Assign subarrays within a uv-data file
\zxhlp{UVCRS} Finds the crossing points of UV-ellipses.
\zxhlp{UVFIT} Fits source models to uv data.
\zxhlp{UVFLG} Flags UV-data
\zxhlp{UVGIT} Fits source models to uv data.
\zxhlp{UVHOL} prints holography data from a UV data base with calibration
\zxhlp{UVMLN} edits data based on the rms of the spectrum
\zxhlp{UVPRT} prints data from a UV data base with calibration
\zxhlp{UVSRT} Sort a UV dataset into a specified order
\zxhlp{UVSUB} Subtracts/divides a model from/into a uv data base
\zxhlp{VBCAL} Scale visibility amplitudes by antenna based constants
\zxhlp{VLABP} \zxhlp{VLA} antenna beam polarization correction for snapshot images
\zxhlp{VLACALIB} Runs \zxhlp{CALIB} and \zxhlp{LISTR} for \zxhlp{VLA} observation
\zxhlp{VLACLCAL} Runs \zxhlp{CLCAL} and prints the results with \zxhlp{LISTR}
\zxhlp{VLALIST} Runs \zxhlp{LISTR} for \zxhlp{VLA} observation
\zxhlp{VLAMODE} \zxhlp{VLA} observing mode
\zxhlp{VLAMP} Makes \zxhlp{ANTAB} file for phased \zxhlp{VLA} used in VLBI observations
\zxhlp{VLANT} applies \zxhlp{VLA}/\zxhlp{EVLA} antenna position corrections from OPs files
\zxhlp{VLAOBS} Observing program or part of observer's name
\zxhlp{VLAPROCS} Procedures to simplify the reduction of VLBA data
\zxhlp{VLARESET} Reset calibration tables to a virginal state
\zxhlp{VLARUN} calibrating amplitude and phase, and imaging \zxhlp{VLA} data
\zxhlp{VLBAAMP} applies a-priori amplitude corrections to VLBA data
\zxhlp{VLBAPSS} computes spectral bandpass correction table
\zxhlp{VLBACALA} applies a-priori amplitude corrections to VLBA data
\zxhlp{VLBACCOR} applies a-priori amplitude corrections to VLBA data
\zxhlp{VLBACPOL} Procedure to calibrate cross-polarization delays
\zxhlp{VLBAEOPS} Corrects Earth orientation parameters
\zxhlp{VLBAFQS} Copies different FQIDS to separate files
\zxhlp{VLBAFRGP} Fringe fit phase referenced data and apply calibration
\zxhlp{VLBAFRNG} Fringe fit data and apply calibration

```

\zxhlp{VLBAKRGF}	Fringe fit phase referenced data and apply calibration
\zxhlp{VLBAKRNG}	Fringe fit data and apply calibration
\zxhlp{VLBAMCAL}	Merges redundant calibration data
\zxhlp{VLBAMPCL}	Calculates and applies manual instrumental phase calibration
\zxhlp{VLBAPANG}	Corrects for parallactic angle
\zxhlp{VLBAPCOR}	Calculates and applies instrumental phase calibration
\zxhlp{VLBAPIPE}	applies amplitude and phase calibration procs to VLBA data
\zxhlp{VLBARUN}	applies amplitude and phase calibration procs to VLBA data
\zxhlp{VLBATECR}	Calculate ionospheric delay and Faraday rotation corrections
\zxhlp{VLBAUTIL}	Procedures to simplify the reduction of VLBA data
\zxhlp{VLOG}	Pre-process external VLBA calibration files
\zxhlp{WEIGHTIT}	Controls modification of weights before gain/fringe solutions
\zxhlp{WETHR}	Plots selected contents of WX tables, flags data based on WX
\zxhlp{WRTPROCS}	Procedures to simplify the reduction of VLBA data
\zxhlp{WTTHRESH}	defines the weight threshold for data acceptance
\zxhlp{WTUV}	Specifies the weight to use for UV data outside \zxhlp{UVRANGE}

13.7 CATALOG

\zxhlp{ABACKUP}	VMS procedure to back up data on tape
\zxhlp{ACTNOISE}	puts estimate of actual image uncertainty and zero in header
\zxhlp{ADDBEAM}	Inserts clean beam parameters in image header
\zxhlp{ADDDISK}	makes a computer's disks available to the current AIPS session
\zxhlp{ALLDEST}	Delete a group or all of a users data files
\zxhlp{ALTDEF}	Sets frequency vs velocity relationship into image header
\zxhlp{ALTSWTCH}	Switches between frequency and velocity in image header
\zxhlp{ARESTORE}	Restores back up tapes of users data
\zxhlp{AX2REF}	Second reference pixel number
\zxhlp{AXDEFINE}	Define or modify an image axis description
\zxhlp{AXINC}	Axis increment - change in coordinate between pixels
\zxhlp{AXREF}	Reference pixel number
\zxhlp{AXTYPE}	Type of coordinate axis
\zxhlp{AXVAL}	Value of axis coordinate at reference pixel
\zxhlp{BAKLD}	reads all files of a catalog entry from \zxhlp{BAKTP} tape
\zxhlp{BAKTP}	writes all files of a catalog entry to tape in host format
\zxhlp{CATALOG}	list one or more entries in the user's data directory
\zxhlp{CATNO}	Specifies AIPS catalog slot number range
\zxhlp{CELGAL}	switches header between celestial and galactic coordinates
\zxhlp{CHKNAME}	Checks for existence of the specified image name
\zxhlp{CLR2NAME}	clears adverbs specifying the second input image
\zxhlp{CLR3NAME}	clears adverbs specifying the third input image
\zxhlp{CLR4NAME}	clears adverbs specifying the fourth input image
\zxhlp{CLR5NAME}	clears adverbs specifying the fourth input image
\zxhlp{CLRNAME}	clears adverbs specifying the first input image
\zxhlp{CLRONAME}	clears adverbs specifying the first output image
\zxhlp{CLRSTAT}	remove any read or write status flags on a directory entry
\zxhlp{COODEFIN}	Define or modify an image axis coordinate description
\zxhlp{COOINC}	Celestial axes increment: change in coordinate between pixels
\zxhlp{COOREF}	Reference pixel number for two coordinate axes
\zxhlp{COOTYPE}	Celestial axes projection type
\zxhlp{DAYNUMBR}	finds day nuumber of an image or uv data set
\zxhlp{DISKU}	shows disk use by one or all users
\zxhlp{DOALPHA}	specifies whether some list is alphabetized
\zxhlp{DOCAT}	specifies whether the output is saved (cataloged) or not
\zxhlp{DOOUTPUT}	selects whether output image or whatever is saved / discarded
\zxhlp{EGETHEAD}	returns parameter value from image header and error code
\zxhlp{EGETNAME}	fills in input name adverbs by catalog slot number, w error

13.7. CATALOG

13. Current AIPS Software

\zxhlp{EPOSWTCH}	Switches between B1950 and J2000 coordinates in header
\zxhlp{ERROR}	was there an error
\zxhlp{EXTDEST}	deletes one or more extension files
\zxhlp{EXTLIST}	lists detailed information about contents of extension files
\zxhlp{GET2NAME}	fills 2nd input image name parameters by catalog slot number
\zxhlp{GET3NAME}	fills 3rd input image name parameters by catalog slot number
\zxhlp{GET4NAME}	fills 4th input image name parameters by catalog slot number
\zxhlp{GET5NAME}	fills 5th input image name parameters by catalog slot number
\zxhlp{GETHEAD}	returns parameter value from image header
\zxhlp{GETNAME}	fills 1st input image name parameters by catalog slot number
\zxhlp{GETONAME}	fills 1st output image name parameters by catalog slot number
\zxhlp{GETVERS}	finds maximum version number of an extension file
\zxhlp{HGEOM}	interpolates image to different gridding and/or geometry
\zxhlp{HIEND}	End record number in a history-file operation
\zxhlp{HINOTE}	adds user-generated lines to the history extension file
\zxhlp{HISTART}	Start record number in a history-file operation
\zxhlp{HITEXT}	writes lines from history extension file to text file
\zxhlp{IM2HEAD}	displays the image 2 header contents to terminal, message file
\zxhlp{IM3HEAD}	displays the image 3 header contents to terminal, message file
\zxhlp{IM4HEAD}	displays the image 4 header contents to terminal, message file
\zxhlp{IM5HEAD}	displays the image 5 header contents to terminal, message file
\zxhlp{IMDIST}	determines spherical distance between two pixels
\zxhlp{IMHEADER}	displays the image header contents to terminal, message file
\zxhlp{IMOHEAD}	displays the output image header contents
\zxhlp{IMPOS}	displays celestial coordinates selected by the TV cursor
\zxhlp{IMVAL}	returns image intensity and coordinate at specified pixel
\zxhlp{IN2CLASS}	specifies the "class" of the 2nd input image or data base
\zxhlp{IN2DISK}	specifies the disk drive of the 2nd input image or data base
\zxhlp{IN2EXT}	specifies the type of the 2nd input extension file
\zxhlp{IN2NAME}	specifies the "name" of the 2nd input image or data base
\zxhlp{IN2SEQ}	specifies the sequence # of the 2nd input image or data base
\zxhlp{IN2TYPE}	specifies the type of the 2nd input image or data base
\zxhlp{IN2VERS}	specifies the version number of the 2nd input extension file
\zxhlp{IN3CLASS}	specifies the "class" of the 3rd input image or data base
\zxhlp{IN3DISK}	specifies the disk drive of the 3rd input image or data base
\zxhlp{IN3EXT}	specifies the type of the 3rd input extension file
\zxhlp{IN3NAME}	specifies the "name" of the 3rd input image or data base
\zxhlp{IN3SEQ}	specifies the sequence # of the 3rd input image or data base
\zxhlp{IN3TYPE}	specifies the type of the 3rd input image or data base
\zxhlp{IN3VERS}	specifies the version number of the 3rd input extension file
\zxhlp{IN4CLASS}	specifies the "class" of the 4th input image or data base
\zxhlp{IN4DISK}	specifies the disk drive of the 4th input image or data base
\zxhlp{IN4NAME}	specifies the "name" of the 4th input image or data base
\zxhlp{IN4SEQ}	specifies the sequence # of the 4th input image or data base
\zxhlp{IN4TYPE}	specifies the type of the 4th input image or data base
\zxhlp{IN5CLASS}	specifies the "class" of the 5th input image or data base
\zxhlp{IN5DISK}	specifies the disk drive of the 5th input image or data base
\zxhlp{IN5NAME}	specifies the "name" of the 5th input image or data base
\zxhlp{IN5SEQ}	specifies the sequence # of the 5th input image or data base
\zxhlp{IN5TYPE}	specifies the type of the 5th input image or data base
\zxhlp{INCLASS}	specifies the "class" of the 1st input image or data base
\zxhlp{INDISK}	specifies the disk drive of the 1st input image or data base
\zxhlp{INEXT}	specifies the type of the 1st input extension file
\zxhlp{INNAME}	specifies the "name" of the 1st input image or data base
\zxhlp{INSEQ}	specifies the sequence # of the 1st input image or data base
\zxhlp{INTYPE}	specifies the type of the 1st input image or data base
\zxhlp{INVERS}	specifies the version number of the 1st input extension file
\zxhlp{KEYSTRNG}	gives contents of character-valued keyword parameter

\zxhlp{KEYTYPE}	Adverb giving the keyword data type code
\zxhlp{KEYVALUE}	gives contents of numeric-valued keyword parameter
\zxhlp{KEYWORD}	gives name of keyword parameter - i.e. name of header field
\zxhlp{LGEOM}	regrids images with rotation, shift using interpolation
\zxhlp{M2CAT}	displays images in the user's catalog directory for \zxhlp{IN2DISK}
\zxhlp{M3CAT}	displays images in the user's catalog directory for \zxhlp{IN3DISK}
\zxhlp{M4CAT}	displays images in the user's catalog directory for \zxhlp{IN4DISK}
\zxhlp{M5CAT}	displays images in the user's catalog directory for \zxhlp{IN5DISK}
\zxhlp{MCAT}	lists images in the user's catalog directory on disk \zxhlp{INDISK}
\zxhlp{MCAT}	displays images in the user's catalog directory for \zxhlp{OUTDISK}
\zxhlp{MOVE}	Task to copy or move data from one user or disk to another
\zxhlp{NAMEGET}	fills 1st input image name parameters by default matching
\zxhlp{OUT2CLAS}	The class of a secondary output file
\zxhlp{OUT2DISK}	The disk number of a secondary output file.
\zxhlp{OUT2NAME}	The name of a secondary output file.
\zxhlp{OUT2SEQ}	The sequence of a secondary output file.
\zxhlp{OUTCLASS}	The class of an output file
\zxhlp{OUTDISK}	The disk number of an output file.
\zxhlp{OUTNAME}	The name of an output file.
\zxhlp{OUTSEQ}	The sequence of an output file.
\zxhlp{OUTVERS}	The output version number of an table or extension file.
\zxhlp{PCAT}	Verb to list entries in the user's catalog (no log file).
\zxhlp{PLVER}	specifies the version number of a PL extension file
\zxhlp{PRTHI}	prints selected contents of the history extension file
\zxhlp{PUTHEAD}	Verb to modify image header parameters.
\zxhlp{Q2HEADER}	Verb to summarize the image 2 header: positions at center
\zxhlp{Q3HEADER}	Verb to summarize the image 3 header: positions at center
\zxhlp{Q4HEADER}	Verb to summarize the image 4 header: positions at center
\zxhlp{Q5HEADER}	Verb to summarize the image 5 header: positions at center
\zxhlp{QGETVERS}	finds maximum version number of an extension file quietly
\zxhlp{QHEADER}	Verb to summarize the image header: positions at center
\zxhlp{QOHEADER}	Verb to summarize the output image header: center positions
\zxhlp{QUAL}	Source qualifier
\zxhlp{REASON}	The reason for an operation
\zxhlp{RECAT}	Verb to compress the entries in a catalog file
\zxhlp{REMDISK}	removes a computer's disks from the current AIPS session
\zxhlp{RENAME}	Rename a file (UV or Image)
\zxhlp{RENUMBER}	Verb to change the catalog number of an image.
\zxhlp{RESCALE}	Verb to modify image scale factor and offset
\zxhlp{REVERSN}	checks disk for presence of extension files
\zxhlp{SCRDEST}	Verb to destroy scratch files left by bombed tasks.
\zxhlp{SLOT}	Specifies AIPS catalog slot number
\zxhlp{STALIN}	revises history by deleting lines from history extension file
\zxhlp{TVDIST}	determines spherical distance between two pixels on TV screen
\zxhlp{U2CAT}	list a user's UV and scratch files on disk \zxhlp{IN2DISK}
\zxhlp{U3CAT}	list a user's UV and scratch files on disk \zxhlp{IN3DISK}
\zxhlp{U4CAT}	list a user's UV and scratch files on disk \zxhlp{IN4DISK}
\zxhlp{U5CAT}	list a user's UV and scratch files on disk \zxhlp{IN5DISK}
\zxhlp{UCAT}	list a user's UV and scratch files on disk \zxhlp{INDISK}
\zxhlp{UOCAT}	list a user's UV and scratch files on disk \zxhlp{OUTDISK}
\zxhlp{USERID}	User number
\zxhlp{ZAP}	Delete a catalog entry and its extension files

13.8 COORDINA

\zxhlp{ALTDEF}	Sets frequency vs velocity relationship into image header
\zxhlp{ALTSWTCH}	Switches between frequency and velocity in image header
\zxhlp{CODECIML}	Convert between decimal and sexagesimal coordinate values
\zxhlp{COODEFIN}	Define or modify an image axis coordinate description
\zxhlp{COORDINA}	Array to hold coordinate values
\zxhlp{COPIXEL}	Convert between physical and pixel coordinate values
\zxhlp{COSTAR}	Verb to plot a symbol at given position on top of a TV image
\zxhlp{COTVLOD}	Proc to load an image into a TV channel about a coordinate
\zxhlp{COWINDOW}	Set a window based on coordinates
\zxhlp{EPOCONV}	Convert between J2000 and B1950 coordinates
\zxhlp{EPOSWTCH}	Switches between B1950 and J2000 coordinates in header
\zxhlp{FRMAP}	Task to build a map using fringe rate spectra
\zxhlp{FSHIFT}	specifies a position shift - output from fitting routines
\zxhlp{NAXIS}	Axis number
\zxhlp{OBEDT}	Task to flag data of orbiting antennas
\zxhlp{OBTAB}	Recalculate orbit parameters and other spacecraft info
\zxhlp{PIX2VAL}	An image value in the units specified in the header.
\zxhlp{PIX2XY}	Specifies a pixel in an image
\zxhlp{RASHIFT}	Shift in RA
\zxhlp{REGRD}	Regrids an image from one co-ordinate frame to another
\zxhlp{RESTFREQ}	Rest frequency of a transition
\zxhlp{ROTATE}	Specifies a rotation
\zxhlp{SHIFT}	specifies a position shift
\zxhlp{SKYVE}	Regrids a DSS image from one co-ordinate frame to another
\zxhlp{SYSVEL}	Systemic velocity
\zxhlp{VELDEF}	Specifies velocity definition
\zxhlp{VELTYP}	Velocity frame of reference
\zxhlp{XINC}	increment associated with an array of numbers
\zxhlp{XPARAM}	General adverb for up to 10 parameters, may refer to X coord
\zxhlp{XTRAN}	Create an image with transformed coordinates
\zxhlp{XTYPE}	Specify type of process, often the X axis type of an image
\zxhlp{XYRATIO}	Ratio of X to Y units per pixel
\zxhlp{ZINC}	Set the increment of the third axis
\zxhlp{ZXRATIO}	Ratio between Z axis (pixel value) and X axis

13.9 EDITING

\zxhlp{ACLIP}	edits auto-corr data for amplitudes, phases, and weights out of range
\zxhlp{BPEDT}	Interactive TV task to edit uv data based on BP tables
\zxhlp{CLIP}	edits data based on amplitudes, phases, and weights out of range
\zxhlp{CROWDED}	allows a task to perform its function in a crowded fashion
\zxhlp{DEFLG}	edits data based on decorrelation over channels and time
\zxhlp{DOFLAG}	Controls closure cutoff in gain solutions and flagging
\zxhlp{DOPLOT}	Controls plotting of something
\zxhlp{DOROBUST}	Controls method of averaging - simple mean/rms or robust
\zxhlp{EDITA}	Interactive TV task to edit uv data based on TY/SY/SN/CL tables
\zxhlp{EDITR}	Interactive baseline-oriented visibility editor using the TV
\zxhlp{EXPERT}	specifies an user experience level or mode
\zxhlp{FGCNT}	Counts samples comparing two flag tables
\zxhlp{FGDIF}	Compares affect of 2 FG tables
\zxhlp{FINDR}	Find normal values for a uv data set
\zxhlp{FLAGR}	Edit data based on internal RMS, amplitudes, weights
\zxhlp{FLGIT}	flags data based on the rms of the spectrum
\zxhlp{FTFLG}	interactive flagging of UV data in channel-time using the TV
HLPEDIBP	Interactive BP table uv-data editor \zxhlp{BPEDT} - run-time help
HLPEDICL	Interactive SN/CL table uv-data editor - run-time help

```

HLPEDIPC  Interactive PC table editor \zxhlp{PCEDT} - run-time help
HLPEDISN  Interactive SN/CL table (not UV) editor - run-time help
HLPEDISS  Interactive SY table (not UV) editor - run-time help
HLPEDISY  Interactive SY table uv-data editor \zxhlp{EDITA} - run-time help
HLPEDITS  Interactive TY table (not UV) editor - run-time help
HLPEDITY  Interactive TY table uv-data editor \zxhlp{EDITA} - run-time help
HLPEDIUV  Interactive uv-data editor \zxhlp{EDITR} - run-time help
HLPFTFLG  Interactive time-channel visibility Editor - run-time help
HLPFLAG   Edit uv-data on a grid \zxhlp{UFLAG} - run-time help
HLPWIPER  edit uv data from \zxhlp{UVPLT}-like plot \zxhlp{WIPER} - run-time help
\zxhlp{IBLED}  Interactive BaseLine based visibility EDitor
\zxhlp{ISCALIB}  states that the current source is a point-source calibrator
\zxhlp{NOISE}  estimates the noise in images, noise level cutoff
\zxhlp{OFLAG}  uses on-line flag table information to write a flag table
\zxhlp{OUTFGVER}  selects version of the flagging table to be written
\zxhlp{PCEDT}  Interactive TV task to edit PC tables
\zxhlp{PCFLG}  interactive flagging of Pulse-cal data in channel-TB using the TV
\zxhlp{REFLG}  Attempts to compress a flag table
\zxhlp{RFI}  Look for \zxhlp{RFI} in uv data
\zxhlp{RFLAG}  Flags data set based on time and freq rms in fringe visibilities
\zxhlp{SCANLENG}  specify length of "scan"
\zxhlp{SCIMG}  Full-featured imaging plus self-calibration loop with editing
\zxhlp{SCMAP}  Imaging plus self-calibration loop with editing
\zxhlp{SCUTOFF}  noise level cutoff
\zxhlp{SDLSF}  least squares fit to channels and subtracts from SD uv data
\zxhlp{SNEDT}  Interactive SN/CL/TY/SY table editor using the TV
\zxhlp{SNFLG}  Writes flagging info based on the contents of SN files
\zxhlp{SPFLG}  interactive flagging of UV data in channel-TB using the TV
\zxhlp{TABED}  Task to edit tables
\zxhlp{TAFLG}  Flags data in a Table extension file
\zxhlp{TVFLG}  interactive flagging of UV data using the TV
\zxhlp{UFLAG}  Plots and edits data using a uv-plane grid and the TV
\zxhlp{UVFLG}  Flags UV-data
\zxhlp{UVFND}  prints selected data from UV data set to search for problems
\zxhlp{UVLIN}  Fits and removes continuum visibility spectrum, also can flag
\zxhlp{UVLSD}  least squares fit to channels and divides the uv data.
\zxhlp{UVLSF}  least squares fit to channels and subtracts from uv data.
\zxhlp{UVMLN}  edits data based on the rms of the spectrum
\zxhlp{VPFLG}  Resets flagging to all or all corss-hand whenever some are flagged
\zxhlp{WETHR}  Plots selected contents of WX tables, flags data based on WX
\zxhlp{WIPER}  plots and edits data from a UV data base using the TV

```

13.10 EXT-APPL

```

\zxhlp{BSPRT}      print BS tables
\zxhlp{GETTHEAD}   returns keyword and other values value from a table header
\zxhlp{MF2ST}       Task to generate an ST ext. file from Model Fit ext. file
\zxhlp{PRTSY}       Task to print statistics from the SY table
\zxhlp{PUTTHEAD}   inserts a given value into a table keyword/value pair
\zxhlp{SY2TY}       Task to generate a TY extension file from an \zxhlp{EVLA} SY table
\zxhlp{TABGET}     returns table entry for specified row, column and subscript.
\zxhlp{TABPUT}     replaces table entry for specified row, column and subscript.

```

13.11 FITS

\zxhlp{CALRD}	Reads calibrator source model-image FITS file
\zxhlp{CALWR}	writes calibrator source model images w CC files to FITS disk files
\zxhlp{DATAIN}	specifies name of input FITS disk file
\zxhlp{DATAOUT}	specifies name of output FITS disk file
\zxhlp{FILEZAP}	Delete an external file
\zxhlp{FIT2A}	reads the fits input file and records it to the output ascii file
\zxhlp{FITAB}	writes images / uv data w extensions to tape in FITS format
\zxhlp{FITDISK}	writes images / uv data w extensions to disk in FITS format
\zxhlp{FITLD}	Reads FITS files to load images or UV (IDI or UVFITS) data to disk
\zxhlp{FITTP}	writes images / uv data w extensions to tape in FITS format
\zxhlp{IMLOD}	reads tape to load images to disk
\zxhlp{READISK}	writes images / uv data w extensions to tape in FITS format
\zxhlp{TCOPY}	Tape to tape copy with some disk FITS support
\zxhlp{UVLOD}	Read export or FITS data from a tape or disk
\zxhlp{VLBALOAD}	Loads VLBA data
\zxhlp{WRTDISK}	writes images / uv data w extensions to tape in FITS format
\zxhlp{WRTPROCS}	Procedures to simplify the reduction of VLBA data

13.12 GENERAL

\zxhlp{ABORTASK}	stops a running task
\zxhlp{ABOUT}	displays lists and information on tasks, verbs, adverbs
\zxhlp{AIPSB}	AIPS main program for executing batch jobs
AIPS	AIPS main program for interactive use
\zxhlp{ANTNAME}	A list of antenna (station) names
\zxhlp{APARM}	General numeric array adverb used many places
\zxhlp{BADDISK}	specifies which disks are to be avoided for scratch files
\zxhlp{BATCHJOB}	Information about \zxhlp{BATCH}
\zxhlp{BCOUNT}	gives beginning location for start of a process
\zxhlp{BITER}	gives beginning point for some iterative process
\zxhlp{BLC}	gives lower-left-corner of selected subimage
\zxhlp{BPARAM}	general numeric array adverb used too many places
\zxhlp{CATEGORY}	List of allowed primary keywords in \zxhlp{HELP} files
\zxhlp{CLRMSG}	deletes messages from the user's message file
\zxhlp{COMMENT}	64-character comment string
\zxhlp{CPARM}	general numeric array adverb used many places
\zxhlp{CPUTIME}	displays current tcpu and real time usage of the AIPS task
\zxhlp{CROWDED}	allows a task to perform its function in a crowded fashion
\zxhlp{DCODE}	General string adverb
\zxhlp{DDISK}	Determines where input \zxhlp{DDT} data is found
\zxhlp{DDT}	verifies correctness and performance using standard problems
\zxhlp{DDTSAVE}	verifies correctness and performance using standard problems
\zxhlp{DDTSIZE}	Determines which type of \zxhlp{DDT} is \zxhlp{RUN}.
\zxhlp{DECIMAL}	specifies if something is in decimal format
\zxhlp{DETIME}	specifies a time interval for an operation (destroy, batch)
\zxhlp{DISKU}	shows disk use by one or all users
\zxhlp{DOALL}	specifies if an operation is done once or for all matching
\zxhlp{DOCONFRM}	selects user confirmation modes of repetitive operation
\zxhlp{DOKEEP}	specifies if something is kept or deleted
\zxhlp{DOSCALE}	specifies if a scaling operation of some sort is to be performed
\zxhlp{DOSCAN}	specifies if a scan-related operation is to be done
\zxhlp{DOWAIT}	selects wait-for-completion mode for running tasks
\zxhlp{DOWEIGHT}	selects operations with data weights
\zxhlp{DPARM}	General numeric array adverb used many places
\zxhlp{DRCHK}	stand-alone program checks system setup files for consistency
\zxhlp{ECOUNT}	give the highest count or iteration for some process

\zxhlp{EDGSKP}	Determines border excluded from comparison or use
\zxhlp{EHEX}	converts decimal to extended hex
\zxhlp{EXPERT}	specifies an user experience level or mode
\zxhlp{EXPLAIN}	displays help + extended information describing a task/symbol
\zxhlp{FILEZAP}	Delete an external file
\zxhlp{FPARM}	General numeric array adverb used in modeling
\zxhlp{FREEESPAC}	displays available disk space for AIPS in local system
\zxhlp{GETDATE}	Convert the current date and time to a string
\zxhlp{GET}	restores previously SAVED full POPS environment
\zxhlp{GNUGPL}	Information about GNU General Public License for AIPS
\zxhlp{GO}	starts a task, detaching it from AIPS or \zxhlp{AIPSB}
\zxhlp{GRADDRES}	specifies user's home address for replies to gripes
\zxhlp{GRDROP}	deletes the specified gripe entry
\zxhlp{GREMAIL}	gives user's e-mail address name for reply to gripe entry
\zxhlp{GRINDEX}	lists users and time of all gripe entries
\zxhlp{GRIPE}	enter a suggestion or bug report for the AIPS programmers
\zxhlp{GRIPR}	standalone program to enter suggestions/complaints to AIPS
\zxhlp{GRLIST}	lists contents of specified gripe entry
\zxhlp{GRNAME}	gives user's name for reply to gripe entry
\zxhlp{GRPHONE}	specifies phone number to call for questions about a gripe
\zxhlp{HELP}	displays information on tasks, verbs, adverbs
\zxhlp{HINOTE}	adds user-generated lines to the history extension file
\zxhlp{HITEXT}	writes lines from history extension file to text file
\zxhlp{IN2FILE}	specifies name of a disk file, outside the regular catalog
\zxhlp{INFILE}	specifies name of a disk file, outside the regular catalog
\zxhlp{INTEXT}	specifies name of input text file, not in regular catalog
\zxhlp{IOTAPE}	Determines which tape drive is used during a \zxhlp{DDT} \zxhlp{RUN}
\zxhlp{LSAPROPO}	Data input to \zxhlp{APROPO} to find what uses what words
\zxhlp{MAPDIF}	Records differences between \zxhlp{DDT} test results and standards
\zxhlp{MDISK}	Determines where input \zxhlp{DDT} data is found
\zxhlp{MSGKILL}	turns on/off the recording of messages in the message file
\zxhlp{MSGSERVER}	Information about the X11-based message server
\zxhlp{MSGSRV}	Information about the X11-based message server
\zxhlp{NBOXES}	Number of boxes
\zxhlp{NCCBOX}	Number of clean component boxes
\zxhlp{NCOUNT}	General adverb, usually a count of something
\zxhlp{NITER}	The number of iterations of a procedure
\zxhlp{NPOINTS}	General adverb giving the number of something
\zxhlp{NTHREAD}	Controls number of threads used by multi-threaded processes in \zxhlp{OBIT}
\zxhlp{OBJECT}	The name of an object
\zxhlp{OFFSET}	General adverb, the offset of something.
\zxhlp{OPCODE}	General adverb, defines an operation
\zxhlp{OPTELL}	The operation to be passed to a task by \zxhlp{TELL}
\zxhlp{OPTYPE}	General adverb, defines a type of operation.
\zxhlp{ORDER}	Adverb used usually to specify the order of polynomial fit
\zxhlp{OTFBS}	Translates on-the-fly continuum SDD format to AIPS UV file
\zxhlp{OTFUV}	Translates on-the-fly single-dish SDD format to AIPS UV file
\zxhlp{OUTFILE}	specifies name of output disk file, not in regular catalog
\zxhlp{OUTTEXT}	specifies name of output text file, not in regular catalog
\zxhlp{OUTVERS}	The output version number of an table or extension file.
\zxhlp{PANIC}	Instructions for what to do when things go wrong
\zxhlp{PIX2VAL}	An image value in the units specified in the header.
\zxhlp{PIXRANGE}	Range of pixel values to display
\zxhlp{PIXVAL}	Value of a pixel
\zxhlp{POSTSCRIP}	General comments about AIPS use of PostScript incl macros
\zxhlp{PRTAC}	prints contents and summaries of the accounting file
\zxhlp{PRTASK}	Task name selected for printed information
\zxhlp{PRTHI}	prints selected contents of the history extension file

13.13. HARDCOPY

13. Current AIPS Software

```

\zxhlp{PRTLIMIT} specifies limits to printing functions
\zxhlp{PRTMSG} prints selected contents of the user's message file
\zxhlp{QUAL} Source qualifier
\zxhlp{READLINE} Information about AIPS use of the GNU readline library.
\zxhlp{REASON} The reason for an operation
\zxhlp{REBYTE} service program to transform byte order of full data sets
\zxhlp{REHEX} converts extended hex string to decimal
\zxhlp{ROTATE} Specifies a rotation
\zxhlp{RPARM} General numeric array adverb used in modeling
\zxhlp{RTIME} Task to test compute times
\zxhlp{SCALR1} General adverb
\zxhlp{SCALR2} General adverb
\zxhlp{SCALR3} General adverb
\zxhlp{SECONDARY} List of allowed secondary keywords in \zxhlp{HELP} files
\zxhlp{SECONDRY} List of allowed secondary keywords in \zxhlp{HELP} files
\zxhlp{SOURCES} A list of source names
\zxhlp{SPARM} General string array adverb
\zxhlp{STALIN} revises history by deleting lines from history extension file
\zxhlp{STRA1} General string adverb
\zxhlp{STRA2} General string adverb
\zxhlp{STRA3} General string adverb
\zxhlp{STRB1} General string adverb
\zxhlp{STRB2} General string adverb
\zxhlp{STRB3} General string adverb
\zxhlp{STRC1} General string adverb
\zxhlp{STRC2} General string adverb
\zxhlp{STRC3} General string adverb
\zxhlp{SUBARRAY} Subarray number
\zxhlp{SYMBOL} General adverb, probably defines a plotting symbol type
\zxhlp{SYS2COM} specifies a command to be sent to the operating system
\zxhlp{SYSCOM} specifies a command to be sent to the operating system
\zxhlp{SYSOUT} specifies the output device used by the system
\zxhlp{SYSTEM} Verb to send a command to the operating system
\zxhlp{TCODE} Deterimins which type of \zxhlp{DDT} is \zxhlp{RUN}.
\zxhlp{TDISK} Deterimins where output \zxhlp{DDT} data is placed
\zxhlp{TELL} Send parameters to tasks that know to read them on the fly
\zxhlp{THEDATE} contains the date and time in a string form
\zxhlp{TIMERANG} Specifies a timerange
\zxhlp{TMASK} Deterimins which tasks are executed when a \zxhlp{DDT} is \zxhlp{RUN}.
\zxhlp{TMODE} Deterimins which input is used when a \zxhlp{DDT} is \zxhlp{RUN}.
\zxhlp{TNAMF} Deterimins which files are input to \zxhlp{DDT}.
\zxhlp{TPMON} Information about the \zxhlp{TPMON} "Daemon"
\zxhlp{VCODE} General string adverb
\zxhlp{VLAC} verifies correctness of continuum calibration software
\zxhlp{VLACSAVE} verifies correctness of continuum calibration
\zxhlp{VLAL} verifies correctness of spectral line calibration software
\zxhlp{VLALSAVE} verifies correctness of continuum calibration
\zxhlp{VLBDDT} Verification tests using simulated data
\zxhlp{VPARM} General numeric array adverb used in modeling
\zxhlp{WHATSNEW} lists changes and new code in the last several AIPS releases
\zxhlp{XHELP} Accesses hypertext help system
\zxhlp{XPARM} General adverb for up to 10 parameters, may refer to X coord
\zxhlp{XTYPE} Specify type of process, often the X axis type of an image
\zxhlp{Y2K} verifies correctness and performance using standard problems
\zxhlp{Y2KSAVE} verifies correctness and performance using standard problems
\zxhlp{YINC} Y axis increment
\zxhlp{YTYPE} Y axis (V) convolving function type

```

13.13 HARDCOPY

\zxhlp{BPRINT}	gives beginning location for start of a printing process
\zxhlp{BSPRT}	print BS tables
\zxhlp{EPRINT}	gives location for end of a printing process
\zxhlp{FACTOR}	scales some display or CLEANing process
\zxhlp{FLMCOMM}	Comment for film recorder image.
\zxhlp{HIEND}	End record number in a history-file operation
\zxhlp{HISTART}	Start record number in a history-file operation
\zxhlp{INTEXT}	specifies name of input text file, not in regular catalog
\zxhlp{ISPEC}	Plots and prints spectrum of region of a cube
\zxhlp{NPLOTS}	gives number of plots per page or per job
\zxhlp{NPRINT}	gives number of items to be printed
\zxhlp{OTFIN}	Lists on-the-fly single-dish SDD format data files
\zxhlp{OUTFILE}	specifies name of output disk file, not in regular catalog
\zxhlp{OUTPRINT}	specifies name of disk file to keep the printer output
\zxhlp{OUTTEXT}	specifies name of output text file, not in regular catalog
\zxhlp{POSTSCRIP}	General comments about AIPS use of PostScript incl macros
\zxhlp{PRINTER}	Verb to set or show the printer(s) used
\zxhlp{PRIORITY}	Limits priority of messages printed
\zxhlp{PRNUMBER}	POPS number of messages
\zxhlp{PRSTART}	First record number in a print operation
\zxhlp{PRTASK}	Task name selected for printed information
\zxhlp{PRTIME}	Time limit
\zxhlp{PRTLEV}	Specified the amount of information requested.
\zxhlp{PRTSD}	prints contents of AIPS single-dish data sets
\zxhlp{PRTUV}	prints contents of a visibility (UV) data set
\zxhlp{RGBGAMMA}	specifies the desired color gamma corrections
\zxhlp{RSPEC}	Plots and prints spectrum of rms of a cube
\zxhlp{TVCPS}	Task to copy a TV screen-image to a PostScript file.
\zxhlp{TVDIC}	Task to copy a TV screen-image to a Dicomed film recorder.
\zxhlp{UVFND}	prints selected data from UV data set to search for problems
\zxhlp{UHVOL}	prints holography data from a UV data base with calibration
\zxhlp{UVPRT}	prints data from a UV data base with calibration

13.14 IMAGE-UT

\zxhlp{BDROP}	gives number of points dropped at the beginning
\zxhlp{BSGEO}	Beam-switched Az-El image to RA-Dec image translation
\zxhlp{CALWR}	writes calibrator source model images w CC files to FITS disk files
\zxhlp{DOMODEL}	selects display of model function
\zxhlp{DORESID}	selects display of differences between model and data
\zxhlp{DOSLICE}	selects display of slice data
\zxhlp{DSKEW}	Geometric interpolation correction for skew
\zxhlp{EDROP}	number of points/iterations to be omitted from end of process
\zxhlp{FIT2A}	reads the fits input file and records it to the output ascii file
\zxhlp{FITAB}	writes images / uv data w extensions to tape in FITS format
\zxhlp{FITDISK}	writes images / uv data w extensions to disk in FITS format
\zxhlp{FITTP}	writes images / uv data w extensions to tape in FITS format
HLPTVHUI	Interactive intensity-hue-saturation display - run-time help
HLPTVRGB	Interactive red-green-blue display - run-time help
HLPTVSPC	Interactive display of spectra from a cube - run-time help
\zxhlp{IMCLP}	Clip an image to a specified range.
\zxhlp{IMLOD}	reads tape to load images to disk
\zxhlp{OGEOM}	Simple image rotation, scaling, and translation
\zxhlp{OHGEO}	Geometric interpolation with correction for 3-D effects
\zxhlp{READISK}	writes images / uv data w extensions to tape in FITS format
\zxhlp{TVSPC}	Display images and spectra from a cube

13.16. IMAGING

13. Current AIPS Software

```
\zxhlp{WRTDISK}      writes images / uv data w extensions to tape in FITS format
\zxhlp{WRTPROCS}    Procedures to simplify the reduction of VLBA data
```

13.15 IMAGE

\zxhlp{CPYRT}	replaces history with readme file, inserts copyright
\zxhlp{IMRMS}	Plot \zxhlp{IMEAN} rms answers
\zxhlp{MAPBM}	Map \zxhlp{VLA} beam polarization
\zxhlp{NANS}	reads an image or a UV data set and looks for NaNs
\zxhlp{PROFL}	Generates plot file for a profile display.
\zxhlp{STRAN}	Task compares ST tables, find image coordinates (e.g. guide star)
\zxhlp{VLABP}	\zxhlp{VLA} antenna beam polarization correction for snapshot images
\zxhlp{XSMTH}	Smooth data along the x axis
\zxhlp{XSUM}	Sum or average images on the x axis
\zxhlp{XTRAN}	Create an image with transformed coordinates

13.16 IMAGING

\zxhlp{AHIST}	Task to convert image intensities by adaptive histogram
\zxhlp{ALLOKAY}	specifies that initial conditions have been met.
\zxhlp{APCLN}	Deconvolves images with CLEAN algorithm
\zxhlp{APGS}	deconvolves image with Gerchberg-Saxton algorithm
\zxhlp{APVC}	Deconvolves images with van Cittert algorithm
\zxhlp{AVOPTION}	Controls type or range of averaging done by a task
\zxhlp{BCOMP}	gives beginning component number for multiple fields
\zxhlp{BLC}	gives lower-left-corner of selected subimage
\zxhlp{BLWUP}	Blow up an image by any positive integer factor.
\zxhlp{BMAJ}	gives major axis size of beam or component
\zxhlp{BMIN}	gives minor axis size of beam or component
\zxhlp{BOX2CC}	Converts \zxhlp{CLBOX} in pixels to \zxhlp{CCBOX} in arc seconds
\zxhlp{BOXES}	Adds Clean boxes to \zxhlp{BOXFILE} around sources from a list
\zxhlp{BOXFILE}	specifies name of Clean box text file
\zxhlp{BOX}	specifies pixel coordinates of subarrays of an image
\zxhlp{BPA}	gives position angle of major axis of beam or component
\zxhlp{BSAVG}	Task to do an \zxhlp{FFT}-weighted sum of beam-switched images
\zxhlp{BSCLN}	Hogbom Clean on beam-switched difference image
\zxhlp{BSGRD}	Task to image beam-switched single-dish data
\zxhlp{BSMAP}	images weak sources with closure phases
\zxhlp{CANDY}	user-definable (paraform) task to create an AIPS image
\zxhlp{CCBOX}	specifies pixel coordinates of subarrays of an image
\zxhlp{CCEDT}	Select CC components in BOXes and above minimum flux.
\zxhlp{CCFND}	prints the contents of a Clean Components extension file.
\zxhlp{CCGAU}	Converts point CLEAN components to Gaussians
\zxhlp{CCMOD}	generates clean components to fit specified source model
\zxhlp{CCMRG}	sums all clean components at the same pixel
\zxhlp{CCRES}	Removes or restores a CC file to a map with a gaussian beam.
\zxhlp{CELLSIZE}	gives the pixel size in physical coordinates
\zxhlp{CHKFC}	makes images of Clean boxes from Boxfile
\zxhlp{CLBOX}	specifies subarrays of an image for Clean to search
\zxhlp{CMETHOD}	specifies the method by which the uv model is computed
\zxhlp{CMODEL}	specifies the method by which the uv model is computed
\zxhlp{COHER}	Baseline Phase coherence measurement
COMAP_DO \zxhlp{MX}	adverbs not changed by \zxhlp{COMAP}
\zxhlp{COMAP}	Procedure to MAP and Self-Calibrate a UVDATA set
COMAP_MX \zxhlp{MX}	adverbs not changed by \zxhlp{COMAP}
COMAP_NA	Procedure to MAP and Self-Calibrate a UVDATA set

```

COMAP_UV Procedure to MAP and Self-Calibrate a UVDATA set
\zxhlp{CONPL} Plots AIPS gridding convolution functions
\zxhlp{CONVL} convolves an image with a gaussian or another image
\zxhlp{CUTOFF} specifies a limit below or above which the operation ends
\zxhlp{CXCLN} Complex Hogbom CLEAN
\zxhlp{DCONV} deconvolves a gaussian from an image
\zxhlp{DEC SHIFT} gives Y-coordinate shift of an image center from reference
\zxhlp{DELBOX} Verb to delete boxes with TV cursor & graphics display.
\zxhlp{DFILEBOX} Verb to delete Clean boxes with TV cursor & write to file
\zxhlp{DO3DIMAG} specifies whether uvw's are reprojected to each field center
\zxhlp{DOGRIDCR} selects correction for gridding convolution function
\zxhlp{DOOSRO} calibrating amplitude and phase, and imaging \zxhlp{VLA} data
\zxhlp{DRAWBOX} Verb to draw Clean boxes on the display
\zxhlp{DTSUM} Task to provide a summary of the contents of a dataset
\zxhlp{EVAUV} Subtracts & divides a model into UV data, does statistics on results
\zxhlp{FACES} makes images of catalog sources for initial calibration
\zxhlp{FACTOR} scales some display or CLEANing process
\zxhlp{FETCH} Reads an image from an external text file.
\zxhlp{FFT} takes Fourier Transform of an image or images
\zxhlp{FGAUSS} Minimum flux to Clean to by widths of Gaussian models
\zxhlp{FILEBOX} Verb to reset Clean boxes with TV cursor & write to file
\zxhlp{FILIT} Interactive \zxhlp{BOXFILE} editing with facet images
\zxhlp{FIXBX} converts a \zxhlp{BOXFILE} to another for input to \zxhlp{IMAGR}
\zxhlp{FLATN} Re-grid multiple fields into one image incl sensitivity
\zxhlp{FLDSIZE} specifies size(s) of images to be processed
\zxhlp{FLUX} gives a total intensity value for image/component or to limit
\zxhlp{FOV} Specifies the field of view
\zxhlp{FSHIFT} specifies a position shift - output from fitting routines
\zxhlp{FSIZE} file size in Megabytes
\zxhlp{GAIN} specifies loop gain for deconvolutions
\zxhlp{GUARD} portion of UV plane to receive no data in gridding
\zxhlp{HA2TI} Converts data processed by \zxhlp{TI2HA} (STUFFER) back to real times
\zxhlp{HISEQ} task to translate image by histogram equalization

HLCLEAN Cleaning tasks - run-time help
HLPCIMG Full-featured image plus self-cal loops, editing - run-time help
HLPSCMAP Imaging plus self-cal and editing \zxhlp{SCMAP} - run-time help
\zxhlp{HOLGR} Read & process holography visibility data to telescope images
\zxhlp{HOLOG} Read & process holography visibility data to telescope images
\zxhlp{HYB} \zxhlp{RUN} to set parameters for HYBRID (\zxhlp{CALIB}/\zxhlp{MX}) self-cal imaging
\zxhlp{IM2CC} Task to convert an image to multi-facet Clean Components
\zxhlp{IM2PARM} Specifies enhancement parameters for OOP-based imaging: 2nd set
\zxhlp{IM2UV} converts an image to a visibility data set
\zxhlp{IMAGR} Wide-field and/or wide-frequency Cleaning / imaging task.
\zxhlp{IMAGRPRM} Specifies enhancement parameters for OOP-based imaging
\zxhlp{IMERG} merges images of different spatial resolutions
\zxhlp{IMSIZE} specifies number of pixels on X and Y axis of an image
\zxhlp{IMTXT} Write an image to an external text file.
\zxhlp{INLIST} specifies name of input disk file, usually a source list
\zxhlp{LINIMAGE} Build image cube from multi-IF data set
\zxhlp{LTESS} makes mosaic images by linear combination
\zxhlp{MANDL} creates an image of a subset of the Mandelbrot Set
\zxhlp{MAPPR} Simplified access to \zxhlp{IMAGR}
\zxhlp{MAXPIXEL} maximum pixels searched for components in Clark CLEAN
\zxhlp{MODVF} task to create a warped velocity field
\zxhlp{MWFLT} applies linear & non-linear filters to images
\zxhlp{NBOXES} Number of boxes
\zxhlp{NCCBOX} Number of clean component boxes
\zxhlp{NCOMP} Number of CLEAN components

```

13.16. IMAGING

13. Current AIPS Software

\zxhlp{NDIG}	Number of digits to display
\zxhlp{NFIELD}	The number of fields imaged
\zxhlp{NMAPS}	Number of maps (images) in an operation
\zxhlp{NOIFS}	makes all IFs into single spectrum
\zxhlp{NOISE}	estimates the noise in images, noise level cutoff
\zxhlp{OBITIMAG}	Access to \zxhlp{OBIT} task Imager without self-cal or peeling
\zxhlp{OBITMAP}	Simplified access to \zxhlp{OBIT} task Imager
\zxhlp{OBITPEEL}	Access to \zxhlp{OBIT} task Imager with self-cal and peeling
\zxhlp{OBITSCAL}	Access to \zxhlp{OBIT} task Imager with self-cal, NOT peeling
\zxhlp{OBOXFILE}	specifies name of output Clean box text file
\zxhlp{ONEBEAM}	specifies whether one beam is made for all facets or one for each
\zxhlp{ONEFREQ}	states that the current CC model was made with one frequency
\zxhlp{OOSUB}	Subtracts/divides a model from/into a uv data base
\zxhlp{OUT2CLAS}	The class of a secondary output file
\zxhlp{OUT2DISK}	The disk number of a secondary output file.
\zxhlp{OUT2NAME}	The name of a secondary output file.
\zxhlp{OUT2SEQ}	The sequence of a secondary output file.
\zxhlp{OVERLAP}	specifies how overlaps are to be handled
\zxhlp{OVRSWTCH}	specifies when \zxhlp{IMAGR} switches from \zxhlp{OVERLAP} >= 2 to \zxhlp{OVERLAP} = 1 mode
\zxhlp{PADIM}	Task to increase image size by padding with some value
\zxhlp{PASTE}	Pastes a selected subimage of one image into another.
\zxhlp{PATGN}	Task to create a user specified test or primary-beam pattern
\zxhlp{PBCOR}	Task to apply or correct an image for a primary beam
\zxhlp{PBPARM}	Primary beam parameters
\zxhlp{PBSIZE}	estimates the primary beam size in interferometer images
\zxhlp{PGEOM}	Task to transform an image into polar coordinates.
\zxhlp{PHASE}	Baseline Phase coherence measurement
\zxhlp{PHAT}	Prussian hat size
\zxhlp{PIPEAIPS}	calibrating amplitude and phase, and imaging \zxhlp{VLA} data
\zxhlp{PIX2VAL}	An image value in the units specified in the header.
\zxhlp{PIX2XY}	Specifies a pixel in an image
\zxhlp{PIXAVG}	Average image value
\zxhlp{PIXRANGE}	Range of pixel values to display
\zxhlp{PIXSTD}	RMS pixel deviation
\zxhlp{PIXVAL}	Value of a pixel
\zxhlp{PIXXY}	Specifies a pixel location.
\zxhlp{PRTCC}	prints the contents of a Clean Components extension file.
\zxhlp{PUTVALUE}	Verb to store a pixel value at specified position
\zxhlp{QCREATE}	adverb controlling the way large files are created
\zxhlp{QUANTIZE}	Quantization level to use
\zxhlp{REBOX}	Verb to reset boxes with TV cursor & graphics display.
\zxhlp{REGRD}	Regrids an image from one co-ordinate frame to another
\zxhlp{REMAG}	Task to replace magic blanks with a user specified value
\zxhlp{RMSD}	Calculate rms for each pixel using data at the box around the pixel
\zxhlp{ROBUST}	Uniform weighting "robustness" parameter
\zxhlp{RSTOR}	Restores a CC file to a map with a gaussian beam.
\zxhlp{SABOX}	create box file from source islands in facet images
\zxhlp{SCIMG}	Full-featured imaging plus self-calibration loop with editing
\zxhlp{SCMAP}	Imaging plus self-calibration loop with editing
\zxhlp{SDCLN}	deconvolves image by Clark and then "SDI" cleaning methods
\zxhlp{SDGRD}	Task to select and image random-position single-dish data
\zxhlp{SDIMG}	Task to select and image random-position single-dish data
\zxhlp{SETFC}	makes a \zxhlp{OBOXFILE} for input to \zxhlp{IMAGR}
\zxhlp{SHADW}	Generates the "shadowed" representation of an image
\zxhlp{SHIFT}	specifies a position shift
\zxhlp{SIZEFILE}	return file size plus estimate of \zxhlp{IMAGR} work file size
\zxhlp{SKEW}	Specifies a skew angle
\zxhlp{SKYVE}	Regrids a DSS image from one co-ordinate frame to another

```

\zxhlp{SMODEL}      Source model
\zxhlp{SNCUT}       Specifies minimum signal-to-noise ratio
\zxhlp{SPCOR}       Task to correct an image for a primary beam and spectral index
\zxhlp{SPECR}        Spectral regridding task for UV data
\zxhlp{SPIXR}        Fits spectral indexes to each row of an image incl curvature
\zxhlp{STACK}        Task to co-add a set of 2-dimensional images with weighting
\zxhlp{STEER}        Task which deconvolves the David Steer way.
\zxhlp{STESS}        Task which finds sensitivity in mosaicing
\zxhlp{STFACTOR}    scales star display or SDI CLEANing process
\zxhlp{STUFFR}       averages together data sets in hour angle
\zxhlp{SUBIM}        Task to select a subimage from up to a 7-dim. image
\zxhlp{SUMIM}        Task to sum overlapping, sequentially-numbered images
\zxhlp{TDEPEND}     Time-dependent imaging procedure sequence
TD_SCANS           Time-dependent imaging procedure sequence: find intervals
TD_SCAN            Time-dependent imaging procedure sequence: find intervals
TD_STEP3           Time-dependent imaging procedure "step 3"
TD_STEP5           Time-dependent imaging procedure sequence: later steps
\zxhlp{TKBOX}       Procedure to set a Clean box with the TK cursor
\zxhlp{TKNBOXS}     Procedure to set Clean boxes 1 - n with the TK cursor
\zxhlp{TRANSCOD}   Specified desired transposition of an image
\zxhlp{TRANS}        Task to transpose a subimage of an up to 7-dim. image
\zxhlp{TRC}          Specified the top right corner of a subimage
\zxhlp{TVBOX}       Verb to set boxes with TV cursor & graphics display.
\zxhlp{UBAVG}        Baseline dependent time averaging of uv data
\zxhlp{UTESS}        deconvolves images by maximizing emptiness
\zxhlp{UVBOX}       radius of the smoothing box used for uniform weighting
\zxhlp{UVBXFN}     type of function used when counting for uniform weighting
\zxhlp{UVFRE}        Makes one data set have the spectral structure of another
\zxhlp{UVIMG}        Grid UV data into an "image"
\zxhlp{UVMAP}       makes images from calibrated UV data.
\zxhlp{UVPOL}       modifies UV data to make complex image and beam
\zxhlp{UVSIZE}      specifies number of pixels on X and Y axes of a UV image
\zxhlp{UVSUB}        Subtracts/divides a model from/into a uv data base
\zxhlp{UWTFN}        Specify weighting function, Uniform or Natural
\zxhlp{VLARUN}      calibrating amplitude and phase, and imaging \zxhlp{VLA} data
\zxhlp{VTESS}        Deconvolves sets of images by the Maximum Entropy Method
\zxhlp{WFCLN}        Wide field and/or widefrequency CLEANing/imaging task.
\zxhlp{WGAUSS}      Widths of Gaussian models (FWHM)
\zxhlp{WTSUM}        Task to do a a sum of images weighted by other images
\zxhlp{XMOM}         Fits one-dimensional moments to each row of an image
\zxhlp{YPARM}        Specifies Y axis convolving function
\zxhlp{ZEROSP}      Specify how to include zero spacing fluxes in FT of UV data

```

13.17 INFORMAT

\zxhlp{ASTROMET}	Describes the process of astrometric/geodetic reduction in AIPS
\zxhlp{BATCHJOB}	Information about \zxhlp{BATCH}
\zxhlp{CALIBRAT}	describes the process of data calibration in AIPS
\zxhlp{CATEGORY}	List of allowed primary keywords in \zxhlp{HELP} files
\zxhlp{GNUGPL}	Information about GNU General Public License for AIPS
\zxhlp{LSAPROPO}	Data input to \zxhlp{APROPO} to find what uses what words
\zxhlp{MSGSERVER}	Information about the X11-based message server
\zxhlp{MSGSRV}	Information about the X11-based message server
\zxhlp{NEWTASK}	Information about installing a new task
\zxhlp{NOADVERB}	Information about the lack of a defined adverb or verb
\zxhlp{PANIC}	Instructions for what to do when things go wrong
\zxhlp{POPSDAT}	lists all POPS symbols, used to create them in MEmory files

13.18. INTERACT

13. Current AIPS Software

\zxhlp{POPSYM}	Describes the symbols used in POPS
\zxhlp{POSTSCRIP}	General comments about AIPS use of PostScript incl macros
\zxhlp{PSEUDO}	Description of POPS pseudoverbs - obsolete list file
\zxhlp{READLINE}	Information about AIPS use of the GNU readline library.
\zxhlp{REBYTE}	service program to transform byte order of full data sets
\zxhlp{SECONDARY}	List of allowed secondary keywords in \zxhlp{HELP} files
\zxhlp{SECONDRY}	List of allowed secondary keywords in \zxhlp{HELP} files
\zxhlp{TEKSERVER}	Information about the X-11 Tektronix emulation server
\zxhlp{TEKSRV}	Information about the X-11 Tektronix emulation server
\zxhlp{TPMON}	Information about the \zxhlp{TPMON} "Daemon"
\zxhlp{USERLIST}	Alphabetic and numeric list of \zxhlp{VLA} users, points to real list
\zxhlp{UV1TYPE}	Convolving function type 1, pillbox or square wave
\zxhlp{UV2TYPE}	Convolving function type 2, exponential function
\zxhlp{UV3TYPE}	Convolving function type 3, sinc function
\zxhlp{UV4TYPE}	Convolving function type 4, exponent times sinc function
\zxhlp{UV5TYPE}	Convolving function type 5, spheroidal function
\zxhlp{UV6TYPE}	Convolving function type 6, exponent times BessJ1(x) / x
\zxhlp{WHATSNEW}	lists changes and new code in the last several AIPS releases
\zxhlp{XAS}	Information about TV-Servers
\zxhlp{XVSS}	Information about older Sun OpenWindows-specific TV-Server

13.18 INTERACT

AIPS	AIPS main program for interactive use
\zxhlp{BPEDT}	Interactive TV task to edit uv data based on BP tables
\zxhlp{DELBOX}	Verb to delete boxes with TV cursor & graphics display.
\zxhlp{DFILEBOX}	Verb to delete Clean boxes with TV cursor & write to file
\zxhlp{EDITA}	Interactive TV task to edit uv data based on TY/SY/SN/CL tables
\zxhlp{EDITR}	Interactive baseline-oriented visibility editor using the TV
\zxhlp{FILEBOX}	Verb to reset Clean boxes with TV cursor & write to file
\zxhlp{FTFLG}	interactive flagging of UV data in channel-time using the TV
HLPAGAUS	Interactive Gaussian absorption fitting task \zxhlp{AGAUS} - run-time help
HLCLEAN	Cleaning tasks - run-time help
HLPEDIBP	Interactive BP table uv-data editor \zxhlp{BPEDT} - run-time help
HLPEDICL	Interactive SN/CL table uv-data editor - run-time help
HLPEDIPC	Interactive PC table editor \zxhlp{PCEDT} - run-time help
HLPEDISN	Interactive SN/CL table (not UV) editor - run-time help
HLPEDISS	Interactive SY table (not UV) editor - run-time help
HLPEDISY	Interactive SY table uv-data editor \zxhlp{EDITA} - run-time help
HLPEDITS	Interactive TY table (not UV) editor - run-time help
HLPEDITY	Interactive TY table uv-data editor \zxhlp{EDITA} - run-time help
HLPEDIUV	Interactive uv-data editor \zxhlp{EDITR} - run-time help
HLPFILIT	Interactive Clean box file editing with image display - run-time help
HLPFTFLG	Interactive time-channel visibility Editor - run-time help
HLPIBLED	Interactive Baseline based visibility Editor - run-time help
HLPPCFLG	Interactive time-channel PC table Editor \zxhlp{PCFLG} - run-time help
HLPPPLAYR	OOP TV class demonstration task - run-time help
HLPRMFIT	Polarization fitting task \zxhlp{RMFIT} - run-time help
HLPSCIMG	Full-featured image plus self-cal loops, editing - run-time help
HLPSCMAP	Imaging plus self-cal and editing \zxhlp{SCMAP} - run-time help
HLPSPFLG	Interactive time-channel visibility Editor \zxhlp{SPFLG} - run-time help
HLPTVFLG	Interactive time-baseline visibility Editor \zxhlp{TVFLG} - run-time help
HLPTVHLD	Interactive image display with histogram equalization - run-time help
HLPTVHUI	Interactive intensity-hue-saturation display - run-time help
HLPTVRGB	Interactive red-green-blue display - run-time help
HLPTVSAD	Find & fit Gaussians to an image with interaction - run-time help
HLPTVSPC	Interactive display of spectra from a cube - run-time help

HLPFLAG	Edit uv-data on a grid \zxhlp{UFLAG} - run-time help
HLPWIPER	edit uv data from \zxhlp{UVPLT}-like plot \zxhlp{WIPER} - run-time help
HLPXGAUS	Interactive Gaussian fitting task \zxhlp{XGAUS} - run-time help
HLPZAMAN	Fits 1-dimensional Zeeman model to absorption data - run-time help
HLPZEMAN	Fits 1-dimensional Zeeman model to data - run-time help
\zxhlp{IBLED}	Interactive BaseLine based visibility EDitor
\zxhlp{IMAGR}	Wide-field and/or wide-frequency Cleaning / imaging task.
\zxhlp{MAPPR}	Simplified access to \zxhlp{IMAGR}
\zxhlp{MFITSET}	gets adverbs for running \zxhlp{IMFIT} and \zxhlp{JMFIT}
\zxhlp{OPTELL}	The operation to be passed to a task by \zxhlp{TELL}
\zxhlp{PCEDT}	Interactive TV task to edit PC tables
\zxhlp{PCFLG}	interactive flagging of Pulse-cal data in channel-TB using the TV
\zxhlp{PLAYR}	Verb to load an image into a TV channel
\zxhlp{READ}	Read a value from the users terminal
\zxhlp{READLINE}	Information about AIPS use of the GNU readline library.
\zxhlp{REBOX}	Verb to reset boxes with TV cursor & graphics display.
\zxhlp{SCIMG}	Full-featured imaging plus self-calibration loop with editing
\zxhlp{SCMAP}	Imaging plus self-calibration loop with editing
\zxhlp{SETSLICE}	Set slice endpoints on the TV interactively
\zxhlp{SNEDT}	Interactive SN/CL/TY/SY table editor using the TV
\zxhlp{SPFLG}	interactive flagging of UV data in channel-TB using the TV
\zxhlp{TDEPEND}	Time-dependent imaging procedure sequence
TD_SCANS	Time-dependent imaging procedure sequence: find intervals
TD_SSCAN	Time-dependent imaging procedure sequence: find intervals
TD_STEP3	Time-dependent imaging procedure "step 3"
TD_STEP5	Time-dependent imaging procedure sequence: later steps
\zxhlp{TK1SET}	Verb to reset 1D gaussian fitting initial guess.
\zxhlp{TKBOX}	Procedure to set a Clean box with the TK cursor
\zxhlp{TKNBOXS}	Procedure to set Clean boxes 1 - n with the TK cursor
\zxhlp{TKPOS}	Read a position from the graphics screen or window
\zxhlp{TKSET}	Verb to set 1D gaussian fitting initial guesses.
\zxhlp{TKWIN}	Procedure to set \zxhlp{BLC} and \zxhlp{TRC} with Graphics cursor
\zxhlp{TV1SET}	Verb to reset 1D gaussian fitting initial guess on TV plot.
\zxhlp{TVBOX}	Verb to set boxes with TV cursor & graphics display.
\zxhlp{TVFLG}	interactive flagging of UV data using the TV
\zxhlp{TVSAD}	Finds and fits Gaussians to portions of an image with interaction
\zxhlp{TVSCROL}	Shift position of image on the TV screen
\zxhlp{TVSET}	Verb to set slice Gaussian fitting initial guesses from TV plot
\zxhlp{TVSPC}	Display images and spectra from a cube
\zxhlp{TVSPLIT}	Compare two TV image planes, showing halves
\zxhlp{TVSTAT}	Find the mean and RMS in a blotch region on the TV
\zxhlp{TVTRANSF}	Interactively alters the TV image plane transfer function
\zxhlp{TVWINDOW}	Set a window on the TV with the cursor
\zxhlp{TVZOOM}	Activate the TV zoom
\zxhlp{UFLAG}	Plots and edits data using a uv-plane grid and the TV
\zxhlp{WDERASE}	Load a wedge portion of the TV with zeros
\zxhlp{WIPER}	plots and edits data from a UV data base using the TV
\zxhlp{TVHELIX}	Verb to activate a helical hue-intensity TV pseudo-coloring

13.19 MODELING

```

\zxhlp{ACTNOISE}      puts estimate of actual image uncertainty and zero in header
\zxhlp{AGAUS}          Fits 1-dimensional Gaussians to absorption-line spectra
\zxhlp{BOXES}          Adds Clean boxes to \zxhlp{BOXFILE} around sources from a list
\zxhlp{BSMOD}          creates single-dish UV beam-switched data with model sources
\zxhlp{BSTST}          Graphical display of solutions to frequency-switched data
\zxhlp{CMETHOD}        specifies the method by which the uv model is computed
\zxhlp{CMODEL}         specifies the method by which the uv model is computed
\zxhlp{CUBIT}          Model a galaxy's density and velocity distribution from full cube
\zxhlp{DIFUV}          Outputs the difference of two matching input uv data sets
\zxhlp{DOMODEL}        selects display of model function
\zxhlp{DORESID}        selects display of differences between model and data
\zxhlp{DOSPIX}         selects solutions for spectral index of model components
\zxhlp{EFACTOR}        scales some error analysis process
\zxhlp{EVAUV}          Subtracts & divides a model into UV data, does statistics on results
\zxhlp{FACES}           makes images of catalog sources for initial calibration
\zxhlp{FITOUT}          specifies name of output text file for results of fitting
\zxhlp{FSHIFT}          specifies a position shift - output from fitting routines
\zxhlp{GLENS}           models galaxy gravitational lens acting on 3 component source
HLPAGAUS   Interactive Gaussian absorption fitting task \zxhlp{AGAUS} - run-time help
HLPRMFIT   Polarization fitting task \zxhlp{RMFIT} - run-time help
HLPTVSAD   Find & fit Gaussians to an image with interaction - run-time help
HLPXGAUS   Interactive Gaussian fitting task \zxhlp{XGAUS} - run-time help
HLPZAMAN   Fits 1-dimensional Zeeman model to absorption data - run-time help
HLPZEMAN   Fits 1-dimensional Zeeman model to data - run-time help
\zxhlp{IMFIT}          Fits Gaussians to portions of an image
\zxhlp{IMMOD}          adds images of model objects to an image
\zxhlp{INLIST}          specifies name of input disk file, usually a source list
\zxhlp{JMFIT}           Fits Gaussians to portions of an image
\zxhlp{MFITSET}         gets adverbs for running \zxhlp{IMFIT} and \zxhlp{JMFIT}
\zxhlp{MFPRT}           prints MF tables in a format needed by modelling software
\zxhlp{MODAB}           Makes simple absorption/emission spectral-line image in I/V
\zxhlp{MODIM}           adds images of model objects to image cubes in IQU polarization
\zxhlp{MODSP}           adds images of model objects to image cubes in I/V polarization
\zxhlp{MODVF}           task to create a warped velocity field
\zxhlp{OMFIT}           Fits sources and, optionally, a self-cal model to uv data
\zxhlp{ONEFREQ}          states that the current CC model was made with one frequency
\zxhlp{OOSUB}            Subtracts/divides a model from/into a uv data base
\zxhlp{RADIUS}          Specify a radius in an image
\zxhlp{RM2PL}           Plots spectrum of a pixel with \zxhlp{RMFIT} fit
\zxhlp{RMFIT}            Fits 1-dimensional polarization spectrum to Q/U cube
\zxhlp{RMSLIMIT}         selects things with RMS above this limit
\zxhlp{SAD}              Finds and fits Gaussians to portions of an image
\zxhlp{SDMOD}            modifies single-dish UV data with model sources
\zxhlp{SLFIT}            Task to fit gaussians to slice data.
\zxhlp{SPMOD}            Modify UV database by adding a model with spectral lines
\zxhlp{STVERS}           star display table version number
\zxhlp{TK1SET}           Verb to reset 1D gaussian fitting initial guess.
\zxhlp{TKAMODEL}         Verb to add slice model display directly on TEK
\zxhlp{TKARESID}         Verb to add slice model residuals directly on TEK
\zxhlp{TKGUESS}          Verb to display slice model guess directly on TEK
\zxhlp{TKMODEL}          Verb to display slice model directly on TEK
\zxhlp{TKRESID}          Verb to display slice model residuals directly on TEK
\zxhlp{TKSET}             Verb to set 1D gaussian fitting initial guesses.
\zxhlp{TKSLICE}           Verb to display slice file directly on TEK
\zxhlp{TV1SET}            Verb to reset 1D gaussian fitting initial guess on TV plot.
\zxhlp{TVACOMPS}          Verb to add slice model components directly on TV graphics
\zxhlp{TVAGUESS}          Verb to re-plot slice model guess directly on TV graphics

```

\zxhlp{TVAMODEL}	Verb to add slice model display directly on TV graphics
\zxhlp{TVARESID}	Verb to add slice model residuals directly on TV graphics
\zxhlp{TVASLICE}	Verb to add a slice display on TV graphics from slice file
\zxhlp{TVCOMPS}	Verb to display slice model components directly on TV graphics
\zxhlp{TVGUESS}	Verb to display slice model guess directly on TV graphics
\zxhlp{TVMODEL}	Verb to display slice model directly on TV graphics
\zxhlp{TVRESID}	Verb to display slice model residuals directly on TV graphics
\zxhlp{TVSAD}	Finds and fits Gaussians to portions of an image with interaction
\zxhlp{TVSET}	Verb to set slice Gaussian fitting initial guesses from TV plot
\zxhlp{TVSLICE}	Verb to display slice file directly on TV
\zxhlp{UVFIT}	Fits source models to uv data.
\zxhlp{UVGIT}	Fits source models to uv data.
\zxhlp{UVMOD}	Modify UV database by adding a model incl spectral index
\zxhlp{UVSUB}	Subtracts/divides a model from/into a uv data base
\zxhlp{XG2PL}	Plots spectrum of a pixel with \zxhlp{XGAUS}/\zxhlp{AGAUS} and \zxhlp{ZEMAN}/\zxhlp{ZAMAN} fits
\zxhlp{XGAUS}	Fits 1-dimensional Gaussians to images: restartable
\zxhlp{ZAMAN}	Fits 1-dimensional Zeeman model to absorption-line data
\zxhlp{ZEMAN}	Fits 1-dimensional Zeeman model to data

13.20 OBSOLETE

\zxhlp{ABACKUP}	VMS procedure to back up data on tape
\zxhlp{ARESTORE}	Restores back up tapes of users data
\zxhlp{CLIPM}	
\zxhlp{CODETYPE}	specifies the desired operation type
\zxhlp{HORUS}	makes images from unsorted UV data, applying any calibration
\zxhlp{MX}	makes images & deconvolves using UV data directly - replaced
\zxhlp{OFFROAM}	Procedure to clear the TV from a Roam condition
\zxhlp{PFT}	The Perley-Feigelson Test; see PFTLOAD.\zxhlp{RUN}, PFTEXEC.\zxhlp{RUN}
\zxhlp{PHCLN}	\zxhlp{PHCLN} has been removed, use \zxhlp{PHAT} adverb in \zxhlp{APCLN}.
\zxhlp{PSEUDO}	Description of POPS pseudoverbs - obsolete list file
\zxhlp{SAMPTYPE}	Specifies sampling type
\zxhlp{SETROAM}	Verb use to set roam image mode, then do roam. OBSOLETE
\zxhlp{SNCUT}	Specifies minimum signal-to-noise ratio
\zxhlp{XVSS}	Information about older Sun OpenWindows-specific TV-Server

13.21 ONED

\zxhlp{AGAUS}	Fits 1-dimensional Gaussians to absorption-line spectra
\zxhlp{PFPL2}	Paraform Task to generate a plot file: (slice intensity)
\zxhlp{PLCUB}	Task to plot intensity vs x panels on grid of y,z pixels
\zxhlp{PLROW}	Plot intensity of a series of rows with an offset.
\zxhlp{RM2PL}	Plots spectrum of a pixel with \zxhlp{RMFIT} fit
\zxhlp{RMFIT}	Fits 1-dimensional polarization spectrum to Q/U cube
\zxhlp{SETSLICE}	Set slice endpoints on the TV interactively
\zxhlp{SL2PL}	Task to convert a Slice File to a Plot File
\zxhlp{SLFIT}	Task to fit gaussians to slice data.
\zxhlp{SLICE}	Task to make a slice file from an image
\zxhlp{SLPRT}	Task to print a Slice File
\zxhlp{TK1SET}	Verb to reset 1D gaussian fitting initial guess.
\zxhlp{TKAMODEL}	Verb to add slice model display directly on TEK
\zxhlp{TKARESID}	Verb to add slice model residuals directly on TEK
\zxhlp{TKASLICE}	Verb to add a slice display on TEK from slice file
\zxhlp{TKGUESS}	Verb to display slice model guess directly on TEK
\zxhlp{TKMODEL}	Verb to display slice model directly on TEK
\zxhlp{TKRESID}	Verb to display slice model residuals directly on TEK

\zxhlp{TKSET}	Verb to set 1D gaussian fitting initial guesses.
\zxhlp{TKSLICE}	Verb to display slice file directly on TEK
\zxhlp{TV1SET}	Verb to reset 1D gaussian fitting initial guess on TV plot.
\zxhlp{TVACOMPS}	Verb to add slice model components directly on TV graphics
\zxhlp{TVAGUESS}	Verb to re-plot slice model guess directly on TV graphics
\zxhlp{TVAMODEL}	Verb to add slice model display directly on TV graphics
\zxhlp{TVARESID}	Verb to add slice model residuals directly on TV graphics
\zxhlp{TVASLICE}	Verb to add a slice display on TV graphics from slice file
\zxhlp{TVCOMPS}	Verb to display slice model components directly on TV graphics
\zxhlp{TVGUESS}	Verb to display slice model guess directly on TV graphics
\zxhlp{TVMODEL}	Verb to display slice model directly on TV graphics
\zxhlp{TVRESID}	Verb to display slice model residuals directly on TV graphics
\zxhlp{TVSET}	Verb to set slice Gaussian fitting initial guesses from TV plot
\zxhlp{TVSLICE}	Verb to display slice file directly on TV
\zxhlp{XG2PL}	Plots spectrum of a pixel with \zxhlp{XGAUS}/\zxhlp{AGAUS} and \zxhlp{ZEMAN}/\zxhlp{ZAMAN} f
\zxhlp{XGAUS}	Fits 1-dimensional Gaussians to images: restartable
\zxhlp{XPLOT}	Plots image rows one at a time on the graphics or TV screen
\zxhlp{ZAMAN}	Fits 1-dimensional Zeeman model to absorption-line data
\zxhlp{ZEMAN}	Fits 1-dimensional Zeeman model to data

13.22 OOP

\zxhlp{BLING}	find residual rate and delay on individual baselines
\zxhlp{BPEDT}	Interactive TV task to edit uv data based on BP tables
\zxhlp{BSCOR}	Combines two beam-switched images
\zxhlp{BSGEO}	Beam-switched Az-El image to RA-Dec image translation
\zxhlp{BSGRD}	Task to image beam-switched single-dish data
\zxhlp{CCEDT}	Select CC components in BOXes and above minimum flux.
\zxhlp{CCSEL}	Select signifigant CC components
\zxhlp{DSKEW}	Geometric interpolation correction for skew
\zxhlp{EDITA}	Interactive TV task to edit uv data based on TY/SY/SN/CL tables
\zxhlp{EDITR}	Interactive baseline-oriented visibility editor using the TV
\zxhlp{FILIT}	Interactive \zxhlp{BOXFILE} editing with facet images
\zxhlp{FINDR}	Find normal values for a uv data set
\zxhlp{FIXWT}	Modify weights to reflect amplitude scatter of data
\zxhlp{FLAGR}	Edit data based on internal RMS, amplitudes, weights
\zxhlp{FLATN}	Re-grid multiple fields into one image incl sensitivity
\zxhlp{FRCAL}	Faraday rotation self calibration task
HLPCLEAN	Cleaning tasks - run-time help
HLPEDIBP	Interactive BP table uv-data editor \zxhlp{BPEDT} - run-time help
HLPEDICL	Interactive SN/CL table uv-data editor - run-time help
HLPEDIPC	Interactive PC table editor \zxhlp{PCEDT} - run-time help
HLPEDISN	Interactive SN/CL table (not UV) editor - run-time help
HLPEDISS	Interactive SY table (not UV) editor - run-time help
HLPEDISY	Interactive SY table uv-data editor \zxhlp{EDITA} - run-time help
HLPEDITS	Interactive TY table (not UV) editor - run-time help
HLPEDITY	Interactive TY table uv-data editor \zxhlp{EDITA} - run-time help
HLPEDIUV	Interactive uv-data editor \zxhlp{EDITR} - run-time help
HLPFILIT	Interactive Clean box file editing with image display - run-time help
HLPPLAYR	OOP TV class demonstration task - run-time help
HLPSCIMG	Full-featured image plus self-cal loops, editing - run-time help
HLPSCMAP	Imaging plus self-cal and editing \zxhlp{SCMAP} - run-time help
\zxhlp{IM2PARM}	Specifies enhancement parameters for OOP-based imaging: 2nd set
\zxhlp{IMAGR}	Wide-field and/or wide-frequency Cleaning / imaging task.
\zxhlp{IMAGRPRM}	Specifies enhancement parameters for OOP-based imaging
\zxhlp{IMCLP}	Clip an image to a specified range.
\zxhlp{IMFRING}	large image delay fitting with \zxhlp{IM2CC} and \zxhlp{OOFRING}

```

\zxhlp{IMSCAL}      large image self-cal with \zxhlp{IM2CC} and \zxhlp{OOCAL}
\zxhlp{MAPBM}        Map \zxhlp{VLA} beam polarization
\zxhlp{MBDLY}        Fits multiband delays from IF phases, updates SN table
\zxhlp{MULIF}        Change number of IFs in output
\zxhlp{OGEOM}        Simple image rotation, scaling, and translation
\zxhlp{OHGEO}        Geometric interpolation with correction for 3-D effects
\zxhlp{OMFIT}        Fits sources and, optionally, a self-cal model to uv data
\zxhlp{OOCAL}        determines antenna complex gain with frequency-dependent models
\zxhlp{OOSRT}        Sort a UV dataset into a specified order
\zxhlp{OOSUB}        Subtracts/divides a model from/into a uv data base
\zxhlp{PASTE}         Pastes a selected subimage of one image into another.
\zxhlp{PCEDT}        Interactive TV task to edit PC tables
\zxhlp{PLAYR}         Verb to load an image into a TV channel
\zxhlp{RFI}          Look for \zxhlp{RFI} in uv data
\zxhlp{SABOX}        create box file from source islands in facet images
\zxhlp{SCIMG}        Full-featured imaging plus self-calibration loop with editing
\zxhlp{SCMAP}        Imaging plus self-calibration loop with editing
\zxhlp{SDGRD}        Task to select and image random-position single-dish data
\zxhlp{SNEDT}        Interactive SN/CL/TY/SY table editor using the TV
\zxhlp{TDEPEND}     Time-dependent imaging procedure sequence
TD_SCANS           Time-dependent imaging procedure sequence: find intervals
TD_SSCAN            Time-dependent imaging procedure sequence: find intervals
TD_STEP3            Time-dependent imaging procedure "step 3"
TD_STEP5            Time-dependent imaging procedure sequence: later steps
\zxhlp{UV2MS}        Append single-source file to multi-source file.
\zxhlp{VLABP}        \zxhlp{VLA} antenna beam polarization correction for snapshot images
\zxhlp{WFCLN}        Wide field and/or widefrequency CLEANing/imaging task.

```

13.23 OPTICAL

```

\zxhlp{COSTAR}      Verb to plot a symbol at given position on top of a TV image
\zxhlp{GSTAR}        Task to read a Guide Star (UK) table and create an ST table.
\zxhlp{IMFLT}        fits and removes a background intensity plane from an image
\zxhlp{STFND}        Task to find stars in an image and generate an ST table.
\zxhlp{STRAN}        Task compares ST tables, find image coordinates (e.g. guide star )
\zxhlp{TVSTAR}       Verb to plot star positions on top of a TV image
\zxhlp{XTRAN}        Create an image with transformed coordinates

```

13.24 PARAFORM

```

\zxhlp{CANDY}        user-definable (paraform) task to create an AIPS image
\zxhlp{DTCHK}        Task to check results of a test using simulated data.
\zxhlp{FUDGE}         modifies UV data with user's algorithm: paraform task
\zxhlp{NEWTASK}      Information about installing a new task
\zxhlp{PFPL1}        Paraform Task to generate a plot file: (does grey scale)
\zxhlp{PFPL2}        Paraform Task to generate a plot file: (slice intensity)
\zxhlp{PFPL3}        Paraform Task to generate a plot file: (does histogram)
\zxhlp{TAFFY}         User definable task to operate on an image
\zxhlp{TBTSK}        Paraform OOP task for tables
\zxhlp{UVFIL}        Create, fill a uv database from user supplied information

```

13.25 PLOT

\zxhlp{AGAUS}	Fits 1-dimensional Gaussians to absorption-line spectra
\zxhlp{ALIAS}	adverb to alias antenna numbers to one another
\zxhlp{ANBPL}	plots and prints uv data converted to antenna based values
\zxhlp{ASPMM}	Plot scaling parameter - arc seconds per millimeter on plot
\zxhlp{AVGCHAN}	Controls averaging of spectral channels
\zxhlp{AVGIF}	Controls averaging of IF channels
\zxhlp{BDROP}	gives number of points dropped at the beginning
\zxhlp{BLSUM}	sums images over irregular sub-images, displays spectra
\zxhlp{BPERR}	Print and plot \zxhlp{BPASS} closure outputs
\zxhlp{BPLOT}	Plots bandpass tables in 2 dimensions as function of time
\zxhlp{CANPL}	translates a plot file to a Canon printer/plotter
\zxhlp{CAPLT}	plots closure amplitude and model from CC file
\zxhlp{CBPLOT}	selects a display of a Clean beam full width at half maximum
\zxhlp{CCNTR}	generate a contour plot file from an image
\zxhlp{CIRCLEVS}	Sets \zxhlp{RGBLEVS} to fill \zxhlp{LEVS} with a circular color scheme
\zxhlp{CLEV}	Contour level multiplier in physical units
\zxhlp{CLPLT}	plots closure phase and model from CC file
\zxhlp{CNTR}	generate a contour plot file or TV plot from an image
\zxhlp{CON3COL}	Controls use of full 3-color graphics for contouring
\zxhlp{CONPL}	Plots AIPS gridding convolution functions
\zxhlp{COPIES}	sets the number of copies to be made
\zxhlp{COSTAR}	Verb to plot a symbol at given position on top of a TV image
\zxhlp{DARKLINE}	The level at which vectors are switched from light to dark
\zxhlp{DEFCOLOR}	Sets adverb \zxhlp{PLCOLORS} to match s default \zxhlp{XAS} TV
\zxhlp{DFTPL}	plots DFT of a UV data set at arbitrary point versus time
\zxhlp{DIST}	gives a distance - \zxhlp{PROFL} uses as distance to observer
\zxhlp{DO3COL}	Controls whether full 3-color graphics are used in a plot
\zxhlp{DOALIGN}	specifies how two or more images are aligned in computations
\zxhlp{DOBBLANK}	controls handling of blanking
\zxhlp{DOCELL}	selects units of cells over angular unit
\zxhlp{DOCENTER}	selects something related to centering
\zxhlp{DOCIRCLE}	select a "circular" display (i.e. trace coordinates, ...)
\zxhlp{DOCOLOR}	specifies whether coloring is done
\zxhlp{DOCONT}	selects a display of contour lines
\zxhlp{DOCRT}	selects printer display or CRT display (giving width)
\zxhlp{DODARK}	specifies whether "dark" vectors are plotted dark or light
\zxhlp{DOEBAR}	Controls display of estimates of the uncertainty in the data
\zxhlp{DOFRACT}	Tells whether to compute a fraction or ratio
\zxhlp{DOGREY}	selects a display of a grey-scale image
\zxhlp{DOHIST}	selects a histogram display
\zxhlp{DOHMS}	selects sexagesimal (hours-mins-secs) display format
\zxhlp{DOMODEL}	selects display of model function
\zxhlp{DOPLOT}	Controls plotting of something
\zxhlp{DOPRINT}	selects printer display or CRT display (giving width)
\zxhlp{DORESID}	selects display of differences between model and data
\zxhlp{DOSLICE}	selects display of slice data
\zxhlp{DOVECT}	selects display of polarization vectors
\zxhlp{DOWEDGE}	selects display of intensity step wedge
\zxhlp{EDROP}	number of points/iterations to be omitted from end of process
\zxhlp{ELFIT}	Plots/fits selected contents of SN, TY, SY, PC or CL files
\zxhlp{EXTAB}	exports AIPS table data as tab-separated text
\zxhlp{EXTLIST}	lists detailed information about contents of extension files
\zxhlp{FACTOR}	scales some display or CLEANing process
\zxhlp{FGPLT}	Plots selected contents of FG table
\zxhlp{FLAMLEVS}	Sets \zxhlp{RGBLEVS} to fill \zxhlp{LEVS} with a red flame color scheme
\zxhlp{FRPLT}	Task to plot fringe rate spectra
\zxhlp{FUNCTYPE}	specifies type of intensity transfer function

```

\zxhlp{GCPLT}      Plots gain curves from text files
\zxhlp{GREYS}      plots images as contours over multi-level grey
\zxhlp{GSTAR}       Task to read a Guide Star (UK) table and create an ST table.
HLPFLAG           Edit uv-data on a grid \zxhlp{UFLAG} - run-time help
HLPWIPER          edit uv data from \zxhlp{UVPLT}-like plot \zxhlp{WIPER} - run-time help
\zxhlp{ICUT}        specifies a cutoff level in units of the image
\zxhlp{IMEAN}       displays the mean & extrema and plots histogram of an image
\zxhlp{IMRMS}       Plot \zxhlp{IMEAN} rms answers
\zxhlp{IMVIM}       plots one image's values against another's
\zxhlp{IRING}       integrates intensity / flux in rings / ellipses
\zxhlp{ISPEC}        Plots and prints spectrum of region of a cube
\zxhlp{KNTR}         make a contour/grey plot file from an image w multiple panels
\zxhlp{LABEL}        selects a type of extra labeling for a plot
\zxhlp{LAYER}        Task to create an RGB image from multiple images
\zxhlp{LEVS}         list of multiples of the basic level to be contoured
\zxhlp{LPEN}         specifies the "pen width" code # => width of plotted lines
\zxhlp{LTYPE}        specifies the type and degree of axis labels on plots
\zxhlp{LWPLA}        translates plot file(s) to a PostScript printer or file
\zxhlp{MF2ST}        Task to generate an ST ext. file from Model Fit ext. file
\zxhlp{NPLOTS}       gives number of plots per page or per job
\zxhlp{NX}           General adverb referring to a number of things in the Y direction
\zxhlp{NY}           General adverb referring to a number of things in the Y direction
\zxhlp{OBPLT}        Plot columns of an OB table.
\zxhlp{OFMFILE}      specifies the name of a text file containing OFM values
\zxhlp{PCFIT}        Finds delays and phases using a PC table
\zxhlp{PCNTR}        Generate plot file with contours plus polarization vectors
\zxhlp{PCPLT}        Plots pulse-cal tables in 2 dimensions as function of time
\zxhlp{PCUT}         Cutoff in polarized intensity
\zxhlp{PFPL1}        Paraform Task to generate a plot file: (does grey scale)
\zxhlp{PFPL2}        Paraform Task to generate a plot file: (slice intensity)
\zxhlp{PFPL3}        Paraform Task to generate a plot file: (does histogram)
\zxhlp{PLCOLORS}     specifies the colors to be used
\zxhlp{PLCUB}        Task to plot intensity vs x panels on grid of y,z pixels
\zxhlp{PLEV}         Percentage of peak to use for contour levels
\zxhlp{PLGET}        gets the adverbs used to make a particular plot file
\zxhlp{PLOTR}        Basic task to generate a plot file from text input
\zxhlp{PLROW}        Plot intensity of a series of rows with an offset.
\zxhlp{PLVER}        specifies the version number of a PL extension file
\zxhlp{POL3COL}     Controls use of full 3-color graphics for polarization lines
\zxhlp{POLPLOT}      specifies the desired polarization ratio before plotting.
\zxhlp{POSTSCRIP}   General comments about AIPS use of PostScript incl macros
\zxhlp{PRINTER}      Verb to set or show the printer(s) used
\zxhlp{PROFL}        Generates plot file for a profile display.
\zxhlp{PRTAB}        prints any table-format extension file
\zxhlp{PRTIM}        prints image intensities from an MA catalog entry
\zxhlp{PRTPL}        Task to send a plot file to the line printer
\zxhlp{QMSPL}        Task to send a plot file to the QMS printer/plotter
\zxhlp{RAINLEVS}    Sets \zxhlp{RGBLEVS} to fill \zxhlp{LEVS} with a rainbow color scheme
\zxhlp{RGBLEVS}     colors to be applied to the contour levels
\zxhlp{RM2PL}        Plots spectrum of a pixel with \zxhlp{RMFIT} fit
\zxhlp{RMFIT}        Fits 1-dimensional polarization spectrum to Q/U cube
\zxhlp{RSPEC}        Plots and prints spectrum of rms of a cube
\zxhlp{SCLIM}        operates on an image with a choice of mathematical functions
\zxhlp{SL2PL}        Task to convert a Slice File to a Plot File
\zxhlp{SLPRT}        Task to print a Slice File
\zxhlp{SNFIT}        Fits parabola to SN amplitudes and plots result
\zxhlp{SNIFS}        Plots selected contents of SN, TY, SY, PC or CL files
\zxhlp{SNPLT}        Plots selected contents of SN, SY, TY, PC or CL files

```

13.25. PLOT

13. Current AIPS Software

\zxhlp{STARS}	Task to generate an ST ext. file with star positions
\zxhlp{STEPLEVS}	Sets \zxhlp{RGBLEVS} to fill \zxhlp{LEVS} with a repeated sequence of colors
\zxhlp{STFND}	Task to find stars in an image and generate an ST table.
\zxhlp{STVERS}	star display table version number
\zxhlp{SYMBOL}	General adverb, probably defines a plotting symbol type
\zxhlp{TAPLT}	Plots data from a Table extension file
\zxhlp{TARPL}	Plot output of \zxhlp{TARS} task
\zxhlp{TEKSERVER}	Information about the X-11 Tektronix emulation server
\zxhlp{TEKSRV}	Information about the X-11 Tektronix emulation server
\zxhlp{TKAMODEL}	Verb to add slice model display directly on TEK
\zxhlp{TKARESID}	Verb to add slice model residuals directly on TEK
\zxhlp{TKASLICE}	Verb to add a slice display on TEK from slice file
\zxhlp{TKBOX}	Procedure to set a Clean box with the TK cursor
\zxhlp{TKERASE}	Erase the graphics screen or window
\zxhlp{TKGUESS}	Verb to display slice model guess directly on TEK
\zxhlp{TKMODEL}	Verb to display slice model directly on TEK
\zxhlp{TKNBOXS}	Procedure to set Clean boxes 1 - n with the TK cursor
\zxhlp{TKPL}	Task to send a plot file to the TEK
\zxhlp{TKPOS}	Read a position from the graphics screen or window
\zxhlp{TKRESID}	Verb to display slice model residuals directly on TEK
\zxhlp{TKSLICE}	Verb to display slice file directly on TEK
\zxhlp{TKWIN}	Procedure to set \zxhlp{BLC} and \zxhlp{TRC} with Graphics cursor
\zxhlp{TVACOMPS}	Verb to add slice model components directly on TV graphics
\zxhlp{TVAGUESS}	Verb to re-plot slice model guess directly on TV graphics
\zxhlp{TVAMODEL}	Verb to add slice model display directly on TV graphics
\zxhlp{TVARESID}	Verb to add slice model residuals directly on TV graphics
\zxhlp{TVASLICE}	Verb to add a slice display on TV graphics from slice file
\zxhlp{TCOLORS}	Sets adverb \zxhlp{PLCOLORS} to match the TV (\zxhlp{DOTV}=1) usage
\zxhlp{TVCOMPS}	Verb to display slice model components directly on TV graphics
\zxhlp{TVGUESS}	Verb to display slice model guess directly on TV graphics
\zxhlp{TVMODEL}	Verb to display slice model directly on TV graphics
\zxhlp{TVPL}	Display a plot file on the TV
\zxhlp{TVRESID}	Verb to display slice model residuals directly on TV graphics
\zxhlp{TVSLICE}	Verb to display slice file directly on TV
\zxhlp{TVSTAR}	Verb to plot star positions on top of a TV image
\zxhlp{TXPL}	Displays a plot (PL) file on a terminal or line printer
\zxhlp{UFLAG}	Plots and edits data using a uv-plane grid and the TV
\zxhlp{UVHGM}	Plots statistics of uv data files as histogram.
\zxhlp{UVPLT}	plots data from a UV data base
\zxhlp{UVPRM}	measures parameters from a UV data base
\zxhlp{VLBACRPL}	Plots crosscorrelations
\zxhlp{VLBASNPL}	Plots selected contents of SN or CL files
\zxhlp{VPLOT}	plots uv data and model from CC file
\zxhlp{WETHR}	Plots selected contents of WX tables, flags data based on WX
\zxhlp{WIPER}	plots and edits data from a UV data base using the TV
\zxhlp{XAXIS}	Which parameter is plotted on the horizontal axis.
\zxhlp{XBASL}	Fits and subtracts nth-order baselines from cube (x axis)
\zxhlp{XG2PL}	Plots spectrum of a pixel with \zxhlp{XGAUS}/\zxhlp{AGAUS} and \zxhlp{ZEMAN}/\zxhlp{ZAMAN} f
\zxhlp{XGAUS}	Fits 1-dimensional Gaussians to images: restartable
\zxhlp{XPLOT}	Plots image rows one at a time on the graphics or TV screen
\zxhlp{ZAMAN}	Fits 1-dimensional Zeeman model to absorption-line data
\zxhlp{ZEMAN}	Fits 1-dimensional Zeeman model to data

13.26 POLARIZA

```
\zxhlp{AFARS}           Is used after \zxhlp{FARS} to determine Position and Value of the maximum
\zxhlp{BANDPOL}         specifies polarizations of individual IFs
\zxhlp{BDEPO}           computes depolarization due to rotation measure gradients
\zxhlp{COMB}             combines two images by a variety of mathematical methods
\zxhlp{DOFARS}           Procedure to aid in Faraday rotation synthesis using the \zxhlp{FARS} task
\zxhlp{DOPOL}            selects application of any polarization calibration
\zxhlp{FARAD}            add ionospheric Faraday rotation to CL table
\zxhlp{FARS}              Faraday rotation synthesys based on the brightness vs wavelength
HLPRMFIT    Polarization fitting task \zxhlp{RMFIT} - run-time help
HLPZAMAN    Fits 1-dimensional Zeeman model to absorption data - run-time help
HLPZEMAN    Fits 1-dimensional Zeeman model to data - run-time help
\zxhlp{LPCAL}            Determines instrumental polarization for UV data
\zxhlp{MAPBM}            Map \zxhlp{VLA} beam polarization
\zxhlp{MEDI}              combines four images by a variety of mathematical methods
\zxhlp{MODAB}            Makes simple absorption/emission spectral-line image in I/V
\zxhlp{MODIM}            adds images of model objects to image cubes in IQU polarization
\zxhlp{MODSP}            adds images of model objects to image cubes in I/V polarization
\zxhlp{PCAL}              Determines instrumental polarization for UV data
\zxhlp{PCNTR}            Generate plot file with contours plus polarization vectors
\zxhlp{PCUT}              Cutoff in polarized intensity
\zxhlp{PDVER}            specifies the version of the spectral polarization table to use
\zxhlp{PMODEL}            Polarization model parameters
\zxhlp{POLANGLE}          Intrinsic polarization angles for up to 30 sources
\zxhlp{POLCO}             Task to correct polarization maps for Ricean bias
\zxhlp{POLPLOT}           specifies the desired polarization ratio before plotting.
\zxhlp{QUFIX}             determines Right minus Left phase difference, corrects cal files
\zxhlp{QUOUT}             writes text file of Q, U versus frequency to be used by \zxhlp{RLDIF}
\zxhlp{QUXTR}              extracts text files from Q,U cubes for input to \zxhlp{TARS}
\zxhlp{RFARS}              Correct Q/U cubes for Faraday rotation synthesis results
\zxhlp{RLCAL}              Determines instrumental right-left phase versus time (a self-cal)
\zxhlp{RLCOR}              corrects a data set for R-L phase differences
\zxhlp{RLDIF}              determines Right minus Left phase difference, corrects cal files
\zxhlp{RM2PL}              Plots spectrum of a pixel with \zxhlp{RMFIT} fit
\zxhlp{RMFIT}              Fits 1-dimensional polarization spectrum to Q/U cube
\zxhlp{RM}                 Task to calculate rotation measure and magnetic field
\zxhlp{SWPOL}              Swap polarizations in a UV data base
\zxhlp{TARPL}              Plot output of \zxhlp{TARS} task
\zxhlp{TARS}               Simulation of Faraday rotation synthesis (mainly task \zxhlp{FARS})
\zxhlp{TRUEP}              determines true antenna polarization from special data sets
\zxhlp{VLABP}              \zxhlp{VLA} antenna beam polarization correction for snapshot images
\zxhlp{VLBACPOL}           Procedure to calibrate cross-polarization delays
\zxhlp{XG2PL}              Plots spectrum of a pixel with \zxhlp{XGAUS}/\zxhlp{AGAUS} and \zxhlp{ZEMAN}/\zxhlp{ZAMAN} fits
\zxhlp{ZAMAN}              Fits 1-dimensional Zeeman model to absorption-line data
\zxhlp{ZEMAN}              Fits 1-dimensional Zeeman model to data
```

13.27 POPS

```
\zxhlp{ABOUT}             displays lists and information on tasks, verbs, adverbs
\zxhlp{ABS}                returns absolute value of argument
\zxhlp{ACOS}               Returns arc cosine of argument (half-circle)
\zxhlp{APROPOS}            displays all help 1-line summaries containing specified words
\zxhlp{ARRAY1}              General scratch array adverb
\zxhlp{ARRAY2}              General scratch array adverb
\zxhlp{ARRAY3}              General scratch array adverb
\zxhlp{ARRAY}               Declares POPS symbol name and dimensions
\zxhlp{ASIN}                Returns arc sine of argument (half-circle)
```

13.27. POPS

13. Current AIPS Software

```

\zxhlp{ATAN2}          Returns arc tangent of two arguments (full circle)
\zxhlp{ATAN}           Returns arc tangent of argument (half-circle)
\zxhlp{BY}              gives increment to use in \zxhlp{FOR} loops in POPS language
\zxhlp{CATNO}          Specifies AIPS catalog slot number range
\zxhlp{CEIL}            returns smallest integer greater than or equal the argument
\zxhlp{CHAR}            converts number to character string
\zxhlp{CLRTEMP}         clears the temporary literal area during a procedure
\zxhlp{COMPRESS}        recovers unused POPS address space and new symbols
\zxhlp{CORE}            displays the used and total space used by parts of POPS table
\zxhlp{COS}              returns cosine of the argument in degrees
\zxhlp{DEBUG}            turns on/off the POPS-language's debug messages
\zxhlp{DEFAULT}          Verb-like sets adverbs for a task or verb to initial values
\zxhlp{DELAY}            Verb to pause AIPS for \zxhlp{DETIME} seconds
\zxhlp{DELTAX}           Increment or size in X direction
\zxhlp{DELTAY}           Increment or size in Y direction
\zxhlp{DENUMB}           a scalar decimal number
\zxhlp{DOOSRO}          calibrating amplitude and phase, and imaging \zxhlp{VLA} data
\zxhlp{DOVLAMP}          Produces amp calibration file for phased-\zxhlp{VLA} VLBI data
\zxhlp{DPARM}            General numeric array adverb used many places
\zxhlp{DUMP}              displays portions of the POPS symbol table in all formats
\zxhlp{EDIT}              enter edit-a-procedure mode in the POPS language
\zxhlp{EHEX}              converts decimal to extended hex
\zxhlp{EHNUMB}           an extended hexadecimal "number"
\zxhlp{ELSE}              starts POPS code done if an IF condition is false (IF-\zxhlp{THEN}...)
\zxhlp{ENDEDIT}           terminates procedure edit mode of POPS input
\zxhlp{END}                marks end of block (\zxhlp{FOR}, \zxhlp{WHILE}, IF) of POPS code
\zxhlp{ERASE}              removes one or more lines from a POPS procedure
\zxhlp{ERROR}             was there an error
\zxhlp{EVLA}               puts the list of eVLA antennas in the current file on stack
\zxhlp{EXIT}               ends an AIPS batch or interactive session
\zxhlp{EXP}                 returns the exponential of the argument
\zxhlp{EXPLAIN}            displays help + extended information describing a task/symbol
\zxhlp{FINISH}             terminates the entry and compilation of a procedure
\zxhlp{FLOOR}              returns largest integer <= argument
\zxhlp{FOR}                  starts an iterative sequence of operations in POPS language
\zxhlp{GETDATE}             Convert the current date and time to a string
\zxhlp{GET}                  restores previously SAVED full POPS environment
\zxhlp{GETPOPSN}            Verb to return the pops number on the stack
\zxhlp{GG}                  spare scalar adverb for use in procedures
\zxhlp{RANDOM}              Finds a random number with mean 0 and rms 1
\zxhlp{HELP}                 displays information on tasks, verbs, adverbs
\zxhlp{HSA}                  puts the list of \zxhlp{HSA} antennas in the current file on stack
IF      causes conditional execution of a set of POPS statements
I       spare scalar adverb for use in procedures
\zxhlp{INP}                  displays adverb values for task, verb, or proc - quick form
\zxhlp{INPUTS}               displays adverb values for task, verb, or proc - to msg file
\zxhlp{ISBATCH}              declares current AIPS to be, or not to be, batch-like
J       spare scalar adverb for use in procedures
\zxhlp{KLEENEX}              ends an AIPS interactive session wiping the slate klean
\zxhlp{LENGTH}               returns length of string to last non-blank character
\zxhlp{LIST}                  displays the source code text for a POPS procedure
\zxhlp{LN}                     returns the natural logarithm of the argument
\zxhlp{LOG}                     returns the base-10 logarithm of the argument
\zxhlp{MAX}                     returns the maximum of its two arguments
\zxhlp{MIN}                     returns the minimum of its two arguments
\zxhlp{MOD}                     returns remainder after division of 1st argument by 2nd
\zxhlp{MODIFY}                modifies the text of a line of a procedure and recompiles
\zxhlp{MODULUS}               returns square root of sum of squares of its two arguments

```

\zxhlp{NOADVERB}	Information about the lack of a defined adverb or verb
\zxhlp{NUMTELL}	selects POPS number of task which is the target of a \zxhlp{TELL}
\zxhlp{NX}	General adverb referring to a number of things in the Y direction
\zxhlp{NY}	General adverb referring to a number of things in the Y direction
\zxhlp{OUTPUTS}	displays adverb values returned from task, verb, or proc
\zxhlp{PARALLEL}	Verb to set or show degree of parallelism
\zxhlp{PASSWORD}	Verb to change the current password for the login user
\zxhlp{PCAT}	Verb to list entries in the user's catalog (no log file).
\zxhlp{PIPEAIPS}	calibrating amplitude and phase, and imaging \zxhlp{VLA} data
\zxhlp{POPSDAT}	lists all POPS symbols, used to create them in MEmory files
\zxhlp{POPSYM}	Describes the symbols used in POPS
\zxhlp{PRINTER}	Verb to set or show the printer(s) used
\zxhlp{PRINT}	Print the value of an expression
\zxhlp{PROCEDUR}	Define a POPS procedure using procedure editor
\zxhlp{PROC}	Define a POPS procedure using procedure editor.
\zxhlp{PSEUDO}	Description of POPS pseudoverbs - obsolete list file
\zxhlp{PSEUDOVB}	Declares a name to be a symbol of type pseudoverb
\zxhlp{QINP}	displays adverb values for task, verb, or proc - restart form
\zxhlp{RANDOM}	Compute a random number from 0 to 1
\zxhlp{READ}	Read a value from the users terminal
\zxhlp{REHEX}	converts extended hex string to decimal
\zxhlp{RENAME}	Rename a file (UV or Image)
\zxhlp{RESTART}	Verb to trim the message log file and restart AIPS
\zxhlp{RESTORE}	Read POPS memory file from a common area.
\zxhlp{RETURN}	Exit a procedure allowing a higher level proc to continue.
\zxhlp{RUN}	Pseudoverb to read an external \zxhlp{RUN} files into AIPS.
\zxhlp{SAVDEST}	Verb to destroy all save files of a user.
\zxhlp{SAVE}	Pseudoverb to save full POPS environment in named file
\zxhlp{SCALAR}	Declares a variable to be a scalar in a procedure
\zxhlp{SCANLENG}	specify length of "scan"
\zxhlp{SCRATCH}	delete a procedure from the symbol table.
\zxhlp{SETDEBUG}	Verb to set the debug print and execution level
\zxhlp{SG2RUN}	Verb copies the K area to a text file suitable for \zxhlp{RUN}
\zxhlp{SGDESTR}	Verb-like to destroy named POPS environment save file
\zxhlp{SGINDEX}	Verb lists \zxhlp{SAVE} areas by name and time of last \zxhlp{SAVE}.
\zxhlp{SIN}	Compute the sine of a value
\zxhlp{SLOT}	Specifies AIPS catalog slot number
\zxhlp{SPY}	Verb to determine the execution status of all AIPS tasks
\zxhlp{SQRT}	Square root function
\zxhlp{STORE}	Store current POPS environment
\zxhlp{STQUEUE}	Verb to list pending \zxhlp{TELL} operations
\zxhlp{STRING}	Declare a symbol to be a string variable in POPS
\zxhlp{SUBMIT}	Verb which submits a batch work file to the job queue
\zxhlp{SUBSTR}	Function verb to specify a portion of a \zxhlp{STRING} variable
\zxhlp{T1VERB}	Temporary verb for testing (also T2VERB...T9VERB)
\zxhlp{TAN}	Tangent function
\zxhlp{TAPES}	Verb to show the \zxhlp{TAPES}(s) available
\zxhlp{TASK}	Name of a task
\zxhlp{TGET}	Verb-like gets adverbs from last \zxhlp{GO} of a task
\zxhlp{TGINDEX}	Verb lists those tasks for which \zxhlp{TGET} will work.
\zxhlp{THEDATE}	contains the date and time in a string form
\zxhlp{THEN}	Specified the action if an IF test is true
\zxhlp{TIMDEST}	Verb to destroy all files which are too old
\zxhlp{TO}	Specifies upper limit of a \zxhlp{FOR} loop
\zxhlp{TPUT}	Verb-like puts adverbs from a task in file for TGETs
\zxhlp{TYPE}	Type the value of an expression
\zxhlp{USAVE}	Pseudoverb to save full POPS environment in named file
\zxhlp{VALUE}	Convert a string to a numeric value

13.28. PROCEDUR

13. Current AIPS Software

```

\zxhlp{VERB}      Declares a name to be a symbol of type verb
\zxhlp{VERSION}   Specify AIPS version or local task area
\zxhlp{VGET}       Verb-like gets adverbs from version task parameter save area
\zxhlp{VGINDEX}   Verb lists those tasks for which \zxhlp{VGET} will work.
\zxhlp{VLA}        puts the list of \zxhlp{VLA} antennas in the current file on stack
\zxhlp{VLARUN}    calibrating amplitude and phase, and imaging \zxhlp{VLA} data
VLBA             puts the list of VLBA antennas in the current file on stack
\zxhlp{VLBAPIPE}  applies amplitude and phase calibration procs to VLBA data
\zxhlp{VLBARUN}   applies amplitude and phase calibration procs to VLBA data
\zxhlp{VNUMBER}   Specifies the task parameter (\zxhlp{VGET}/\zxhlp{VPUT}) save area
\zxhlp{VPUT}       Verb-like puts adverbs from a task in files for VGETs
\zxhlp{WAITTASK}  halt AIPS until specified task is finished
\zxhlp{WHILE}     Start a conditional statement
\zxhlp{XHELP}     Accesses hypertext help system
X                spare scalar adverb for use in procedures
Y                spare scalar adverb for use in procedures

```

13.28 PROCEDUR

```

\zxhlp{ANTNUM}    Returns number of a named antenna
\zxhlp{BASFIT}    fits antenna locations from SN-table data
\zxhlp{BOX2CC}    Converts \zxhlp{CLBOX} in pixels to \zxhlp{CCBOX} in arc seconds
\zxhlp{BREAK}     procedure to \zxhlp{TELL} \zxhlp{FILLM} to break all current uv files, start new
\zxhlp{CIRCLEVS}  Sets \zxhlp{RGBLEVS} to fill \zxhlp{LEVS} with a circular color scheme
\zxhlp{COTVLOD}  Proc to load an image into a TV channel about a coordinate
\zxhlp{CROSSPOL} Procedure to make complex poln. images and beam.
\zxhlp{CRSFRING} Procedure to calibrate cross pol. delay and phase offsets
\zxhlp{CXPOLN}   Procedure to make complex poln. images and beam.
\zxhlp{DEFCOLOR}  Sets adverb \zxhlp{PLCOLORS} to match s default \zxhlp{XAS} TV
\zxhlp{DOFARS}   Procedure to aid in Faraday rotation synthesis using the \zxhlp{FARS} task
\zxhlp{FEW}       procedure to \zxhlp{TELL} \zxhlp{FILLM} to append incoming data to existing uv files
\zxhlp{FITDISK}  writes images / uv data w extensions to disk in FITS format
\zxhlp{FLAMLEVS} Sets \zxhlp{RGBLEVS} to fill \zxhlp{LEVS} with a red flame color scheme
\zxhlp{FXALIAS}  least squares fit aliasing function and remove
\zxhlp{FXAVG}    Procedure to enable VLBA delay de-correlation corrections
\zxhlp{GRANDOM}  Finds a random number with mean 0 and rms 1
\zxhlp{IMFRING}  large image delay fitting with \zxhlp{IM2CC} and \zxhlp{OOFRING}
\zxhlp{IMSCAL}   large image self-cal with \zxhlp{IM2CC} and \zxhlp{OOCAL}
\zxhlp{LINIMAGE} Build image cube from multi-IF data set
\zxhlp{MANY}     procedure to \zxhlp{TELL} \zxhlp{FILLM} to start new uv files on each scan
\zxhlp{MAPPR}    Simplified access to \zxhlp{IMAGR}
\zxhlp{MAXTAB}   Returns maximum version number of named table
\zxhlp{MERGECAL} Procedure to merge calibration records after concatenation
\zxhlp{OFFROAM}  Procedure to clear the TV from a Roam condition
\zxhlp{OOCAL}    determines antenna complex gain with frequency-dependent models
\zxhlp{OOFRING}  fringe fit data to determine antenna calibration, delay, rate
\zxhlp{PEELR}    calibrates interfering sources in multi-facet imges
\zxhlp{PFT}      The Perley-Feigelson Test; see PFTLOAD.\zxhlp{RUN}, PFTEXEC.\zxhlp{RUN}
\zxhlp{QUIT}     procedure to \zxhlp{TELL} \zxhlp{FILLM} to stop at the end of the current scan
\zxhlp{RAINLEVS} Sets \zxhlp{RGBLEVS} to fill \zxhlp{LEVS} with a rainbow color scheme
\zxhlp{READISK}  writes images / uv data w extensions to tape in FITS format
\zxhlp{RUNWAIT}  Runs a task and waits for it to finish
\zxhlp{SCANTIME} Returns time range for a given scan number
\zxhlp{SETXWIN}  Procedure to set \zxhlp{BLC} and \zxhlp{TRC} with TV cursor
\zxhlp{STEPLEVS} Sets \zxhlp{RGBLEVS} to fill \zxhlp{LEVS} with a repeated sequence of colors
\zxhlp{STOP}     procedure to \zxhlp{TELL} \zxhlp{FILLM} to break all current uv files and stop
\zxhlp{STUFFR}   averages together data sets in hour angle

```

```
\zxhlp{TDEPEND} Time-dependent imaging procedure sequence
TD_SCANS Time-dependent imaging procedure sequence: find intervals
TD_SSCAN Time-dependent imaging procedure sequence: find intervals
TD_STEP3 Time-dependent imaging procedure "step 3"
TD_STEP5 Time-dependent imaging procedure sequence: later steps
\zxhlp{TEFLM} procedure to \zxhlp{TELL} real-time \zxhlp{FILLM} a new \zxhlp{APARM}(1) value
\zxhlp{TKBOX} Procedure to set a Clean box with the TK cursor
\zxhlp{TKNBOXS} Procedure to set Clean boxes 1 - n with the TK cursor
\zxhlp{TKWIN} Procedure to set \zxhlp{BLC} and \zxhlp{TRC} with Graphics cursor
\zxhlp{TVALL} Procedure loads image to TV, shows labeled wedge, enhances
\zxhlp{TVCOLORS} Sets adverb \zxhlp{PLCOLORS} to match the TV (\zxhlp{DOTV}=1) usage
\zxhlp{TVDIST} determines spherical distance between two pixels on TV screen
\zxhlp{TVFLUX} displays coordinates and values selected with the TV cursor
\zxhlp{TVMAXFIT} displays fit pixel positions and intensity at maxima on TV
\zxhlp{TVRESET} Reset the TV without erasing the image planes
\zxhlp{VLACALIB} Runs \zxhlp{CALIB} and \zxhlp{LISTR} for \zxhlp{VLA} observation
\zxhlp{VLACLCAL} Runs \zxhlp{CLCAL} and prints the results with \zxhlp{LISTR}
\zxhlp{VLALIST} Runs \zxhlp{LISTR} for \zxhlp{VLA} observation
\zxhlp{VLARESET} Reset calibration tables to a virginal state
\zxhlp{VLASUMM} Plots selected contents of SN or CL files
\zxhlp{VLATECR} Calculate ionospheric delay and Faraday rotation corrections
\zxhlp{VLBAAMP} applies a-priori amplitude corrections to VLBA data
\zxhlp{VLBACALA} applies a-priori amplitude corrections to VLBA data
\zxhlp{VLBACCOR} applies a-priori amplitude corrections to VLBA data
\zxhlp{VLBACPOL} Procedure to calibrate cross-polarization delays
\zxhlp{VLBACRPL} Plots crosscorrelations
\zxhlp{VLBAEOPS} Corrects Earth orientation parameters
\zxhlp{VLBAFIX} Procedure that fixes VLBA data, if necessary
\zxhlp{VLBAFPOL} Checks and corrects polarization labels for VLBA data
\zxhlp{VLBAFQGS} Copies different FQIDS to separate files
\zxhlp{VLBAFRGP} Fringe fit phase referenced data and apply calibration
\zxhlp{VLBAFRNG} Fringe fit data and apply calibration
\zxhlp{VLBAIT} Procedure to read and process VLBA data (Phil Diamond)
\zxhlp{VLBAKRG} Fringe fit phase referenced data and apply calibration
\zxhlp{VLBAKRNG} Fringe fit data and apply calibration
\zxhlp{VLBALOAD} Loads VLBA data
\zxhlp{VLBAMCAL} Merges redundant calibration data
\zxhlp{VLBAMPCL} Calculates and applies manual instrumental phase calibration
\zxhlp{VLBAPANG} Corrects for parallactic angle
\zxhlp{VLBAPCOR} Calculates and applies instrumental phase calibration
\zxhlp{VLBASNPL} Plots selected contents of SN or CL files
\zxhlp{VLBASRT} Sorts VLBA data, if necessary
\zxhlp{VLBASUBS} looks for subarrays in VLBA data
\zxhlp{VLBASUMM} Prints a summary of a VLBI experiment
\zxhlp{VLBATECR} Calculate ionospheric delay and Faraday rotation corrections
\zxhlp{WRTDISK} writes images / uv data w extensions to tape in FITS format
```

13.29 PSEUDOVE

Type: General type of POPS symbol

Use: Pseudoverbs are magic symbols which cause FORTRAN routines to carry out specific actions. Unlike verbs, pseudoverbs are executed as soon as they are encountered by the compiler even in compile mode. In general, the FORTRAN routines which are invoked will parse the remainder of the input line under special, non-standard rules. Any normal code typed on the line ahead of the pseudoverb will not be executed.

Grammar: See the \zxhlp{HELP} listings for the specific pseudoverb.

Examples: \zxhlp{HELP} \zxhlp{HELP}
\zxhlp{ARRAY} JUNK(4, -7 \zxhlp{T0} 9)
\zxhlp{PROC} DUMMY (I,J)
\zxhlp{LIST} DUMMY
\zxhlp{DEBUG} \zxhlp{TRUE}
\zxhlp{INPUTS} MLOAD

\zxhlp{ABORTASK}	stops a running task
\zxhlp{ARRAY}	Declares POPS symbol name and dimensions
\zxhlp{COMPRESS}	recovers unused POPS address space and new symbols
\zxhlp{CORE}	displays the used and total space used by parts of POPS table
\zxhlp{DEBUG}	turns on/off the POPS-language's debug messages
\zxhlp{EDIT}	enter edit-a-procedure mode in the POPS language
\zxhlp{ELSE}	starts POPS code done if an IF condition is false (IF-\zxhlp{THEN}..)
\zxhlp{ENDBATCH}	terminates input to batch work file
\zxhlp{ENDEDIT}	terminates procedure edit mode of POPS input
\zxhlp{ERASE}	removes one or more lines from a POPS procedure
\zxhlp{FINISH}	terminates the entry and compilation of a procedure
\zxhlp{GET}	restores previously SAVED full POPS environment
IF causes	conditional execution of a set of POPS statements
\zxhlp{ISBATCH}	declares current AIPS to be, or not to be, batch-like
\zxhlp{LIST}	displays the source code text for a POPS procedure
\zxhlp{MODIFY}	modifies the text of a line of a procedure and recompiles
\zxhlp{MSGKILL}	turns on/off the recording of messages in the message file
\zxhlp{TELL}	Send parameters to tasks that know to read them on the fly
\zxhlp{WHILE}	Start a conditional statement

13.30 RUN

\zxhlp{ANCHECK}	Checks By sign in Antenna files
\zxhlp{DDT}	verifies correctness and performance using standard problems
\zxhlp{DDTSAVE}	verifies correctness and performance using standard problems
\zxhlp{DOOSRO}	calibrating amplitude and phase, and imaging \zxhlp{VLA} data
\zxhlp{DOVLAMP}	Produces amp calibration file for phased-\zxhlp{VLA} VLBI data
\zxhlp{HYB}	\zxhlp{RUN} to set parameters for HYBRID (\zxhlp{CALIB}/\zxhlp{MX}) self-cal imaging
\zxhlp{PIPEAIPS}	calibrating amplitude and phase, and imaging \zxhlp{VLA} data
\zxhlp{VLAC}	verifies correctness of continuum calibration software
\zxhlp{VLACSAVE}	verifies correctness of continuum calibration
\zxhlp{VLAL}	verifies correctness of spectral line calibration software
\zxhlp{VLALSAVE}	verifies correctness of continuum calibration
\zxhlp{VLAPROCS}	Procedures to simplify the reduction of VLBA data
\zxhlp{VLARUN}	calibrating amplitude and phase, and imaging \zxhlp{VLA} data
\zxhlp{VLBAARCH}	Procedure to archive VLBA correlator data
\zxhlp{VLBAPIPE}	applies amplitude and phase calibration procs to VLBA data
\zxhlp{VLBARUN}	applies amplitude and phase calibration procs to VLBA data
\zxhlp{VLBAUTIL}	Procedures to simplify the reduction of VLBA data

\zxhlp{VLBDDT}	Verification tests using simulated data
\zxhlp{WRTPROCS}	Procedures to simplify the reduction of VLBA data
\zxhlp{Y2K}	verifies correctness and performance using standard problems
\zxhlp{Y2KSAVE}	verifies correctness and performance using standard problems

13.31 SINGLEDI

\zxhlp{BSAVG}	Task to do an \zxhlp{FFT}-weighted sum of beam-switched images
\zxhlp{BSCLN}	Hogbom Clean on beam-switched difference image
\zxhlp{BSCOR}	Combines two beam-switched images
\zxhlp{BSFIX}	Corrects the ra/dec offsets recorded by the 12m
\zxhlp{BSGRD}	Task to image beam-switched single-dish data
\zxhlp{BSMOD}	creates single-dish UV beam-switched data with model sources
\zxhlp{BSROT}	modifies SD beam-switch continuum data for error in throw
\zxhlp{BSTST}	Graphical display of solutions to frequency-switched data
\zxhlp{CSCOR}	applies specified corrections to CS tables
\zxhlp{DATA2IN}	specifies name of input FITS disk file
\zxhlp{DATAIN}	specifies name of input FITS disk file
\zxhlp{DIFUV}	Outputs the difference of two matching input uv data sets
\zxhlp{INDXH}	writes index file describing contents of UV data base
\zxhlp{INDXR}	writes index file describing contents of UV data base
\zxhlp{OTFBS}	Translates on-the-fly continuum SDD format to AIPS UV file
\zxhlp{OTFIN}	Lists on-the-fly single-dish SDD format data files
\zxhlp{OTFUV}	Translates on-the-fly single-dish SDD format to AIPS UV file
\zxhlp{PRTSD}	prints contents of AIPS single-dish data sets
\zxhlp{SDCAL}	Task to apply single dish calibration
\zxhlp{SDGRD}	Task to select and image random-position single-dish data
\zxhlp{SDIMG}	Task to select and image random-position single-dish data
\zxhlp{SDLSF}	least squares fit to channels and subtracts from SD uv data
\zxhlp{SDMOD}	modifies single-dish UV data with model sources
\zxhlp{SDTUV}	Task to convert SD table files to UV like data.
\zxhlp{SDVEL}	shifts spectral-line single-dish data to a given velocity
\zxhlp{VTEST}	Measures velocity discrepancy across fields
\zxhlp{WTSUM}	Task to do a a sum of images weighted by other images

13.32 SPECTRAL

Almost all parts of AIPS are general enough to handle multiple dimensions of data including multiple frequency channels in the uv domain and 3 or more dimensional "cubes" in the image domain.

\zxhlp{ACFIT}	Determine antenna gains from autocorrelations
\zxhlp{AGAUS}	Fits 1-dimensional Gaussians to absorption-line spectra
\zxhlp{ALTDEF}	Sets frequency vs velocity relationship into image header
\zxhlp{ALTSWTCH}	Switches between frequency and velocity in image header
\zxhlp{AVSPC}	Averages uv-data in the frequency domain
\zxhlp{BASRM}	Task to remove a spectral baseline from total power spectra
\zxhlp{BCHAN}	sets the beginning channel number
\zxhlp{BDAPL}	Applies a BD table to another data set
\zxhlp{BLOAT}	converts line data to greater number channels
\zxhlp{BLSUM}	sums images over irregular sub-images, displays spectra
\zxhlp{BPASS}	computes spectral bandpass correction table
\zxhlp{BPASSPRM}	Control adverb array for bandpass calibration
\zxhlp{BPERR}	Print and plot \zxhlp{BPASS} closure outputs
\zxhlp{BPLOT}	Plots bandpass tables in 2 dimensions as function of time
\zxhlp{BPSMO}	Smooths or interpolates bandpass tables to regular times
\zxhlp{BPVER}	specifies the version of the bandpass table to be applied
\zxhlp{CHANNEL}	sets the spectral channel number
\zxhlp{CHANSEL}	Array of start, stop, increment channel numbers to average the increment between selected channels
\zxhlp{CHINC}	computes polynomial spectral bandpass correction table
\zxhlp{CPASS}	Model a galaxy's density and velocity distribution from full cube
\zxhlp{CUBIT}	shifts spectral-line UV data to a given velocity
\zxhlp{CVEL}	Array of start, stop, increment channel #S + IF to avoid
\zxhlp{DCHANSEL}	Flag to indicate whether an operation is applied to the data
\zxhlp{DOAPPLY}	define an end for a range of channel numbers
\zxhlp{ECHAN}	least squares fit aliasing function and remove
\zxhlp{FIXAL}	flags data based on the rms of the spectrum
\zxhlp{FLGIT}	collects n-dimensional images into n+1-dimensional \zxhlp{FREQID} image
\zxhlp{FQUBE}	Task to build a map using fringe rate spectra
\zxhlp{FRMAP}	Task to plot fringe rate spectra
\zxhlp{FRPLT}	interactive flagging of UV data in channel-time using the TV
\zxhlp{FTFLG}	least squares fit aliasing function and remove
\zxhlp{FXALIAS}	
HLPFTFLG	Interactive time-channel visibility Editor - run-time help
HLPPCFLG	Interactive time-channel PC table Editor \zxhlp{PCFLG} - run-time help
HLPSPFLG	Interactive time-channel visibility Editor \zxhlp{SPFLG} - run-time help
HLPTVHUI	Interactive intensity-hue-saturation display - run-time help
HLPTVRGB	Interactive red-green-blue display - run-time help
HLPZAMAN	Fits 1-dimensional Zeeman model to absorption data - run-time help
HLPZEMAN	Fits 1-dimensional Zeeman model to data - run-time help
\zxhlp{ICHANSEL}	Array of start, stop, increment channel #S + IF to average
\zxhlp{IMLIN}	Fits and removes continuum emission from cube
\zxhlp{IRING}	integrates intensity / flux in rings / ellipses
\zxhlp{ISPEC}	Plots and prints spectrum of region of a cube
\zxhlp{MCUBE}	collects n-dimensional images into n+1-dimensional image
\zxhlp{MODAB}	Makes simple absorption/emission spectral-line image in I/V
\zxhlp{MORIF}	Combines IFs or breaks spectral windows into multiple windows (IFs)
\zxhlp{NCHAV}	Number of channels averaged in an operation
\zxhlp{ORDER}	Adverb used usually to specify the order of polynomial fit
\zxhlp{PCASS}	Finds amplitude bandpass shape from pulse-cal table data
\zxhlp{PCFIT}	Finds delays and phases using a PC table
\zxhlp{PCPLT}	Plots pulse-cal tables in 2 dimensions as function of time
\zxhlp{PCVEL}	shifts spectral-line UV data to a given velocity: planet version
\zxhlp{PDVER}	specifies the version of the spetral polarization table to use
\zxhlp{PLCUB}	Task to plot intensity vs x panels on grid of y,z pixels

```
\zxhlp{POLANGLE} Intrinsic polarization angles for up to 30 sources
\zxhlp{POSSM} Task to plot total and cross-power spectra.
\zxhlp{QUFIX} determines Right minus Left phase difference, corrects cal files
\zxhlp{REIFS} Breaks spectral windows into multiple spectral windows (IFs)
\zxhlp{RM2PL} Plots spectrum of a pixel with \zxhlp{RMFIT} fit
\zxhlp{RMFIT} Fits 1-dimensional polarization spectrum to Q/U cube
\zxhlp{RSPEC} Plots and prints spectrum of rms of a cube
\zxhlp{SDLSF} least squares fit to channels and subtracts from SD uv data
\zxhlp{SDVEL} shifts spectral-line single-dish data to a given velocity
\zxhlp{SERCH} Finds line signals in transposed data cube
\zxhlp{SMOTH} Task to smooth a subimage from upto a 7-dim. image
\zxhlp{SPECIDX} Spectral index used to correct calibrations
\zxhlp{SPECPARM} Spectral index per polarization per source
\zxhlp{SPECTRAL} Flag to indicate whether an operation is spectral or continuum
\zxhlp{SPECURVE} Spectral index curvature used to correct calibrations
\zxhlp{SPFLG} interactive flagging of UV data in channel-TB using the TV
\zxhlp{SPMOD} Modify UV database by adding a model with spectral lines
\zxhlp{SQASH} Task to collapse several planes in a cube into one plane or row
\zxhlp{SYSVEL} Systemic velocity
\zxhlp{UJOIN} modifies UV data converting IFs to spectral channels
\zxhlp{UV2TB} Converts UV autocorrelation spectra to tables
\zxhlp{UVBAS} averages several channels and subtracts from uv data.
\zxhlp{UVDEC} Decrement the number of spectral channels, keeping every nth
\zxhlp{UVFRE} Makes one data set have the spectral structure of another
\zxhlp{UVGLU} Glues UV data frequency blocks back together
\zxhlp{UVLIN} Fits and removes continuum visibility spectrum, also can flag
\zxhlp{UVLSD} least squares fit to channels and divides the uv data.
\zxhlp{UVLSF} least squares fit to channels and subtracts from uv data.
\zxhlp{UVMLN} edits data based on the rms of the spectrum
\zxhlp{UVMOD} Modify UV database by adding a model incl spectral index
\zxhlp{VBGLU} Glues together data from multiple passes thru the VLBA corr.
\zxhlp{VLBABPSS} computes spectral bandpass correction table
\zxhlp{VTEST} Measures velocity discrepancy across fields
\zxhlp{WTSUM} Task to do a a sum of images weighted by other images
\zxhlp{XBASL} Fits and subtracts nth-order baselines from cube (x axis)
\zxhlp{XG2PL} Plots spectrum of a pixel with \zxhlp{XGAUS}/\zxhlp{AGAUS} and \zxhlp{ZEMAN}/\zxhlp{ZAMAN} fits
\zxhlp{XGAUS} Fits 1-dimensional Gaussians to images: restartable
\zxhlp{XPLOT} Plots image rows one at a time on the graphics or TV screen
\zxhlp{ZAMAN} Fits 1-dimensional Zeeman model to absorption-line data
\zxhlp{ZEMAN} Fits 1-dimensional Zeeman model to data
```

13.33 TABLE

```
\zxhlp{BSPRT} print BS tables
\zxhlp{CLCOP} copy CL/SN file calibration between polarizations or IFs
\zxhlp{CLINV} copy CL/SN file inverting the calibration
\zxhlp{DOTABLE} selects use of table-format for data
\zxhlp{EXTAB} exports AIPS table data as tab-separated text
\zxhlp{HF2SV} convert HF tables from \zxhlp{FRING}/\zxhlp{MBDLY} to form used by Calc/Solve
\zxhlp{HFPRT} write HF tables from \zxhlp{CL2HF}
HLPEDIBP Interactive BP table uv-data editor \zxhlp{BPEDT} - run-time help
HLPEDICL Interactive SN/CL table uv-data editor - run-time help
HLPEDIPC Interactive PC table editor \zxhlp{PCEDT} - run-time help
HLPEDISN Interactive SN/CL table (not UV) editor - run-time help
HLPEDISS Interactive SY table (not UV) editor - run-time help
HLPEDISY Interactive SY table uv-data editor \zxhlp{EDITA} - run-time help
HLPEDITIS Interactive TY table (not UV) editor - run-time help
```

```

HLPEDITY Interactive TY table uv-data editor \zxhlp{EDITA} - run-time help
\zxhlp{MFPRT} prints MF tables in a format needed by modelling software
\zxhlp{OBEDT} Task to flag data of orbiting antennas
\zxhlp{OBTAB} Recalculate orbit parameters and other spacecraft info
\zxhlp{OFLAG} uses on-line flag table information to write a flag table
\zxhlp{PRTAB} prints any table-format extension file
\zxhlp{PRTOF} prints on-line flag table information
\zxhlp{SNCOP} Task to copy SN table averaging some input IFs
\zxhlp{TABED} Task to edit tables
\zxhlp{TACOP} task to copy tables, other extension files
\zxhlp{TAFLG} Flags data in a Table extension file
\zxhlp{TAMRG} Task to merge table rows under specified conditions
\zxhlp{TAPLT} Plots data from a Table extension file
\zxhlp{TAPPE} Task to append 2 tables and merge to output table
\zxhlp{TASAV} Task to copy all extension tables to a dummy uv or map file
\zxhlp{TASRT} Task to sort extension tables.
\zxhlp{TBDIF} Compare entries in two tables
\zxhlp{TBIN} Reads a text file AIPS table into AIPS
\zxhlp{TBOUT} Writes an AIPS table into a text file for user editting.
\zxhlp{TYCOP} copy TY or SY table calibration between IFs
\zxhlp{VLBAMCAL} Merges redundant calibration data

```

13.34 TAPE

```

\zxhlp{ATLOD} Reads ATCA data in RPFITS format into AIPS
\zxhlp{AVEOT} Advances tape to end-of-information point
\zxhlp{AVFILE} Moves tape forward or back to end-of-file marks
\zxhlp{AVMAP} Advance tape by one image (IBM-CV = obsolete tape file)
\zxhlp{AVTP} Positions tape to desired file
\zxhlp{BAKLD} reads all files of a catalog entry from \zxhlp{BAKTP} tape
\zxhlp{BAKTP} writes all files of a catalog entry to tape in host format
\zxhlp{BLOCKING} specifies blocking factor to use on e.g. tape records
\zxhlp{DENSITY} gives the desired tape density
\zxhlp{DISMOUNT} disables a magnetic tape and dismounts it from the tape drive
\zxhlp{DOEOF} selects end-of-file writing or reading until
\zxhlp{DOEOT} selects tape positioning before operation: present or EOI
\zxhlp{DONEWTAB} do we make new tables, use a new table format, etc.
\zxhlp{DOTABLE} selects use of table-format for data
\zxhlp{DOTWO} do we make two of something
\zxhlp{FILLM} reads \zxhlp{VLA} on-line/archive format uv data tapes (post Jan 88)
\zxhlp{FILLR} reads old \zxhlp{VLA} on-line-system tapes into AIPS
\zxhlp{FIT2A} reads the fits input file and records it to the output ascii file
\zxhlp{FITAB} writes images / uv data w extensions to tape in FITS format
\zxhlp{FITLD} Reads FITS files to load images or UV (IDI or UVFITS) data to disk
\zxhlp{FITTP} writes images / uv data w extensions to tape in FITS format
\zxhlp{FORMAT} gives a format code number: e.g. FITS accuracy required
\zxhlp{GSCAT} reads Fits Guide star catalog file
\zxhlp{IMLOD} reads tape to load images to disk
\zxhlp{INTAPE} specifies the input tape drive number
\zxhlp{MOUNT} makes a tape drive available to user's AIPS and tasks
\zxhlp{NFILES} The number of files to skip, usually on a tape.
\zxhlp{NPIECE} The number of pieces to make
\zxhlp{OUTTAPE} The output tape drive number.
\zxhlp{PRTTP} prints contents of tapes, all supported formats
\zxhlp{QUANTIZE} Quantization level to use
\zxhlp{REMHOST} gives the name of another computer which will provide service
\zxhlp{REMTAPE} gives the number of another computer's tape device

```

\zxhlp{REWIND}	Verb to rewind a tape
\zxhlp{TAPES}	Verb to show the \zxhlp{TAPES}(s) available
\zxhlp{TCOPY}	Tape to tape copy with some disk FITS support
\zxhlp{TPHEAD}	Verb to list image header from FITS or IBM-CV tape
\zxhlp{TPMON}	Information about the \zxhlp{TPMON} "Daemon"
\zxhlp{UVLOD}	Read export or FITS data from a tape or disk
\zxhlp{VLAMODE}	\zxhlp{VLA} observing mode
\zxhlp{VLAOBS}	Observing program or part of observer's name

13.35 TASK

TASKS

Type: General type of POPS symbol (not in symbol table)

Use: Tasks are separate programs which may be started by AIPS and which receive their input parameters from AIPS. In the interactive AIPS, tasks run asynchronously from AIPS. In the batch AIPS, the language processor waits for each task to finish before starting another one.

Grammar: \zxhlp{TASK} = 'name' ; \zxhlp{GO}
will cause the task whose name is assigned to the string adverb \zxhlp{TASK} to be started. Note: the name should have no leading blanks and should be no longer than 5 characters.

Alternative grammar: \zxhlp{GO} name ;
where name is the name of the task to be run.

Related adverbs:

\zxhlp{TASK}	Task name
\zxhlp{DOWAIT}	On "\zxhlp{GO}", wait for task completion before returning to AIPS control
\zxhlp{VERSION}	Version of task to be executed.
\zxhlp{QCREATE}	Create files by fast or more reliable methods

Related verbs:

\zxhlp{GO}	Initiate a shed task
\zxhlp{HELP}	List information about a task
\zxhlp{INP}	List adverb values for a task
\zxhlp{INPUTS}	Same as \zxhlp{INP} but also written to MSG file
\zxhlp{SPY}	Inquire which tasks are active
\zxhlp{WAITTASK}	Suspend AIPS operation until a specific task is complete
\zxhlp{ABORTASK}	Kill a task immediately
\zxhlp{TGET}	Get adverb values from last execution of \zxhlp{TASK}
\zxhlp{TPUT}	Save adverb values without execution of \zxhlp{TASK}
\zxhlp{TGINDEX}	List all \zxhlp{TGET}/\zxhlp{SAVE} files

\zxhlp{ACCOR}	Corrects cross amplitudes using auto correlation measurements
\zxhlp{ACFIT}	Determine antenna gains from autocorrelations
\zxhlp{ACLIP}	edits suto-corr data for amplitudes, phases, and weights out of range
\zxhlp{ACSCL}	Corrects cross amplitudes using auto correlation measurements
\zxhlp{ADDIF}	Adds an IF axis to a uv data set
\zxhlp{AFARS}	Is used after \zxhlp{FARS} to determine Position and Value of the maximum
\zxhlp{AFILE}	sorts and edits MkIII correlator A-file.
\zxhlp{AGAUS}	Fits 1-dimensional Gaussians to absorption-line spectra
\zxhlp{AHIST}	Task to convert image intensities by adaptive histogram
\zxhlp{AIPSB}	AIPS main program for executing batch jobs
\zxhlp{AIPSC}	AIPS main program for testing and queuing batch jobs

13.35. TASK

13. Current AIPS Software

AIPS	AIPS main program for interactive use
\zxhlp{ALVAR}	plots the Allan Variance statistic of a UV data set
\zxhlp{ALVPR}	prints statistics on the Allan Variance of a UV data set
\zxhlp{ANBPL}	plots and prints uv data converted to antenna based values
\zxhlp{ANCAL}	Places antenna-based Tsys and gain corrections in CL table
\zxhlp{ANTAB}	Read amplitude calibration information into AIPS
\zxhlp{APCAL}	Apply TY and GC tables to generate an SN table
\zxhlp{APCLN}	Deconvolves images with CLEAN algorithm
\zxhlp{APGPS}	Apply GPS-derived ionospheric corrections
\zxhlp{APGS}	deconvolves image with Gerchberg-Saxton algorithm
\zxhlp{APVC}	Deconvolves images with van Cittert algorithm
\zxhlp{ATLOD}	Reads ATCA data in RPFITS format into AIPS
\zxhlp{ATMCA}	Determines delay/phase gradient from calibrator observations
\zxhlp{AVER}	Averages over time UV data sets in 'BT' order
\zxhlp{AVSPC}	Averages uv-data in the frequency domain
\zxhlp{AVTP}	Positions tape to desired file
\zxhlp{BAKLD}	reads all files of a catalog entry from \zxhlp{BAKTP} tape
\zxhlp{BAKTP}	writes all files of a catalog entry to tape in host format
\zxhlp{BASRM}	Task to remove a spectral baseline from total power spectra
\zxhlp{BATER}	stand-alone program to prepare and submit batch jobs
\zxhlp{BDAPL}	Applies a BD table to another data set
\zxhlp{BDEPO}	computes depolarization due to rotation measure gradients
\zxhlp{BDF2AIPS}	Read \zxhlp{EVLA} \zxhlp{ASDM}/BDF data into AIPS
\zxhlp{BLANK}	blanks out selected, e.g. non-signal, portions of an image
\zxhlp{BLAPP}	applies baseline-based fringe solutions a la \zxhlp{BLAPP}
\zxhlp{BLAVG}	Average cross-polarized UV data over baselines.
\zxhlp{BLCAL}	Compute closure offset corrections
\zxhlp{BLCHN}	Compute closure offset corrections on a channel-by-channel basis
\zxhlp{BLING}	find residual rate and delay on individual baselines
\zxhlp{BLOAT}	converts line data to greater number channels
\zxhlp{BLSUM}	sums images over irregular sub-images, displays spectra
\zxhlp{BLWUP}	Blow up an image by any positive integer factor.
\zxhlp{BOXES}	Adds Clean boxes to \zxhlp{BOXFILE} around sources from a list
\zxhlp{BPASS}	computes spectral bandpass correction table
\zxhlp{BPCOR}	Correct BP table.
\zxhlp{BPEDT}	Interactive TV task to edit uv data based on BP tables
\zxhlp{BPERR}	Print and plot \zxhlp{BPASS} closure outputs
\zxhlp{BPLOT}	Plots bandpass tables in 2 dimensions as function of time
\zxhlp{BPSMO}	Smooths or interpolates bandpass tables to regular times
\zxhlp{BPWAY}	Determines channel-dependent relative weights
\zxhlp{PWGT}	Calibrates data and scales weights by bandpass correction
\zxhlp{BSAVG}	Task to do an \zxhlp{FFT}-weighted sum of beam-switched images
\zxhlp{BSCAN}	seeks best scan to use for phase cal, fringe search, ..
\zxhlp{BSCLN}	Hogbom Clean on beam-switched difference image
\zxhlp{BSCOR}	Combines two beam-switched images
\zxhlp{BSFIX}	Corrects the ra/dec offsets recorded by the 12m
\zxhlp{BSGEO}	Beam-switched Az-El image to RA-Dec image translation
\zxhlp{BSGRD}	Task to image beam-switched single-dish data
\zxhlp{BSMAP}	images weak sources with closure phases
\zxhlp{BSMOD}	creates single-dish UV beam-switched data with model sources
\zxhlp{BSPRT}	print BS tables
\zxhlp{BSROT}	modifies SD beam-switch continuum data for error in throw
\zxhlp{BSTST}	Graphical display of solutions to frequency-switched data
\zxhlp{CALIB}	determines antenna calibration: complex gain
\zxhlp{CALRD}	Reads calibrator source model-image FITS file
\zxhlp{CALWR}	writes calibrator source model images w CC files to FITS disk files
\zxhlp{CANDY}	user-definable (paraform) task to create an AIPS image
\zxhlp{CANPL}	translates a plot file to a Canon printer/plotter

```

\zxhlp{CAPLT} plots closure amplitude and model from CC file
\zxhlp{CC2IM} Make model image from a CC file
\zxhlp{CCEDT} Select CC components in BOXes and above mininum flux.
\zxhlp{CCFND} prints the contents of a Clean Components extension file.
\zxhlp{CCGAU} Converts point CLEAN components to Gaussians
\zxhlp{CCMOD} generates clean components to fit specified source model
\zxhlp{CCMRG} sums all clean components at the same pixel
\zxhlp{CCNTR} generate a contour plot file from an image
\zxhlp{CCRES} Removes or restores a CC file to a map with a gaussian beam.
\zxhlp{CCSEL} Select signifigant CC components
\zxhlp{CENTR} modifies UV data to center the reference channel
\zxhlp{CHKFC} makes images of Clean boxes from Boxfile
\zxhlp{CL2HF} Convert CL table to HF table
\zxhlp{CLCAL} merges and smooths SN tables, applies them to CL tables
\zxhlp{CLCOP} copy CL/SN file calibration between polarizations or IFs
\zxhlp{CLCOR} applies user-selected corrections to the calibration CL table
\zxhlp{CLINV} copy CL/SN file inverting the calibration
\zxhlp{CLIP} edits data based on amplitudes, phases, and weights out of range
\zxhlp{CLIPM} plots closure phase and model from CC file
\zxhlp{CLSMO} smooths a calibration CL table
\zxhlp{CLVLB} Corrects CL table gains for pointing offsets in VLBI data
\zxhlp{CNTR} generate a contour plot file or TV plot from an image
\zxhlp{COHER} Baseline Phase coherence measurement
COMAP_DO \zxhlp{MX} adverbs not changed by \zxhlp{COMAP}
\zxhlp{COMAP} Procedure to MAP and Self-Calibrate a UV DATA set
COMAP_MX \zxhlp{MX} adverbs not changed by \zxhlp{COMAP}
COMAP_NA Procedure to MAP and Self-Calibrate a UV DATA set
COMAP_UV Procedure to MAP and Self-Calibrate a UV DATA set
\zxhlp{COMB} combines two images by a variety of mathematical methods
\zxhlp{CONF1} Optimizes array configuration by minimum side lobes
\zxhlp{CONPL} Plots AIPS gridding convolution functions
\zxhlp{CONVL} convolves an image with a gaussian or another image
\zxhlp{CORER} calculates correlator statistics and flags bad ones
\zxhlp{CORFQ} corrects uvw for incorrect observing frequency
\zxhlp{CPASS} computes polynomial spectral bandpass correction table
\zxhlp{CPYRT} replaces history with readme file, inserts copyright
\zxhlp{CSCOR} applies specified corrections to CS tables
\zxhlp{CUBIT} Model a galaxy's density and velocity distribution from full cube
\zxhlp{CVEL} shifts spectral-line UV data to a given velocity
\zxhlp{CXCLN} Complex Hogbom CLEAN
\zxhlp{DAYFX} Fixes day number problems left by \zxhlp{FILLM}
\zxhlp{DBAPP} appends one or more data sets to the output data set
\zxhlp{DBCON} concatenates two UV data sets
\zxhlp{DCONV} deconvolves a gaussian from an image
\zxhlp{DECOR} Measures the decorrelation between channels and IF of uv data
\zxhlp{DEFLG} edits data based on decorrelation over channels and time
\zxhlp{DELZN} Determines residual atmosphere depth at zenith and clock errors
\zxhlp{DESCM} copies a portion of a UV data set
\zxhlp{DFCOR} applies user-selected corrections to the calibration CL table
\zxhlp{DFQID} modifies UV data changing the indicated FQIDs
\zxhlp{DFTIM} Makes image of DFT at arbitrary point showing time vs frequency
\zxhlp{DFTPL} plots DFT of a UV data set at arbitrary point versus time
\zxhlp{DIFRL} divides the RR data by LL data
\zxhlp{DIFUV} Outputs the difference of two matching input uv data sets
\zxhlp{DISKU} shows disk use by one or all users
\zxhlp{DQUAL} Rearranges source list, dropping qualifiers
\zxhlp{DRCHK} stand-alone program checks system setup files for consistency

```

13.35. TASK

13. Current AIPS Software

```

\zxhlp{DSKEW}           Geometric interpolation correction for skew
\zxhlp{DSORC}           copies a data set eliminating some source numbers
\zxhlp{DSTOK}           Drops the cross-hand polarizations
\zxhlp{DTCHK}           Task to check results of a test using simulated data.
\zxhlp{DTSIM}           Generate fake UV data
\zxhlp{DTSUM}           Task to provide a summary of the contents of a dataset
\zxhlp{EDITA}            Interactive TV task to edit uv data based on TY/SY/SN/CL tables
\zxhlp{EDITR}            Interactive baseline-oriented visibility editor using the TV
\zxhlp{ELFIT}            Plots/fits selected contents of SN, TY, SY, PC or CL files
\zxhlp{ELINT}            Determines and removes gain dependence on elevation
\zxhlp{EVASN}            Evaluates statistics in SN/CL tables
\zxhlp{EVAUV}           Subtracts & divides a model into UV data, does statistics on results
\zxhlp{EXTAB}            exports AIPS table data as tab-separated text
\zxhlp{FACES}            makes images of catalog sources for initial calibration
\zxhlp{FARAD}            add ionospheric Faraday rotation to CL table
\zxhlp{FARS}             Faraday rotation synthesys based on the brightness vs wavelength
\zxhlp{FETCH}            Reads an image from an external text file.
\zxhlp{FFT}              takes Fourier Transform of an image or images
\zxhlp{FGCNT}           Counts samples comparing two flag tables
\zxhlp{FGDIF}            Compares affect of 2 FG tables
\zxhlp{FGPLT}           Plots selected contents of FG table
\zxhlp{FGTAB}           Plots selected contents of FG table
\zxhlp{FILIT}            Interactive \zxhlp{BOXFILE} editing with facet images
\zxhlp{FILLM}            reads \zxhlp{VLA} on-line/archive format uv data tapes (post Jan 88)
\zxhlp{FILLR}            reads old \zxhlp{VLA} on-line-system tapes into AIPS
\zxhlp{FINDR}            Find normal values for a uv data set
\zxhlp{FIT2A}            reads the fits input file and records it to the output ascii file
\zxhlp{FITAB}            writes images / uv data w extensions to tape in FITS format
\zxhlp{FITLD}            Reads FITS files to load images or UV (IDI or UVFITS) data to disk
\zxhlp{FITTP}            writes images / uv data w extensions to tape in FITS format
\zxhlp{FIXAL}            least squares fit aliasing function and remove
\zxhlp{FIXAN}            fixes the contents of the ANtenna extension file
\zxhlp{FIXBX}            converts a \zxhlp{BOXFILE} to another for input to \zxhlp{IMAGR}
\zxhlp{FIXRL}            correctes right vs left polarizations for a list of antennas
\zxhlp{FIXWT}            Modify weights to reflect amplitude scatter of data
\zxhlp{FLAGR}            Edit data based on internal RMS, amplitudes, weights
\zxhlp{FLATN}            Re-grid multiple fields into one image incl sensitivity
\zxhlp{FLGIT}            flags data based on the rms of the spectrum
\zxhlp{FLOPM}            reverses the spectral order of UV data, can fix \zxhlp{VLA} error
\zxhlp{FQUBE}            collects n-dimensional images into n+1-dimensional \zxhlp{FREQID} image
\zxhlp{FRCAL}            Faraday rotation self calibration task
\zxhlp{FRING}            fringe fit data to determine antenna calibration, delay, rate
\zxhlp{FRMAP}            Task to build a map using fringe rate spectra
\zxhlp{FRPLT}            Task to plot fringe rate spectra
\zxhlp{FTFLG}            interactive flagging of UV data in channel-time using the TV
\zxhlp{FUDGE}            modifies UV data with user's algorithm: paraform task
\zxhlp{FXPOL}            Corrects VLBA polarization assignments
\zxhlp{FXTIM}            fixes start date so all times are positive
\zxhlp{FXVLA}            Task to correct \zxhlp{VLA} data for on-line errors in special cases.
\zxhlp{FXVLB}            Builds a CQ table to enable VLBA correlator loss corrections
\zxhlp{GAL}               Determine parameters from a velocity field
\zxhlp{GCPLT}            Plots gain curves from text files
\zxhlp{GETJY}            determines calibrator flux densities
\zxhlp{GLENS}            models galaxy gravitational lens acting on 3 component source
\zxhlp{GPSDL}            Calculate ionospheric delay and Faraday rotation corrections
\zxhlp{GREYS}            plots images as contours over multi-level grey
\zxhlp{GRIPR}            standalone program to enter suggestions/complaints to AIPS
\zxhlp{GSCAT}            reads Fits Guide star catalog file

```

```

\zxhlp{GSTAR} Task to read a Guide Star (UK) table and create an ST table.
\zxhlp{HA2TI} Converts data processed by \zxhlp{TI2HA} (STUFFER) back to real times
\zxhlp{HAFIX} Recomputes u,v,w when time is hour angle (UVdata is output of \zxhlp{TI2HA})
\zxhlp{HF2SV} convert HF tables from \zxhlp{FRING}/\zxhlp{MBDLY} to form used by Calc/Solve
\zxhlp{HFPRT} write HF tables from \zxhlp{CL2HF}
\zxhlp{HGEOM} interpolates image to different gridding and/or geometry
\zxhlp{HISEQ} task to translate image by histogram equalization

HLPAGAUS Interactive Gaussian absorption fitting task \zxhlp{AGAUS} - run-time help
HLPCLEAN Cleaning tasks - run-time help
HLPEDIBP Interactive BP table uv-data editor \zxhlp{BPEDT} - run-time help
HLPEDICL Interactive SN/CL table uv-data editor - run-time help
HLPEDIPC Interactive PC table editor \zxhlp{PCEDT} - run-time help
HLPEDISN Interactive SN/CL table (not UV) editor - run-time help
HLPEDISS Interactive SY table (not UV) editor - run-time help
HLPEDISY Interactive SY table uv-data editor \zxhlp{EDITA} - run-time help
HLPEDITS Interactive TY table (not UV) editor - run-time help
HLPEDITY Interactive TY table uv-data editor \zxhlp{EDITA} - run-time help
HLPEDIUV Interactive uv-data editor \zxhlp{EDITR} - run-time help
HLPFILIT Interactive Clean box file editing with image display - run-time help
HLPFTFLG Interactive time-channel visibility Editor - run-time help
HLPIBLED Interactive Baseline based visibility Editor - run-time help
HLPPCFLG Interactive time-channel PC table Editor \zxhlp{PCFLG} - run-time help
HLPPLAYR OOP TV class demonstration task - run-time help
HLPRMFIT Polarization fitting task \zxhlp{RMFIT} - run-time help
HLPSCIMG Full-featured image plus self-cal loops, editing - run-time help
HLPSCMAP Imaging plus self-cal and editing \zxhlp{SCMAP} - run-time help
HLPSPFLG Interactive time-channel visibility Editor \zxhlp{SPFLG} - run-time help
HLPTVFLG Interactive time-baseline visibility Editor \zxhlp{TVFLG} - run-time help
HLPTVHLD Interactive image display with histogram equalization - run-time help
HLPTVHUI Interactive intensity-hue-saturation display - run-time help
HLPTVRGB Interactive red-green-blue display - run-time help
HLPTVSAD Find & fit Gaussians to an image with interaction - run-time help
HLPTVSPC Interactive display of spectra from a cube - run-time help
HLPFLAG Edit uv-data on a grid \zxhlp{UFLAG} - run-time help
HLPWIPER edit uv data from \zxhlp{UVPLT}-like plot \zxhlp{WIPER} - run-time help
HLPXGAUS Interactive Gaussian fitting task \zxhlp{XGAUS} - run-time help
HLPZAMAN Fits 1-dimensional Zeeman model to absorption data - run-time help
HLPZEMAN Fits 1-dimensional Zeeman model to data - run-time help

\zxhlp{HOLGR} Read & process holography visibility data to telescope images
\zxhlp{HOLOG} Read & process holography visibility data to telescope images
\zxhlp{HORUS} makes images from unsorted UV data, applying any calibration
\zxhlp{HUINT} make RGB image from images of intensity & hue, like \zxhlp{TVHUEINT}
\zxhlp{IBLED} Interactive BaseLine based visibility EDitor
\zxhlp{IM2CC} Task to convert an image to multi-facet Clean Components
\zxhlp{IM2UV} converts an image to a visibility data set
\zxhlp{IMAGR} Wide-field and/or wide-frequency Cleaning / imaging task.
\zxhlp{IMCLP} Clip an image to a specified range.
\zxhlp{IMEAN} displays the mean & extrema and plots histogram of an image
\zxhlp{IMERG} merges images of different spatial resolutions
\zxhlp{IMFIT} Fits Gaussians to portions of an image
\zxhlp{IMFLT} fits and removes a background intensity plane from an image
\zxhlp{IMLHS} converts images to luminosity/hue TV display
\zxhlp{IMLIN} Fits and removes continuum emission from cube
\zxhlp{IMLOD} reads tape to load images to disk
\zxhlp{IMMOD} adds images of model objects to an image
\zxhlp{IMRMS} Plot \zxhlp{IMEAN} rms answers
\zxhlp{IMTXT} Write an image to an external text file.
\zxhlp{IMVIM} plots one image's values against another's

```

13.35. TASK

13. Current AIPS Software

```

\zxhlp{INDXH}      writes index file describing contents of UV data base
\zxhlp{INDXR}      writes index file describing contents of UV data base
\zxhlp{IRING}      integrates intensity / flux in rings / ellipses
\zxhlp{ISPEC}       Plots and prints spectrum of region of a cube
\zxhlp{JMFIT}       Fits Gaussians to portions of an image
\zxhlp{KNTR}        make a contour/grey plot file from an image w multiple panels
\zxhlp{KRING}       fringe fit data to determine antenna calibration, delay, rate
\zxhlp{LAYER}        Task to create an RGB image from multiple images
\zxhlp{LDGPS}        load GPS data from an ASCII file
\zxhlp{LGEOM}        regrids images with rotation, shift using interpolation
\zxhlp{LISTR}        prints contents of UV data sets and assoc. calibration tables
\zxhlp{LOCIT}        fits antenna locations from SN-table data
\zxhlp{LPCAL}        Determines instrumental polarization for UV data
\zxhlp{LTESS}        makes mosaic images by linear combination
\zxhlp{LWPLA}        translates plot file(s) to a PostScript printer or file
\zxhlp{M3TAR}        translate Haystack MKIII VLBI format "A" TAR's into AIPS
\zxhlp{MANDL}        creates an image of a subset of the Mandlebrot Set
\zxhlp{MAPBM}        Map \zxhlp{VLA} beam polarization
\zxhlp{MATCH}        changes antenna, source, FQ numbers to match a data set
\zxhlp{MATHS}        operates on an image with a choice of mathematical functions
\zxhlp{MBDLY}        Fits multiband delays from IF phases, updates SN table
\zxhlp{MCUBE}        collects n-dimensional images into n+1-dimensional image
\zxhlp{MEDI}         combines four images by a variety of mathematical methods
\zxhlp{MF2ST}        Task to generate an ST ext. file from Model Fit ext. file
\zxhlp{MFPRT}        prints MF tables in a format needed by modelling software
\zxhlp{MK3IN}        translate Haystack MKIII VLBI format "A" tapes into AIPS
\zxhlp{MK3TX}        extract text files from a MKIII VLBI archive tape
\zxhlp{MODAB}        Makes simple absorption/emission spectral-line image in I/V
\zxhlp{MODIM}        adds images of model objects to image cubes in IQU polarization
\zxhlp{MODSP}        adds images of model objects to image cubes in I/V polarization
\zxhlp{MODVF}        task to create a warped velocity field
\zxhlp{MOMFT}        calculates images of moments of a sub-image
\zxhlp{MOMNT}        calculates images of moments along x-axis (vel, freq, ch)
\zxhlp{MORIF}        Combines IFs or breaks spectral windows into multiple windows (IFs)
\zxhlp{MOVE}         Task to copy or move data from one user or disk to another
\zxhlp{MSORT}        Sort a UV dataset into a specified order
\zxhlp{MULIF}        Change number of IFs in output
\zxhlp{MULTI}        Task to convert single-source to multi-source UV data
\zxhlp{MWFLT}        applies linear & non-linear filters to images
\zxhlp{MX}           makes images & deconvolves using UV data directly - replaced
\zxhlp{NANS}         reads an image or a UV data set and looks for NaNs
\zxhlp{NINER}        Applies various 3x3 area operators to an image.
\zxhlp{NNLSQ}        Non-Negative-Least-Squares decomposition of spectrum
\zxhlp{NOBAT}        Task to lock lower priority users out of the AP
\zxhlp{NOIFS}        makes all IFs into single spectrum
\zxhlp{OBEDT}        Task to flag data of orbiting antennas
\zxhlp{OBPLT}        Plot columns of an OB table.
\zxhlp{OBTAB}        Recalculate orbit parameters and other spacecraft info
\zxhlp{OFLAG}        uses on-line flag table information to write a flag table
\zxhlp{OGEOM}        Simple image rotation, scaling, and translation
\zxhlp{OHGEO}        Geometric interpolation with correction for 3-D effects
\zxhlp{OMFIT}        Fits sources and, optionally, a self-cal model to uv data
\zxhlp{OOSRT}        Sort a UV dataset into a specified order
\zxhlp{OOSUB}        Subtracts/divides a model from/into a uv data base
\zxhlp{OTFBS}        Translates on-the-fly continuum SDD format to AIPS UV file
\zxhlp{OTFIN}        Lists on-the-fly single-dish SDD format data files
\zxhlp{UTFUV}        Translates on-the-fly single-dish SDD format to AIPS UV file
\zxhlp{PADIM}        Task to increase image size by padding with some value

```

```

\zxhlp{PANEL}          Convert \zxhlp{HOLGR} output to panel adjustment table
\zxhlp{PASTE}           Pastes a selected subimage of one image into another.
\zxhlp{PATGN}           Task to create a user specified test or primary-beam pattern
\zxhlp{PBCOR}           Task to apply or correct an image for a primary beam
\zxhlp{PBEAM}           Fits the analytic function to the measured values of the beam
\zxhlp{PCAL}             Determines instrumental polarization for UV data
\zxhlp{PCASS}           Finds amplitude bandpass shape from pulse-cal table data
\zxhlp{PCCOR}           Corrects phases using \zxhlp{PCAL} tones data from PC table
\zxhlp{PCEDT}           Interactive TV task to edit PC tables
\zxhlp{PCFIT}           Finds delays and phases using a PC table
\zxhlp{PCFLG}           interactive flagging of Pulse-cal data in channel-TB using the TV
\zxhlp{PCLOD}           Reads ascii file containing pulse-cal info to PC table.
\zxhlp{PCNTR}            Generate plot file with contours plus polarization vectors
\zxhlp{PCPLT}            Plots pulse-cal tables in 2 dimensions as function of time
\zxhlp{PCVEL}            shifts spectral-line UV data to a given velocity: planet version
\zxhlp{PFPL1}            Paraform Task to generate a plot file: (does grey scale)
\zxhlp{PFPL2}            Paraform Task to generate a plot file: (slice intensity)
\zxhlp{PFPL3}            Paraform Task to generate a plot file: (does histogram)
\zxhlp{PGEOM}            Task to transform an image into polar coordinates.
\zxhlp{PHASE}             Baseline Phase coherence measurement
\zxhlp{PHCLN}            \zxhlp{PHCLN} has been removed, use \zxhlp{PHAT} adverb in \zxhlp{APCLN}.
\zxhlp{PHNEG}            Negates a UV datafile's visibility phase.
\zxhlp{PHSRF}            Perform phase-referencing within a spectral line database.
\zxhlp{PLAYR}             Verb to load an image into a TV channel
\zxhlp{PLCUB}            Task to plot intensity vs x panels on grid of y,z pixels
\zxhlp{PLOTR}            Basic task to generate a plot file from text input
\zxhlp{PLROW}            Plot intensity of a series of rows with an offset.
\zxhlp{POLCO}            Task to correct polarization maps for Ricean bias
\zxhlp{POLSN}            Make a SN table from cross polarized fringe fit
\zxhlp{POSSM}            Task to plot total and cross-power spectra.
\zxhlp{PROFL}             Generates plot file for a profile display.
\zxhlp{PRTAB}             prints any table-format extension file
\zxhlp{PRTAC}             prints contents and summaries of the accounting file
\zxhlp{PRTAN}             prints the contents of the ANtenna extension file
\zxhlp{PRTCC}             prints the contents of a Clean Components extension file.
\zxhlp{PRTIM}             prints image intensities from an MA catalog entry
\zxhlp{PRTOF}             prints on-line flag table information
\zxhlp{PRTPL}             Task to send a plot file to the line printer
\zxhlp{PRTSD}             prints contents of AIPS single-dish data sets
\zxhlp{PRTSY}             Task to print statistics from the SY table
\zxhlp{PRTTP}             prints contents of tapes, all supported formats
\zxhlp{PRTUV}             prints contents of a visibility (UV) data set
\zxhlp{QMSPL}             Task to send a plot file to the QMS printer/plotter
\zxhlp{QUACK}              Flags beginning or end portions of UV-data scans
\zxhlp{QUFIX}             determines Right minus Left phase difference, corrects cal files
\zxhlp{QUOUT}             writes text file of Q, U versus frequency to be used by \zxhlp{RLDIF}
\zxhlp{QUXTR}             extracts text files from Q,U cubes for input to \zxhlp{TARS}
\zxhlp{REAMP}              modifies UV data re-scaling the amplitudes
\zxhlp{REFLG}             Attempts to compress a flag table
\zxhlp{REGRD}             Regrads an image from one co-ordinate frame to another
\zxhlp{REIFS}              Breaks spectral windows into multiple spectral windows (IFs)
\zxhlp{REMAG}             Task to replace magic blanks with a user specified value
\zxhlp{RESEQ}              Renumber antennas
\zxhlp{REWAY}             computes weights based in rms in spectra
\zxhlp{REWGT}             modifies UV data re-scaling the weights only
\zxhlp{RFARS}             Correct Q/U cubes for Faraday rotation synthesis results
\zxhlp{RFI}                Look for \zxhlp{RFI} in uv data
\zxhlp{RFLAG}             Flags data set based on time and freq rms in fringe visibilities

```

13.35. TASK

13. Current AIPS Software

\zxhlp{RGBMP}	Task to create an RGB image from the 3rd dim of an image
\zxhlp{RLCAL}	Determines instrumental right-left phase versus time (a self-cal)
\zxhlp{RLCOR}	corrects a data set for R-L phase differences
\zxhlp{RLDIF}	determines Right minus Left phase difference, corrects cal files
\zxhlp{RLDLY}	fringe fit data to determine antenna R-L delay difference
\zxhlp{RM2PL}	Plots spectrum of a pixel with \zxhlp{RMFIT} fit
\zxhlp{RMFIT}	Fits 1-dimensional polarization spectrum to Q/U cube
\zxhlp{RM}	Task to calculate rotation measure and magnetic field
\zxhlp{RMSD}	Calculate rms for each pixel using data at the box around the pixel
\zxhlp{RSPEC}	Plots and prints spectrum of rms of a cube
\zxhlp{RSTOR}	Restores a CC file to a map with a gaussian beam.
\zxhlp{RTIME}	Task to test compute times
\zxhlp{SABOX}	create box file from source islands in facet images
\zxhlp{SAD}	Finds and fits Gaussians to portions of an image
\zxhlp{SBCOR}	Task to correct VLBA data for phase shift between USB & LSB
\zxhlp{SCIMG}	Full-featured imaging plus self-calibration loop with editing
\zxhlp{SCLIM}	operates on an image with a choice of mathematical functions
\zxhlp{SCMAP}	Imaging plus self-calibration loop with editing
\zxhlp{SDCAL}	Task to apply single dish calibration
\zxhlp{SDCLN}	deconvolves image by Clark and then "SDI" cleaning methods
\zxhlp{SDGRD}	Task to select and image random-position single-dish data
\zxhlp{SDIMG}	Task to select and image random-position single-dish data
\zxhlp{SDLSF}	least squares fit to channels and subtracts from SD uv data
\zxhlp{SDMOD}	modifies single-dish UV data with model sources
\zxhlp{SDTUV}	Task to convert SD table files to UV like data.
\zxhlp{SDVEL}	shifts spectral-line single-dish data to a given velocity
\zxhlp{SERCH}	Finds line signals in transposed data cube
\zxhlp{SETAN}	Reads an ANtenna file info from a text file
\zxhlp{SETFC}	makes a \zxhlp{BOXFILE} for input to \zxhlp{IMAGR}
\zxhlp{SETJY}	Task to enter source info into source (SU) table.
\zxhlp{SHADO}	Calculate the shadowing of antennas at the array
\zxhlp{SHADW}	Generates the "shadowed" representation of an image
\zxhlp{SHOUV}	displays uv data in various ways.
\zxhlp{SKYVE}	Regrids a DSS image from one co-ordinate frame to another
\zxhlp{SL2PL}	Task to convert a Slice File to a Plot File
\zxhlp{SLCOL}	Task to collate slice data and models.
\zxhlp{SLFIT}	Task to fit gaussians to slice data.
\zxhlp{SLICE}	Task to make a slice file from an image
\zxhlp{SLPRT}	Task to print a Slice File
\zxhlp{SMOTH}	Task to smooth a subimage from upto a 7-dim. image
\zxhlp{SNCOP}	Task to copy SN table averaging some input IFs
\zxhlp{SNCOR}	applies user-selected corrections to the calibration SN table
\zxhlp{SNDUP}	copies and duplicates SN table from single pol file to dual pol
\zxhlp{SNEDT}	Interactive SN/CL/TY/SY table editor using the TV
\zxhlp{SNFIT}	Fits parabola to SN amplitudes and plots result
\zxhlp{SNFLG}	Writes flagging info based on the contents of SN files
\zxhlp{SNIFS}	Plots selected contents of SN, TY, SY, PC or CL files
\zxhlp{SNP2D}	Task to convert SN table single-channel phase to delay
\zxhlp{SNPLT}	Plots selected contents of SN, SY, TY, PC or CL files
\zxhlp{SNREF}	Chooses best reference antenna to minimize R-L differences
\zxhlp{SNSMO}	smooths and filters a calibration SN table
\zxhlp{SOLCL}	adjust gains for solar data according to nominal sensitivity
\zxhlp{SOUSP}	fits source spectral index from SU table or adverbs
\zxhlp{SPCAL}	Determines instrumental polzn. for spec. line UV data
\zxhlp{SPCOR}	Task to correct an image for a primary beam and spectral index
\zxhlp{SPECR}	Spectral regridding task for UV data
\zxhlp{SPFLG}	interactive flagging of UV data in channel-TB using the TV
\zxhlp{SPIXR}	Fits spectral indexes to each row of an image incl curvature

\zxhlp{SPLAT}	Applies calibration and splits or assemble selected sources.
\zxhlp{SPLIT}	converts multi-source to single-source UV files w calibration
\zxhlp{SPMOD}	Modify UV database by adding a model with spectral lines
\zxhlp{SQASH}	Task to collapse several planes in a cube into one plane or row
\zxhlp{STACK}	Task to co-add a set of 2-dimensional images with weighting
\zxhlp{STARS}	Task to generate an ST ext. file with star positions
\zxhlp{STEER}	Task which deconvolves the David Steer way.
\zxhlp{STESS}	Task which finds sensitivity in mosaicing
\zxhlp{STFND}	Task to find stars in an image and generate an ST table.
\zxhlp{STFUN}	Task to calculate a structure function image
\zxhlp{STRAN}	Task compares ST tables, find image coordinates (e.g. guide star)
\zxhlp{SUBIM}	Task to select a subimage from up to a 7-dim. image
\zxhlp{SUFIX}	modifies source numbers on uv data
\zxhlp{SUMIM}	Task to sum overlapping, sequentially-numbered images
\zxhlp{SUMSQ}	Task to sum the squared pixel values of overlapping,
\zxhlp{SWAPR}	modifies UV data by swapping real and imaginary parts
\zxhlp{SWPOL}	Swap polarizations in a UV data base
\zxhlp{SY2TY}	Task to generate a TY extension file from an \zxhlp{EVLA} SY table
\zxhlp{SYSOL}	undoes and re-does nominal sensitivity application for Solar data
\zxhlp{TABED}	Task to edit tables
\zxhlp{TACOP}	task to copy tables, other extension files
\zxhlp{TAFFY}	User definable task to operate on an image
\zxhlp{TAFLG}	Flags data in a Table extension file
\zxhlp{TAMRG}	Task to merge table rows under specified conditions
\zxhlp{TAPLT}	Plots data from a Table extension file
\zxhlp{TAPPE}	Task to append 2 tables and merge to output table
\zxhlp{TARPL}	Plot output of \zxhlp{TARS} task
\zxhlp{TARS}	Simulation of Faraday rotation synthesis (mainly task \zxhlp{FARS})
\zxhlp{TASAV}	Task to copy all extension tables to a dummy uv or map file
\zxhlp{TASRT}	Task to sort extension tables.
\zxhlp{TBAVG}	Time averages data combining all baselines.
\zxhlp{TBDIF}	Compare entries in two tables
\zxhlp{TBIN}	Reads a text file AIPS table into AIPS
\zxhlp{TBOUT}	Writes an AIPS table into a text file for user editting.
\zxhlp{TBSUB}	Make a new table from a subset of an old table
\zxhlp{TBTSK}	Paraform OOP task for tables
\zxhlp{TCOPY}	Tape to tape copy with some disk FITS support
\zxhlp{TECOR}	Calculate ionospheric delay and Faraday rotation corrections
\zxhlp{TFILE}	sorts and edits MkIII correlator UNIX-based A-file.
\zxhlp{TI2HA}	modifies times in UV data to hour angles
\zxhlp{TIORD}	checks data for time baseline ordering, displays failures
\zxhlp{TKPL}	Task to send a plot file to the TEK
\zxhlp{TRANS}	Task to transpose a subimage of an up to 7-dim. image
\zxhlp{TRUEP}	determines true antenna polarization from special data sets
\zxhlp{TVCPSS}	Task to copy a TV screen-image to a PostScript file.
\zxhlp{TVDIC}	Task to copy a TV screen-image to a Dicommed film recorder.
\zxhlp{TVFLG}	interactive flagging of UV data using the TV
\zxhlp{TVHLD}	Task to load an image to the TV with histogram equalization
\zxhlp{TVHUI}	make TV image from images of intensity, hue, saturation
\zxhlp{TVPL}	Display a plot file on the TV
\zxhlp{TVRGB}	make TV image from images of true color (RGB) images
\zxhlp{TVSAD}	Finds and fits Gaussians to portions of an image with interaction
\zxhlp{TVSPC}	Display images and spectra from a cube
\zxhlp{TXPL}	Displays a plot (PL) file on a terminal or line printer
\zxhlp{TYAPL}	undoes and re-does nominal sensitivity application
\zxhlp{TYCOP}	copy TY or SY table calibration between IFs
\zxhlp{TYSMO}	smooths and filters a calibration TY or SY table
\zxhlp{UBAVG}	Baseline dependent time averaging of uv data

13.35. TASK

13. Current AIPS Software

```

\zxhlp{UFLAG}          Plots and edits data using a uv-plane grid and the TV
\zxhlp{UJOIN}           modifies UV data converting IFs to spectral channels
\zxhlp{UNCAL}           sets up tables for uncalibrating Australia Telescope data
\zxhlp{USUBA}           Assign subarrays within a uv-data file
\zxhlp{UTESS}           deconvolves images by maximizing emptiness
\zxhlp{UV2MS}           Append single-source file to multi-source file.
\zxhlp{UV2TB}           Converts UV autocorrelation spectra to tables
\zxhlp{UVADC}           Fourier transforms and corrects a model and adds to uv data.
\zxhlp{UVAVG}           Average or merge a sorted (BT, TB) uv database
\zxhlp{UVBAS}           averages several channels and subtracts from uv data.
\zxhlp{UVCMP}           Convert a UV database to or from compressed format
\zxhlp{UVCON}           Generate sample UV coverage given a user defined array layout
\zxhlp{UVCOP}           Task to copy a subset of a UV data file
\zxhlp{UVCRS}           Finds the crossing points of UV-ellipses.
\zxhlp{UVDEC}           Decrement the number of spectral channels, keeping every nth
\zxhlp{UVDGP}           Copy a UV data file, deleting a portion of it
\zxhlp{UVDI1}           Subtract UV data(averaged up to one time) from the other UV data
\zxhlp{UVDIF}           prints differences between two UV data sets
\zxhlp{UVFIL}           Create, fill a uv database from user supplied information
\zxhlp{UVFIT}           Fits source models to uv data.
\zxhlp{UVFIX}           Recomputes u,v,w for a uv database
\zxhlp{UVFLG}           Flags UV-data
\zxhlp{UVFND}           prints selected data from UV data set to search for problems
\zxhlp{UVFRE}           Makes one data set have the spectral structure of another
\zxhlp{UVGIT}           Fits source models to uv data.
\zxhlp{UVGLU}           Glues UV data frequency blocks back together
\zxhlp{UVHGM}           Plots statistics of uv data files as histogram.
\zxhlp{UVHIM}           Makes image of the histogram on two user-chosen axes
\zxhlp{UVHOL}           prints holography data from a UV data base with calibration
\zxhlp{UVIMG}           Grid UV data into an "image"
\zxhlp{UVLIN}           Fits and removes continuum visibility spectrum, also can flag
\zxhlp{UVLOD}           Read export or FITS data from a tape or disk
\zxhlp{UVLSD}           least squares fit to channels and divides the uv data.
\zxhlp{UVLSF}           least squares fit to channels and subtracts from uv data.
\zxhlp{UVMAP}           makes images from calibrated UV data.
\zxhlp{UVMLN}           edits data based on the rms of the spectrum
\zxhlp{UVMOD}           Modify UV database by adding a model incl spectral index
\zxhlp{UVMTH}           Averages one data set and applied it to another.
\zxhlp{UVNOU}           flags uv samples near the U,V axes to reduce interference
\zxhlp{UVPLT}           plots data from a UV data base
\zxhlp{UVPOL}           modifies UV data to make complex image and beam
\zxhlp{UVPRM}           measures parameters from a UV data base
\zxhlp{UVPRT}           prints data from a UV data base with calibration
\zxhlp{UVRFI}           Mitigate \zxhlp{RFI} by Fourier transform or fitting the circle
\zxhlp{UVSEN}           Determine RMS sidelobe level and brightness sensitivity
\zxhlp{UVSIM}           Generate sample UV coverage given a user defined array layout
\zxhlp{UVSRT}           Sort a UV dataset into a specified order
\zxhlp{UVSUB}           Subtracts/divides a model from/into a uv data base
\zxhlp{UWWAX}           flags uv samples near the U,V axes to reduce interference
\zxhlp{VBCAL}           Scale visibility amplitudes by antenna based constants
\zxhlp{VBGLU}           Glues together data from multiple passes thru the VLBA corr.
\zxhlp{VBMRG}           Merge VLBI data, eliminate duplicate correlations
\zxhlp{VLABP}           \zxhlp{VLA} antenna beam polarization correction for snapshot images
\zxhlp{VLAMP}           Makes \zxhlp{ANTAB} file for phased \zxhlp{VLA} used in VLBI observations
\zxhlp{VLANT}           applies \zxhlp{VLA}/\zxhlp{EVLA} antenna position corrections from OPs files
\zxhlp{VLBAPSS}          computes spectral bandpass correction table
\zxhlp{VLBIN}           Task to read VLBI data from an NRAO/MPI MkII correlator
\zxhlp{VLOG}            Pre-process external VLBA calibration files

```

\zxhlp{VPFLG}	Resets flagging to all or all corss-hand whenever some are flagged
\zxhlp{VPLOT}	plots uv data and model from CC file
\zxhlp{VTESS}	Deconvolves sets of images by the Maximum Entropy Method
\zxhlp{VTEST}	Measures velocity discrepancy across fields
\zxhlp{WARP}	Model warps in Galaxies
\zxhlp{WETHR}	Plots selected contents of WX tables, flags data based on WX
\zxhlp{WFCLN}	Wide field and/or widefrequency CLEANing/imaging task.
\zxhlp{WIPER}	plots and edits data from a UV data base using the TV
\zxhlp{WTMOD}	modifies weights in a UV data set
\zxhlp{WTSUM}	Task to do a a sum of images weighted by other images
\zxhlp{XBASL}	Fits and subtracts nth-order baselines from cube (x axis)
\zxhlp{XG2PL}	Plots spectrum of a pixel with \zxhlp{XGAUS}/\zxhlp{AGAUS} and \zxhlp{ZEMAN}/\zxhlp{ZAMAN} fits
\zxhlp{XGAUS}	Fits 1-dimensional Gaussians to images: restartable
\zxhlp{XMOM}	Fits one-dimensional moments to each row of an image
\zxhlp{XPLOT}	Plots image rows one at a time on the graphics or TV screen
\zxhlp{XSMTH}	Smooth data along the x axis
\zxhlp{XSUM}	Sum or average images on the x axis
\zxhlp{XTRAN}	Create an image with transformed coordinates
\zxhlp{ZAMAN}	Fits 1-dimensional Zeeman model to absorption-line data
\zxhlp{ZEMAN}	Fits 1-dimensional Zeeman model to data

13.36 TV

\zxhlp{BLANK}	blanks out selected, e.g. non-signal, portions of an image
\zxhlp{CHARMULT}	Changes the multiplication factor for TV characters
\zxhlp{CNTR}	generate a contour plot file or TV plot from an image
\zxhlp{COLORS}	specifies the desired TV colors
\zxhlp{COSTAR}	Verb to plot a symbol at given position on top of a TV image
\zxhlp{COTVLOD}	Proc to load an image into a TV channel about a coordinate
\zxhlp{CURBLINK}	switch TV cursor between steady and blinking displays
\zxhlp{CURVALUE}	displays image intensities selected via the TV cursor
\zxhlp{DEFCOLOR}	Sets adverb \zxhlp{PLCOLORS} to match s default \zxhlp{XAS} TV
\zxhlp{DELBOX}	Verb to delet boxes with TV cursor & graphics display.
\zxhlp{DELTAX}	Increment or size in X direction
\zxhlp{DELTAY}	Increment or size in Y direction
\zxhlp{DFILEBOX}	Verb to delete Clean boxes with TV cursor & write to file
\zxhlp{DONEWTAB}	do we make new tables, use a new table format, etc.
\zxhlp{DOTV}	selects use of TV display option in operation
\zxhlp{DRAWBOX}	Verb to draw Clean boxes on the display
\zxhlp{FACTOR}	scales some display or CLEANing process
\zxhlp{FILEBOX}	Verb to reset Clean boxes with TV cursor & write to file
\zxhlp{FILIT}	Interactive \zxhlp{BOXFILE} editing with facet images
\zxhlp{GRBLINK}	Verb which blinks 2 TV graphics planes
\zxhlp{GRCHAN}	specifies the TV graphics channel to be used
\zxhlp{GRCLEAR}	clears the contents of the specified TV graphics channels
\zxhlp{GREAD}	reads the colors of the specified TV graphics channel
\zxhlp{GROFF}	turns off specified TV graphics channels
\zxhlp{GRON}	turns on specified TV graphics channels
\zxhlp{GWRITE}	reads the colors of the specified TV graphics channel
HLPTVHLD	Interactive image display with histogram equalization - run-time help
HLPTVSAD	Find & fit Gaussians to an image with interaction - run-time help
HLPTVSPC	Interactive display of spectra from a cube - run-time help
\zxhlp{HUEWEDGE}	Show a wedge on the TV suitable for \zxhlp{TVHUEINT} displays
\zxhlp{HUINT}	make RGB image from images of intensity & hue, like \zxhlp{TVHUEINT}
\zxhlp{IM2TV}	Verb to convert pixel coordinates to TV pixels
\zxhlp{IMERASE}	replaces an image portion of the TV screen with zeros
\zxhlp{IMLHS}	converts images to luminosity/hue TV display

13.36. TV

13. Current AIPS Software

\zxhlp{IMPOS}	displays celestial coordinates selected by the TV cursor
\zxhlp{IMWEDGE}	load step wedge of full range of image values to TV
\zxhlp{IMXY}	returns pixel coordinates selected by the TV cursor
\zxhlp{MFITSET}	gets adverbs for running \zxhlp{IMFIT} and \zxhlp{JMFIT}
\zxhlp{NBOXES}	Number of boxes
\zxhlp{NCCBOX}	Number of clean component boxes
\zxhlp{OFFHUINT}	Proc which restores TV functions to normal after \zxhlp{TVHUE}
\zxhlp{OFFPSEUD}	Verb which deactivates all pseudo-color displays
\zxhlp{OFFSCROL}	Verb which deactivates scroll of an image
\zxhlp{OFFTRAN}	Verb which restores transfer function to normal
\zxhlp{OFFZOOM}	Verb which returns the hardware IIS zoom to normal
\zxhlp{OFMFILE}	specifies the name of a text file containing OFM values
\zxhlp{PCNTR}	Generate plot file with contours plus polarization vectors
\zxhlp{PIX2XY}	Specifies a pixel in an image
\zxhlp{PIXAVG}	Average image value
\zxhlp{PIXRANGE}	Range of pixel values to display
\zxhlp{PIXSTD}	RMS pixel deviation
\zxhlp{PIXVAL}	Value of a pixel
\zxhlp{PROFL}	Generates plot file for a profile display.
\zxhlp{REBOX}	Verb to reset boxes with TV cursor & graphics display.
\zxhlp{REMOVIE}	Verb to rerun a previously loaded (\zxhlp{TMOVIE}) movie
\zxhlp{REROAM}	Verb to use previous roam image mode, then does roam
\zxhlp{RGBCOLOR}	specifies the desired TV graphics color
\zxhlp{RGBGAMMA}	specifies the desired color gamma corrections
\zxhlp{RGBMP}	Task to create an RGB image from the 3rd dim of an image
\zxhlp{ROAM}	Roam around an image too large for the display.
\zxhlp{ROAMOFF}	Verb to recover image from roam display in simple display mode
\zxhlp{ROMODE}	Specified roam mode
\zxhlp{SETMAXAP}	Examines/alters system parameter limiting dynamic pseudo-AP
\zxhlp{SETROAM}	Verb use to set roam image mode, then do roam. OBSOLETE
\zxhlp{SETSLICE}	Set slice endpoints on the TV interactively
\zxhlp{SETXWIN}	Procedure to set \zxhlp{BLC} and \zxhlp{TRC} with TV cursor
\zxhlp{TBLC}	Gives the bottom left corner of an image to be displayed
\zxhlp{TRC}	Specifies the top right corner of a subimage to be displayed
\zxhlp{TV1SET}	Verb to reset 1D gaussian fitting initial guess on TV plot.
\zxhlp{TV3COLOR}	Verb to initiate 3-color display using 3 TV channels
\zxhlp{TVACOMPS}	Verb to add slice model components directly on TV graphics
\zxhlp{TVAGUESS}	Verb to re-plot slice model guess directly on TV graphics
\zxhlp{TVALL}	Procedure loads image to TV, shows labeled wedge, enhances
\zxhlp{TVAMODEL}	Verb to add slice model display directly on TV graphics
\zxhlp{TVANOT}	Verb to load annotation to the TV image or graphics
\zxhlp{TVARESID}	Verb to add slice model residuals directly on TV graphics
\zxhlp{TVASLICE}	Verb to add a slice display on TV graphics from slice file
\zxhlp{TVBLINK}	Verb which blinks 2 TV planes, can do enhancement also
\zxhlp{TVBOX}	Verb to set boxes with TV cursor & graphics display.
\zxhlp{TVBUT}	Tells which AIPS TV button was pushed
\zxhlp{TVCHAN}	Specified a TV channel (plane)
\zxhlp{TCLEAR}	Verb to clear image from TV channel(s)
\zxhlp{TVCOLORS}	Sets adverb \zxhlp{PLCOLORS} to match the TV (\zxhlp{DOTV}=1) usage
\zxhlp{TVCOMPS}	Verb to display slice model components directly on TV graphics
\zxhlp{TVCORN}	Specified the TV pixel for the bottom left corner of an image
\zxhlp{TVCPS}	Task to copy a TV screen-image to a PostScript file.
\zxhlp{TVCUBE}	Verb to load a cube into tv channel(s) & run a movie
\zxhlp{TVDIC}	Task to copy a TV screen-image to a Dicomed film recorder.
\zxhlp{TVDIST}	determines spherical distance between two pixels on TV screen
\zxhlp{TVFIDDLE}	Verb enhances B/W or color TV image with zooms
\zxhlp{TVFLUX}	displays coordinates and values selected with the TV cursor
\zxhlp{TVGUESS}	Verb to display slice model guess directly on TV graphics

\zxhlp{TVHELIX}	Verb to activate a helical hue-intensity TV pseudo-coloring
\zxhlp{TVHLD}	Task to load an image to the TV with histogram equalization
\zxhlp{TVHUEINT}	Verb to make hue/intensity display from 2 TV channels
\zxhlp{TVHUI}	make TV image from images of intensity, hue, saturation
\zxhlp{TVILINE}	Verb to draw a straight line on an image on the TV
\zxhlp{TVINIT}	Verb to return TV display to a virgin state
\zxhlp{TVLABEL}	Verb to label the (map) image on the TV
\zxhlp{TVLAYOUT}	Verb to label the holography image on the TV with panel layout
\zxhlp{TVLEVS}	Gives the peak intensity to be displayed in levels
\zxhlp{TVLINE}	Verb to load a straight line to the TV image or graphics
\zxhlp{TVLOD}	Verb to load an image into a TV channel
\zxhlp{TVLUT}	Verb which modifies the transfer function of the image
\zxhlp{TVMAXFIT}	displays fit pixel positions and intensity at maxima on TV
\zxhlp{TVMBLINK}	Verb which blinks 2 TV planes either auto or manually
\zxhlp{TVMLUT}	Verb which modifies the transfer function of the image
\zxhlp{TVMODEL}	Verb to display slice model directly on TV graphics
\zxhlp{TVMOVIE}	Verb to load a cube into tv channel(s) & run a movie
\zxhlp{TVNAME}	Verb to fill image name of that under cursor
\zxhlp{TVOFF}	Verb which turns off TV channel(s).
\zxhlp{TVON}	Turns on one or all TV image planes
\zxhlp{TPHLAME}	Verb to activate "flame-like" pseudo-color displays
\zxhlp{TVPL}	Display a plot file on the TV
\zxhlp{TVPOS}	Read a TV screen position using cursor
\zxhlp{TVPSEUDO}	Verb to activate three types of pseudo-color displays
\zxhlp{TVRESET}	Reset the TV without erasing the image planes
\zxhlp{TVRESID}	Verb to display slice model residuals directly on TV graphics
\zxhlp{TVRGB}	make TV image from images of true color (RGB) images
\zxhlp{TVROAM}	Load up to 16 TV image planes and roam a subset thereof
\zxhlp{TVSAD}	Finds and fits Gaussians to portions of an image with interaction
\zxhlp{TVSCROL}	Shift position of image on the TV screen
\zxhlp{TVSET}	Verb to set slice Gaussian fitting initial guesses from TV plot
\zxhlp{TVSLICE}	Verb to display slice file directly on TV
\zxhlp{TVSPC}	Display images and spectra from a cube
\zxhlp{TVSPLIT}	Compare two TV image planes, showing halves
\zxhlp{TVSTAR}	Verb to plot star positions on top of a TV image
\zxhlp{TVSTAT}	Find the mean and RMS in a blotch region on the TV
\zxhlp{TVTRANSF}	Interactively alters the TV image plane transfer function
\zxhlp{TVWEDGE}	Show a linear wedge on the TV
\zxhlp{TVWINDOW}	Set a window on the TV with the cursor
\zxhlp{TVWLABEL}	Put a label on the wedge that you just put on the TV
\zxhlp{TVXY}	Pixel position on the TV screen
\zxhlp{TVZOOM}	Activate the TV zoom
\zxhlp{TXINC}	TV X coordinate increment
\zxhlp{TYINC}	TV Y coordinate increment
\zxhlp{TZINC}	TV Z coordinate increment
\zxhlp{WEDERASE}	Load a wedge portion of the TV with zeros
\zxhlp{XAS}	Information about TV-Servers
\zxhlp{XVSS}	Information about older Sun OpenWindows-specific TV-Server

13.37 TV-APPL

\zxhlp{AGAUS}	Fits 1-dimensional Gaussians to absorption-line spectra
\zxhlp{BLSUM}	sums images over irregular sub-images, displays spectra
\zxhlp{BPEDT}	Interactive TV task to edit uv data based on BP tables
\zxhlp{EDITA}	Interactive TV task to edit uv data based on TY/SY/SN/CL tables
\zxhlp{EDITR}	Interactive baseline-oriented visibility editor using the TV
\zxhlp{FTFLG}	interactive flagging of UV data in channel-time using the TV

13.38. UTILITY

13. Current AIPS Software

```
\zxhlp{GAMMASET}    changes the gamma-correction exponent used in the TV OFM
HLPAGAUS  Interactive Gaussian absorption fitting task \zxhlp{AGAUS} - run-time help
HLCLEAN   Cleaning tasks - run-time help
HLPEDIBP  Interactive BP table uv-data editor \zxhlp{BPEDT} - run-time help
HLPEDICL  Interactive SN/CL table uv-data editor - run-time help
HLPEDIPC  Interactive PC table editor \zxhlp{PCEDT} - run-time help
HLPEDISN  Interactive SN/CL table (not UV) editor - run-time help
HLPEDISS  Interactive SY table (not UV) editor - run-time help
HLPEDISY  Interactive SY table uv-data editor \zxhlp{EDITA} - run-time help
HLPEDITS  Interactive TY table (not UV) editor - run-time help
HLPEDITY  Interactive TY table uv-data editor \zxhlp{EDITA} - run-time help
HLPEDIUV  Interactive uv-data editor \zxhlp{EDITR} - run-time help
HLPFILIT  Interactive Clean box file editing with image display - run-time help
HLPFTFLG  Interactive time-channel visibility Editor - run-time help
HLPIBLED  Interactive Baseline based visibility Editor - run-time help
HLPPCFLG  Interactive time-channel PC table Editor \zxhlp{PCFLG} - run-time help
HLPPLAYR  OOP TV class demonstration task - run-time help
HLPRMFIT  Polarization fitting task \zxhlp{RMFIT} - run-time help
HLPSCIMG  Full-featured image plus self-cal loops, editing - run-time help
HLPSCMAP  Imaging plus self-cal and editing \zxhlp{SCMAP} - run-time help
HLPSPFLG  Interactive time-channel visibility Editor \zxhlp{SPFLG} - run-time help
HLPTVFLG  Interactive time-baseline visibility Editor \zxhlp{TVFLG} - run-time help
HLPTVHUI  Interactive intensity-hue-saturation display - run-time help
HLPTVRGB  Interactive red-green-blue display - run-time help
HLPFLAG   Edit uv-data on a grid \zxhlp{UFLAG} - run-time help
HLPWIPER  edit uv data from \zxhlp{UVPLT}-like plot \zxhlp{WIPER} - run-time help
HLPXGAUS  Interactive Gaussian fitting task \zxhlp{XGAUS} - run-time help
HLPZAMAN  Fits 1-dimensional Zeeman model to absorption data - run-time help
HLPZEMAN  Fits 1-dimensional Zeeman model to data - run-time help
\zxhlp{IBLED}  Interactive BaseLine based visibility EDitor
\zxhlp{IMAGR}   Wide-field and/or wide-frequency Cleaning / imaging task.
\zxhlp{OFMADJUS} interactive linear adjustment of current TV OFM lookup tables
\zxhlp{OFMCONT}  creates/modifies TV color OFMs with level or wedged contours
\zxhlp{OFMDIR}   lists names of the user's and system's OFM files from \zxhlp{OFMFIL}
\zxhlp{OFMGET}   loads TV \zxhlp{OFMS} from an OFM save file
\zxhlp{OFMLIST}  lists the current TV OFM table(s) on the terminal or printer
\zxhlp{OFMSAVE}  saves the TV's current OFM lookup table in a text file
\zxhlp{OFMTWEAK} interactive modification of current TV OFM lookup tables
\zxhlp{OFMZAP}   deletes an OFM lookup table save file
\zxhlp{PCEDT}    Interactive TV task to edit PC tables
\zxhlp{PCFLG}    interactive flagging of Pulse-cal data in channel-TB using the TV
\zxhlp{PLAYR}    Verb to load an image into a TV channel
\zxhlp{RM2PL}    Plots spectrum of a pixel with \zxhlp{RMFIT} fit
\zxhlp{RMFIT}    Fits 1-dimensional polarization spectrum to Q/U cube
\zxhlp{SNEDT}    Interactive SN/CL/TY/SY table editor using the TV
\zxhlp{SPFLG}    interactive flagging of UV data in channel-TB using the TV
\zxhlp{TDEPEND}  Time-dependent imaging procedure sequence
\zxhlp{TVFLG}    interactive flagging of UV data using the TV
\zxhlp{UFLAG}    Plots and edits data using a uv-plane grid and the TV
\zxhlp{WIPER}    plots and edits data from a UV data base using the TV
\zxhlp{XBASL}    Fits and subtracts nth-order baselines from cube (x axis)
\zxhlp{XG2PL}    Plots spectrum of a pixel with \zxhlp{XGAUS}/\zxhlp{AGAUS} and \zxhlp{ZEMAN}/\zxhlp{ZAMAN} f
\zxhlp{XGAUS}    Fits 1-dimensional Gaussians to images: restartable
\zxhlp{XPLOT}    Plots image rows one at a time on the graphics or TV screen
\zxhlp{ZAMAN}    Fits 1-dimensional Zeeman model to absorption-line data
\zxhlp{ZEMAN}    Fits 1-dimensional Zeeman model to data
```

13.38 UTILITY

\zxhlp{ANTNUM}	Returns number of a named antenna
\zxhlp{CCEDT}	Select CC components in BOXes and above mininum flux.
\zxhlp{CCSEL}	Select signifigant CC components
\zxhlp{CL2HF}	Convert CL table to HF table
\zxhlp{DOOSRO}	calibrating amplitude and phase, and imaging \zxhlp{VLA} data
\zxhlp{DOVLAMP}	Produces amp calibration file for phased-\zxhlp{VLA} VLBI data
\zxhlp{EPOCONV}	Convert between J2000 and B1950 coordinates
\zxhlp{MAXTAB}	Returns maximum version number of named table
\zxhlp{MBDLY}	Fits multiband delays from IF phases, updates SN table
\zxhlp{MK3TX}	extract text files from a MKIII VLBI archive tape
\zxhlp{MOVE}	Task to copy or move data from one user or disk to another
\zxhlp{NANS}	reads an image or a UV data set and looks for NaNs
\zxhlp{OPCODE}	General adverb, defines an operation
\zxhlp{OPTELL}	The operation to be passed to a task by \zxhlp{TELL}
\zxhlp{PIPEAIPS}	calibrating amplitude and phase, and imaging \zxhlp{VLA} data
\zxhlp{PRNUMBER}	POPS number of messages
\zxhlp{PRTIME}	Time limit
\zxhlp{RUNWAIT}	Runs a task and waits for it to finish
\zxhlp{SCANTIME}	Returns time range for a given scan number
\zxhlp{SHOW}	Verblike to display the \zxhlp{TELL} adverbs of a task.
\zxhlp{SORT}	Specified desired sort order
\zxhlp{SQASH}	Task to collapse several planes in a cube into one plane or row
\zxhlp{STRAN}	Task compares ST tables, find image coordinates (e.g. guide star)
\zxhlp{SWAPR}	modifies UV data by swapping real and imaginary parts
\zxhlp{TBDIF}	Compare entries in two tables
\zxhlp{TBIN}	Reads a text file AIPS table into AIPS
\zxhlp{TBOUT}	Writes an AIPS table into a text file for user editting.
\zxhlp{TBSUB}	Make a new table from a subset of an old table
\zxhlp{TBTSK}	Paraform OOP task for tables
\zxhlp{TCOPY}	Tape to tape copy with some disk FITS support
\zxhlp{UVAVG}	Average or merge a sorted (BT, TB) uv database
\zxhlp{UVCMP}	Convert a UV database to or from compressed format
\zxhlp{UVDI1}	Subtract UV data(averaged up to one time) from the other UV data
\zxhlp{UVNOU}	flags uv samples near the U,V axes to reduce interference
\zxhlp{UVRFI}	Mitigate \zxhlp{RFI} by Fourier transform or fitting the circle
\zxhlp{UVWAX}	flags uv samples near the U,V axes to reduce interference
\zxhlp{VLAPROCS}	Procedures to simplify the reduction of VLBA data
\zxhlp{VLARUN}	calibrating amplitude and phase, and imaging \zxhlp{VLA} data
\zxhlp{VLASUMM}	Plots selected contents of SN or CL files
\zxhlp{VLBAAMP}	applies a-priori amplitude corrections to VLBA data
\zxhlp{VLBAARCH}	Procedure to archive VLBA correlator data
\zxhlp{VLBACALA}	applies a-priori amplitude corrections to VLBA data
\zxhlp{VLBACCOR}	applies a-priori amplitude corrections to VLBA data
\zxhlp{VLBACPOL}	Procedure to calibrate cross-polarization delays
\zxhlp{VLBACRPL}	Plots crosscorrelations
\zxhlp{VLBAEOPS}	Corrects Earth orientation parameters
\zxhlp{VLBAFIX}	Procedure that fixes VLBA data, if necessary
\zxhlp{VLBAFPOL}	Checks and corrects polarization labels for VLBA data
\zxhlp{VLBAFQS}	Copies different FQIDS to separate files
\zxhlp{VLBAFRGP}	Fringe fit phase referenced data and apply calibration
\zxhlp{VLBAFRNG}	Fringe fit data and apply calibration
\zxhlp{VLBAKRGP}	Fringe fit phase referenced data and apply calibration
\zxhlp{VLBAKRNG}	Fringe fit data and apply calibration
\zxhlp{VLBALOAD}	Loads VLBA data
\zxhlp{VLBAMCAL}	Merges redundant calibration data
\zxhlp{VLBAMPCL}	Calculates and applies manual instrumental phase calibration
\zxhlp{VLBAPANG}	Corrects for parallactic angle

\zxhlp{VLBAPCOR}	Calculates and applies instrumental phase calibration
\zxhlp{VLBAPIPE}	applies amplitude and phase calibration procs to VLBA data
\zxhlp{VLBARUN}	applies amplitude and phase calibration procs to VLBA data
\zxhlp{VLBASNPL}	Plots selected contents of SN or CL files
\zxhlp{VLBASRT}	Sorts VLBA data, if necessary
\zxhlp{VLBASUBS}	looks for subarrays in VLBA data
\zxhlp{VLBASUMM}	Prints a summary of a VLBI experiment
\zxhlp{VLBATECR}	Calculate ionospheric delay and Faraday rotation corrections
\zxhlp{VLBAUTIL}	Procedures to simplify the reduction of VLBA data

13.39 UV

\zxhlp{ACCOR}	Corrects cross amplitudes using auto correlation measurements
\zxhlp{ACLIP}	edits suto-corr data for amplitudes, phases, and weights out of range
\zxhlp{ACSCL}	Corrects cross amplitudes using auto correlation measurements
\zxhlp{ADDIF}	Adds an IF axis to a uv data set
\zxhlp{AFILE}	sorts and edits MkIII correlator A-file.
\zxhlp{ALIAS}	adverb to alias antenna numbers to one another
\zxhlp{ALVAR}	plots the Allan Variance statistic of a UV data set
\zxhlp{ALVPR}	prints statistics on the Allan Variance of a UV data set
\zxhlp{ANBPL}	plots and prints uv data converted to antenna based values
\zxhlp{ASDMFILE}	Full path to \zxhlp{EVLA} \zxhlp{ASDM}/BDF directory
\zxhlp{ATLOD}	Reads ATCA data in RPFITS dormat into AIPS
\zxhlp{AVER}	Averages over time UV data sets in 'BT' order
\zxhlp{AVOPTION}	Controls type or range of averaging done by a task
\zxhlp{AVSPC}	Averages uv-data in the frequency domain
\zxhlp{BAND}	specifies the approximate frequency of UV data to be selected
\zxhlp{BASFIT}	fits antenna locations from SN-table data
\zxhlp{BASRM}	Task to remove a spectral baseline from total power spectra
\zxhlp{BDF2AIPS}	Read \zxhlp{EVLA} \zxhlp{ASDM}/BDF data into AIPS
\zxhlp{BDFLIST}	List contents of \zxhlp{EVLA} ASD data file
\zxhlp{BIF}	gives first IF to be included
\zxhlp{BLAVG}	Average cross-polarized UV data over baselines.
\zxhlp{BLOAT}	converts line data to greater number channels
\zxhlp{BPASSPRM}	Control adverb array for bandpass calibration
\zxhlp{BPEDT}	Interactive TV task to edit uv data based on BP tables
\zxhlp{BPLOT}	Plots bandpass tables in 2 dimensions as function of time
\zxhlp{BPSMO}	Smooths or interpolates bandpass tables to regular times
\zxhlp{BPWAY}	Determines channel-dependent relative weights
\zxhlp{BPWGT}	Calibrates data and scales weights by bandpass correction
\zxhlp{BREAK}	procedure to \zxhlp{TELL} \zxhlp{FILLM} to break all current uv files, start new
\zxhlp{BSCAN}	seeks best scan to use for phase cal, fringe search, ..
\zxhlp{BSMOD}	creates single-dish UV beam-switched data with model sources
\zxhlp{BSROT}	modifies SD beam-switch continuum data for error in throw
\zxhlp{CALIB}	determines antenna calibration: complex gain
\zxhlp{CAPLT}	plots closure amplitude and model from CC file
\zxhlp{CENTR}	modifies UV data to center the reference channel
\zxhlp{CLIP}	edits data based on amplitudes, phases, and weights out of range
\zxhlp{CLPLT}	plots closure phase and model from CC file
\zxhlp{CMETHOD}	specifies the method by which the uv model is computed
\zxhlp{CMODEL}	specifies the method by which the uv model is computed
\zxhlp{COHER}	Baseline Phase coherence measurement
\zxhlp{CONFIG}	Configuration ID number within an \zxhlp{EVLA} \zxhlp{ASDM}/BDF data set
\zxhlp{CORER}	calculates correlator statistics and flags bad ones
\zxhlp{CORFQ}	corrects uvw for incorrect observing frequency
\zxhlp{CVEL}	shifts spectral-line UV data to a given velocity
\zxhlp{DAYFX}	Fixes day number problems left by \zxhlp{FILLM}

```

\zxhlp{DBAPP}      appends one or more data sets to the output data set
\zxhlp{DBCON}      concatenates two UV data sets
\zxhlp{DECOR}      Measures the decorrelation between channels and IF of uv data
\zxhlp{DEFER}      Controls when file creation takes place
\zxhlp{DEFLG}      edits data based on decorrelation over channels and time
\zxhlp{DESCM}      copies a portion of a UV data set
\zxhlp{DFQID}      modifies UV data changing the indicated FQIDs
\zxhlp{DFTIM}      Makes image of DFT at arbitrary point showing time vs frequency
\zxhlp{DFTPL}      plots DFT of a UV data set at arbitrary point versus time
\zxhlp{DIFRL}      divides the RR data by LL data
\zxhlp{DIFUV}      Outputs the difference of two matching input uv data sets
\zxhlp{DOACOR}     specifies whether autocorrelation data are included
\zxhlp{DOARRAY}    specifies if subarrays are ignored or the information used
\zxhlp{DOBWEEN}    Controls smoothing between sources in calibration tables
\zxhlp{DOCONCAT}   selects concatenated or individual output files
\zxhlp{DOEBAR}     Controls display of estimates of the uncertainty in the data
\zxhlp{DOIFS}       controls functions done across IFs
\zxhlp{DOROBUST}   Controls method of averaging - simple mean/rms or robust
\zxhlp{DOSTOKES}   selects options related to polarizations
\zxhlp{DOUVCOMP}   selects use of compression in writing UV data to disk
\zxhlp{DQUAL}       Rearranges source list, dropping qualifiers
\zxhlp{DSORC}       copies a data set eliminating some source numbers
\zxhlp{DSTOK}      Drops the cross-hand polarizations
\zxhlp{DTCHK}      Task to check results of a test using simulated data.
\zxhlp{DTSUM}      Task to provide a summary of the contents of a dataset
\zxhlp{EDITA}       Interactive TV task to edit uv data based on TY/SY/SN/CL tables
\zxhlp{EDITR}       Interactive baseline-oriented visibility editor using the TV
\zxhlp{EIF}        last IF number to be included in operation
\zxhlp{ELFIT}      Plots/fits selected contents of SN, TY, SY, PC or CL files
\zxhlp{EVASN}       Evaluates statistics in SN/CL tables
\zxhlp{EVAUV}      Subtracts & divides a model into UV data, does statistics on results
\zxhlp{EVLA}        puts the list of eVLA antennas in the current file on stack
\zxhlp{FEW}         procedure to \zxhlp{TELL} \zxhlp{FILLM} to append incoming data to existing uv files
\zxhlp{FGCNT}      Counts samples comparing two flag tables
\zxhlp{FGDIF}      Compares affect of 2 FG tables
\zxhlp{FGPLT}      Plots selected contents of FG table
\zxhlp{FGTAB}      Plots selected contents of FG table
\zxhlp{FILLM}      reads \zxhlp{VLA} on-line/archive format uv data tapes (post Jan 88)
\zxhlp{FILLR}      reads old \zxhlp{VLA} on-line-system tapes into AIPS
\zxhlp{FINDR}      Find normal values for a uv data set
\zxhlp{FITAB}      writes images / uv data w extensions to tape in FITS format
\zxhlp{FITDISK}    writes images / uv data w extensions to disk in FITS format
\zxhlp{FITTP}      writes images / uv data w extensions to tape in FITS format
\zxhlp{FIXAL}      least squares fit aliasing function and remove
\zxhlp{FIXAN}      fixes the contents of the ANtenna extension file
\zxhlp{FIXRL}      correctes right vs left polarizations for a list of antennas
\zxhlp{FIXWT}      Modify weights to reflect amplitude scatter of data
\zxhlp{FLAGR}      Edit data based on internal RMS, amplitudes, weights
\zxhlp{FLGIT}      flags data based on the rms of the spectrum
\zxhlp{FLOPM}      reverses the spectral order of UV data, can fix \zxhlp{VLA} error
\zxhlp{FOV}        Specifies the field of view
\zxhlp{FQCENTER}   specifies that the frequency axis should be centered
\zxhlp{FRMAP}      Task to build a map using fringe rate spectra
\zxhlp{FRPLT}      Task to plot fringe rate spectra
\zxhlp{FTFLG}      interactive flagging of UV data in channel-time using the TV
\zxhlp{FUDGE}      modifies UV data with user's algorithm: paraform task
\zxhlp{FXALIAS}    least squares fit aliasing function and remove
\zxhlp{FXPOL}      Corrects VLBA polarization assignments

```

13.39. UV

13. Current AIPS Software

```

\zxhlp{FXTIM}      fixes start date so all times are positive
\zxhlp{FXVLA}      Task to correct \zxhlp{VLA} data for on-line errors in special cases.
\zxhlp{FXVLB}      Builds a CQ table to enable VLBA correlator loss corrections
\zxhlp{GCPLT}      Plots gain curves from text files
\zxhlp{HA2TI}      Converts data processed by \zxhlp{TI2HA} (STUFFER) back to real times
\zxhlp{HAFIX}      Recomputes u,v,w when time is hour angle (UVdata is output of \zxhlp{TI2HA})
HLPEDIBP   Interactive BP table uv-data editor \zxhlp{BPEDT} - run-time help
HLPEDICL   Interactive SN/CL table uv-data editor - run-time help
HLPEDIPC   Interactive PC table editor \zxhlp{PCEDT} - run-time help
HLPEDISN   Interactive SN/CL table (not UV) editor - run-time help
HLPEDISS   Interactive SY table (not UV) editor - run-time help
HLPEDISY   Interactive SY table uv-data editor \zxhlp{EDITA} - run-time help
HLPEDITS   Interactive TY table (not UV) editor - run-time help
HLPEDITY   Interactive TY table uv-data editor \zxhlp{EDITA} - run-time help
HLPEDIUV   Interactive uv-data editor \zxhlp{EDITR} - run-time help
HLPFTFLG   Interactive time-channel visibility Editor - run-time help
HLPIBLED   Interactive Baseline based visibility Editor - run-time help
HLPPCFLG   Interactive time-channel PC table Editor \zxhlp{PCFLG} - run-time help
HLPSPFLG   Interactive time-channel visibility Editor \zxhlp{SPFLG} - run-time help
HLPTVFLG   Interactive time-baseline visibility Editor \zxhlp{TVFLG} - run-time help
HLPFLAG    Edit uv-data on a grid \zxhlp{UFLAG} - run-time help
HLPWIPER   edit uv data from \zxhlp{UVPLT}-like plot \zxhlp{WIPER} - run-time help
\zxhlp{HOLGR}   Read & process holography visibility data to telescope images
\zxhlp{HOLOG}   Read & process holography visibility data to telescope images
\zxhlp{HSA}      puts the list of \zxhlp{HSA} antennas in the current file on stack
\zxhlp{IBLED}   Interactive BaseLine based visibility EDitor
\zxhlp{IM2UV}   converts an image to a visibility data set
\zxhlp{IMFRING} large image delay fitting with \zxhlp{IM2CC} and \zxhlp{O0FRING}
\zxhlp{IMSCAL}  large image self-cal with \zxhlp{IM2CC} and \zxhlp{OOCAL}
\zxhlp{LISTR}   prints contents of UV data sets and assoc. calibration tables
\zxhlp{LOCIT}   fits antenna locations from SN-table data
\zxhlp{LPCAL}   Determines instrumental polarization for UV data
\zxhlp{M3STAR}  translate Haystack MKIII VLBI format "A" TAR's into AIPS
\zxhlp{MANY}    procedure to \zxhlp{TELL} \zxhlp{FILLM} to start new uv files on each scan
\zxhlp{MAPBM}   Map \zxhlp{VLA} beam polarization
\zxhlp{MATCH}   changes antenna, source, FQ numbers to match a data set
\zxhlp{MK3IN}   translate Haystack MKIII VLBI format "A" tapes into AIPS
\zxhlp{MORIF}   Combines IFs or breaks spectral windows into multiple windows (IFs)
\zxhlp{MSORT}   Sort a UV dataset into a specified order
\zxhlp{MULIF}   Change number of IFs in output
\zxhlp{MULTI}   Task to convert single-source to multi-source UV data
\zxhlp{NANS}    reads an image or a UV data set and looks for NaNs
\zxhlp{NCHAN}   Number of spectral channels in each spectral window
\zxhlp{NIF}     Number of IFs (spectral windows) in a data set
\zxhlp{NOIFS}   makes all IFs into single spectrum
\zxhlp{NPIECE}  The number of pieces to make
\zxhlp{OBJECT}  The name of an object
\zxhlp{OBPLT}   Plot columns of an OB table.
\zxhlp{OMFIT}   Fits sources and, optionally, a self-cal model to uv data
\zxhlp{OOCAL}   determines antenna complex gain with frequency-dependent models
\zxhlp{OOSRT}   Sort a UV dataset into a specified order
\zxhlp{OOSUB}   Subtracts/divides a model from/into a uv data base
\zxhlp{PCAL}    Determines instrumental polarization for UV data
\zxhlp{PCASS}   Finds amplitude bandpass shape from pulse-cal table data
\zxhlp{PCCOR}   Corrects phases using \zxhlp{PCAL} tones data from PC table
\zxhlp{PCEDT}   Interactive TV task to edit PC tables
\zxhlp{PCFIT}   Finds delays and phases using a PC table
\zxhlp{PCFLG}   interactive flagging of Pulse-cal data in channel-TB using the TV

```

```

\zxhlp{PCPLT}      Plots pulse-cal tables in 2 dimensions as function of time
\zxhlp{PCVEL}      shifts spectral-line UV data to a given velocity: planet version
\zxhlp{PEELR}      calibrates interfering sources in multi-facet images
\zxhlp{PHASE}      Baseline Phase coherence measurement
\zxhlp{PHNEG}      Negates a UV datafile's visibility phase.
\zxhlp{PHSLIMIT}   gives a phase value in degrees
\zxhlp{PHSRF}      Perform phase-referencing within a spectral line database.
\zxhlp{POSSM}      Task to plot total and cross-power spectra.
\zxhlp{PRTAN}      prints the contents of the ANtenna extension file
\zxhlp{PRTUV}      prints contents of a visibility (UV) data set
\zxhlp{QCREATE}    adverb controlling the way large files are created
\zxhlp{QUACK}      Flags beginning or end portions of UV-data scans
\zxhlp{QUAL}       Source qualifier
\zxhlp{QUFIX}      determines Right minus Left phase difference, corrects cal files
\zxhlp{QUIT}       procedure to \zxhlp{TELL} \zxhlp{FILLM} to stop at the end of the current scan
\zxhlp{QUOUT}      writes text file of Q, U versus frequency to be used by \zxhlp{RLDIF}
\zxhlp{READISK}   writes images / uv data w extensions to tape in FITS format
\zxhlp{REAMP}      modifies UV data re-scaling the amplitudes
\zxhlp{REFDATE}   To specify the initial or reference date of a data set
\zxhlp{REFLG}      Attempts to compress a flag table
\zxhlp{REIFS}      Breaks spectral windows into multiple spectral windows (IFs)
\zxhlp{RESEQ}      Renumber antennas
\zxhlp{REWAVY}    computes weights based in rms in spectra
\zxhlp{REWEIGHT}  Reweighting factors for UV data weights.
\zxhlp{REWGT}     modifies UV data re-scaling the weights only
\zxhlp{RFLAG}     Flags data set based on time and freq rms in fringe visibilities
\zxhlp{RLCAL}     Determines instrumental right-left phase versus time (a self-cal)
\zxhlp{RLCOR}     corrects a data set for R-L phase differences
\zxhlp{RLDIF}     determines Right minus Left phase difference, corrects cal files
\zxhlp{ROBUST}    Uniform weighting "robustness" parameter
\zxhlp{SBCOR}     Task to correct VLBA data for phase shift between USB & LSB
\zxhlp{SDLSF}     least squares fit to channels and subtracts from SD uv data
\zxhlp{SDMOD}     modifies single-dish UV data with model sources
\zxhlp{SDVEL}    shifts spectral-line single-dish data to a given velocity
\zxhlp{SETAN}     Reads an ANtenna file info from a text file
\zxhlp{SHADO}     Calculate the shadowing of antennas at the array
\zxhlp{SHOUV}    displays uv data in various ways.
\zxhlp{SMOOTH}   Specifies spectral smoothing
\zxhlp{SNEDT}    Interactive SN/CL/TY/SY table editor using the TV
\zxhlp{SNFIT}    Fits parabola to SN amplitudes and plots result
\zxhlp{SNFLG}    Writes flagging info based on the contents of SN files
\zxhlp{SNIFS}    Plots selected contents of SN, TY, SY, PC or CL files
\zxhlp{SNPLT}    Plots selected contents of SN, SY, TY, PC or CL files
\zxhlp{SNREF}    Chooses best reference antenna to minimize R-L differences
\zxhlp{SORT}     Specified desired sort order
\zxhlp{SOUSP}    fits source spectral index from SU table or adverbs
\zxhlp{SPCAL}    Determines instrumental polzn. for spec. line UV data
\zxhlp{SPECRR}  Spectral regridding task for UV data
\zxhlp{SPFLG}    interactive flagging of UV data in channel-TB using the TV
\zxhlp{SPLAT}    Applies calibration and splits or assemble selected sources.
\zxhlp{SPLIT}    converts multi-source to single-source UV files w calibration
\zxhlp{SPMOD}    Modify UV database by adding a model with spectral lines
\zxhlp{STOKES}   Stokes parameter
\zxhlp{STOP}     procedure to \zxhlp{TELL} \zxhlp{FILLM} to break all current uv files and stop
\zxhlp{STUFFR}   averages together data sets in hour angle
\zxhlp{SUFIX}    modifies source numbers on uv data
\zxhlp{SWAPR}    modifies UV data by swapping real and imaginary parts
\zxhlp{SWPOL}   Swap polarizations in a UV data base

```

```

\zxhlp{TBAVG} Time averages data combining all baselines.
\zxhlp{TEFLM} procedure to \zxhlp{TELL} real-time \zxhlp{FILLM} a new \zxhlp{APARM}(1) value
\zxhlp{TFILE} sorts and edits MkIII correlator UNIX-based A-file.
\zxhlp{TI2HA} modifies times in UV data to hour angles
\zxhlp{TIORD} checks data for time baseline ordering, displays failures
\zxhlp{TRUEP} determines true antenna polarization from special data sets
\zxhlp{TVFLG} interactive flagging of UV data using the TV
\zxhlp{U2CAT} list a user's UV and scratch files on disk \zxhlp{IN2DISK}
\zxhlp{U3CAT} list a user's UV and scratch files on disk \zxhlp{IN3DISK}
\zxhlp{U4CAT} list a user's UV and scratch files on disk \zxhlp{IN4DISK}
\zxhlp{U5CAT} list a user's UV and scratch files on disk \zxhlp{IN5DISK}
\zxhlp{UBAVG} Baseline dependent time averaging of uv data
\zxhlp{UCAT} list a user's UV and scratch files on disk \zxhlp{INDISK}
\zxhlp{UFLAG} Plots and edits data using a uv-plane grid and the TV
\zxhlp{UJOIN} modifies UV data converting IFs to spectral channels
\zxhlp{UOCAT} list a user's UV and scratch files on disk \zxhlp{OUTDISK}
\zxhlp{USUBA} Assign subarrays within a uv-data file
\zxhlp{UV1TYPE} Convolving function type 1, pillbox or square wave
\zxhlp{UV2MS} Append single-source file to multi-source file.
\zxhlp{UV2TB} Converts UV autocorrelation spectra to tables
\zxhlp{UV2TYPE} Convolving function type 2, exponential function
\zxhlp{UV3TYPE} Convolving function type 3, sinc function
\zxhlp{UV4TYPE} Convolving function type 4, exponent times sinc function
\zxhlp{UV5TYPE} Convolving function type 5, spheroidal function
\zxhlp{UV6TYPE} Convolving function type 6, exponent times BessJ1(x) / x
\zxhlp{UVADC} Fourier transforms and corrects a model and adds to uv data.
\zxhlp{UVAVG} Average or merge a sorted (BT, TB) uv database
\zxhlp{UVBAS} averages several channels and subtracts from uv data.
\zxhlp{UVBOX} radius of the smoothing box used for uniform weighting
\zxhlp{UVBXFN} type of function used when counting for uniform weighting
\zxhlp{UVCMP} Convert a UV database to or from compressed format
\zxhlp{UVCON} Generate sample UV coverage given a user defined array layout
\zxhlp{UVCOP} Task to copy a subset of a UV data file
\zxhlp{UVCOPPRM} Parameter adverb array for task \zxhlp{UVCOP}
\zxhlp{UVCRS} Finds the crossing points of UV-ellipses.
\zxhlp{UVDEC} Decrement the number of spectral channels, keeping every nth
\zxhlp{UDGP} Copy a UV data file, deleting a portion of it
\zxhlp{UVDI1} Subtract UV data(averaged up to one time) from the other UV data
\zxhlp{UVDIF} prints differences between two UV data sets
\zxhlp{UVFIL} Create, fill a uv database from user supplied information
\zxhlp{UVFIT} Fits source models to uv data.
\zxhlp{UVFIX} Recomputes u,v,w for a uv database
\zxhlp{UVFIXPRM} Parameter adverb array for task \zxhlp{UVFIX}
\zxhlp{UVFLG} Flags UV-data
\zxhlp{UVFND} prints selected data from UV data set to search for problems
\zxhlp{UVFRE} Makes one data set have the spectral structure of another
\zxhlp{UVGIT} Fits source models to uv data.
\zxhlp{UVGLU} Glues UV data frequency blocks back together
\zxhlp{UVHGM} Plots statistics of uv data files as histogram.
\zxhlp{Uvhim} Makes image of the histogram on two user-chosen axes
\zxhlp{UVHOL} prints holography data from a UV data base with calibration
\zxhlp{UVIMG} Grid UV data into an "image"
\zxhlp{UVLIN} Fits and removes continuum visibility spectrum, also can flag
\zxhlp{UVLOD} Read export or FITS data from a tape or disk
\zxhlp{UVLSD} least squares fit to channels and divides the uv data.
\zxhlp{UVLSF} least squares fit to channels and subtracts from uv data.
\zxhlp{UVMLN} edits data based on the rms of the spectrum
\zxhlp{UVMOD} Modify UV database by adding a model incl spectral index

```

\zxhlp{UVMTH}	Averages one data set and applied it to another.
\zxhlp{UVNOU}	flags uv samples near the U,V axes to reduce interference
\zxhlp{UVPLT}	plots data from a UV data base
\zxhlp{UVPOL}	modifies UV data to make complex image and beam
\zxhlp{UVPRM}	measures parameters from a UV data base
\zxhlp{UVPRT}	prints data from a UV data base with calibration
\zxhlp{UVRANGE}	Specify range of projected baselines
\zxhlp{UVRFI}	Mitigate \zxhlp{RFI} by Fourier transform or fitting the circle
\zxhlp{UVSEN}	Determine RMS sidelobe level and brightness sensitivity
\zxhlp{UVSIM}	Generate sample UV coverage given a user defined array layout
\zxhlp{UVSIZE}	specifies number of pixels on X and Y axes of a UV image
\zxhlp{UVSRT}	Sort a UV dataset into a specified order
\zxhlp{UVSUB}	Subtracts/divides a model from/into a uv data base
\zxhlp{UVTAPER}	Widths in U and V of gaussian weighting taper function
\zxhlp{UVWAX}	flags uv samples near the U,V axes to reduce interference
\zxhlp{UVWTFN}	Specify weighting function, Uniform or Natural
\zxhlp{VBGLU}	Glues together data from multiple passes thru the VLBA corr.
\zxhlp{VECTOR}	selects method of averaging UV data
\zxhlp{VLA}	puts the list of \zxhlp{VLA} antennas in the current file on stack
VLBA	puts the list of VLBA antennas in the current file on stack
\zxhlp{VLBIN}	Task to read VLBI data from an NRAO/MPI MkII correlator
\zxhlp{VPFLG}	Resets flagging to all or all corss-hand whenever some are flagged
\zxhlp{VPLOT}	plots uv data and model from CC file
\zxhlp{VTEST}	Measures velocity discrepancy across fields
\zxhlp{WEIGHTIT}	Controls modification of weights before gain/fringe solutions
\zxhlp{WETHR}	Plots selected contents of WX tables, flags data based on WX
\zxhlp{WIPER}	plots and edits data from a UV data base using the TV
\zxhlp{WRDISK}	writes images / uv data w extensions to tape in FITS format
\zxhlp{WRTPROCS}	Procedures to simplify the reduction of VLBA data
\zxhlp{WTMOD}	modifies weights in a UV data set
\zxhlp{WTUV}	Specifies the weight to use for UV data outside \zxhlp{UVRANGE}
\zxhlp{ZEROESP}	Specify how to include zero spacing fluxes in FT of UV data

13.40 VERB

\zxhlp{VERB}

Type: General type of POPS symbol

Use: Verbs are the magic words which cause FORTRAN code to execute some function. They are compiled into AIPS by the programmers and their meaning remains fixed at least until the programmers change their minds.

Grammar: Verbs may be given either in compile mode or in regular execute mode. In the former, their pointers are stored with the procedure and they are executed when the procedure is invoked. In the latter, they are compiled with the other statements and parameters on the input line and then executed before a new input line is read.

Execution: Verbs are executed when the line in which they appear is executed and are simply referenced by their name. The syntax "\zxhlp{GO} verb_name" is converted by AIPS to "\zxhlp{TPUT} verb_name ; verb_name" which saves the adverbs of "verb_name" for a later \zxhlp{TGET} and then executes "verb_name". The syntax "\zxhlp{TASK} = 'verb_name' ; \zxhlp{GO}" will not work.

13.40. VERB

13. Current AIPS Software

\zxhlp{ABOUT}	displays lists and information on tasks, verbs, adverbs
\zxhlp{ABS}	returns absolute value of argument
\zxhlp{ACOS}	Returns arc cosine of argument (half-circle)
\zxhlp{ACTNOISE}	puts estimate of actual image uncertainty and zero in header
\zxhlp{ADDBEAM}	Inserts clean beam parameters in image header
\zxhlp{ADDDISK}	makes a computer's disks available to the current AIPS session
\zxhlp{ALLDEST}	Delete a group or all of a users data files
\zxhlp{ALTDEF}	Sets frequency vs velocity relationship into image header
\zxhlp{ALTSWTCH}	Switches between frequency and velocity in image header
\zxhlp{APROPOS}	displays all help 1-line summaries containing specified words
\zxhlp{ASIN}	Returns arc sine of argument (half-circle)
\zxhlp{ATAN2}	Returns arc tangent of two arguments (full circle)
\zxhlp{ATAN}	Returns arc tangent of argument (half-circle)
\zxhlp{AVEOT}	Advances tape to end-of-information point
\zxhlp{AVFILE}	Moves tape forward or back to end-of-file marks
\zxhlp{AVMAP}	Advance tape by one image (IBM-CV = obsolete tape file)
\zxhlp{AXDEFINE}	Define or modify an image axis description
\zxhlp{BAMODIFY}	edits characters in a line of a batch work file
\zxhlp{BATCH}	starts entry of commands into batch-job work file
\zxhlp{BATCLEAR}	removes all text from a batch work file
\zxhlp{BATEDIT}	starts an edit (replace, insert) session on a batch work file
\zxhlp{BATLIST}	lists the contents of a batch work file
\zxhlp{BDFLIST}	List contents of \zxhlp{EVLA} ASD data file
\zxhlp{BY}	gives increment to use in \zxhlp{FOR} loops in POPS language
\zxhlp{CALDIR}	lists calibrator source models available as AIPS FITS files
\zxhlp{CATALOG}	list one or more entries in the user's data directory
\zxhlp{CEIL}	returns smallest integer greater than or equal the argument
\zxhlp{CELGAL}	switches header between celestial and galactic coordinates
\zxhlp{CHAR}	converts number to character string
\zxhlp{CHARMULT}	Changes the multiplication factor for TV characters
\zxhlp{CHKNAME}	Checks for existence of the specified image name
\zxhlp{CLR2NAME}	clears adverbs specifying the second input image
\zxhlp{CLR3NAME}	clears adverbs specifying the third input image
\zxhlp{CLR4NAME}	clears adverbs specifying the fourth input image
\zxhlp{CLR5NAME}	clears adverbs specifying the fifth input image
\zxhlp{CLRMSG}	deletes messages from the user's message file
\zxhlp{CLRNAME}	clears adverbs specifying the first input image
\zxhlp{CLRONAME}	clears adverbs specifying the first output image
\zxhlp{CLRSTAT}	remove any read or write status flags on a directory entry
\zxhlp{CLRTEMP}	clears the temporary literal area during a procedure
\zxhlp{CODECIML}	Convert between decimal and sexagesimal coordinate values
\zxhlp{COODEFIN}	Define or modify an image axis coordinate description
\zxhlp{COPIXEL}	Convert between physical and pixel coordinate values
\zxhlp{COS}	returns cosine of the argument in degrees
\zxhlp{COSTAR}	Verb to plot a symbol at given position on top of a TV image
\zxhlp{COWINDOW}	Set a window based on coordinates
\zxhlp{CPUTIME}	displays current tcpu and real time usage of the AIPS task
\zxhlp{CURBLINK}	switch TV cursor between steady and blinking displays
\zxhlp{CURVALUE}	displays image intensities selected via the TV cursor
\zxhlp{DAYNUMBR}	finds day number of an image or uv data set
\zxhlp{DEFAULT}	Verb-like sets adverbs for a task or verb to initial values
\zxhlp{DELAY}	Verb to pause AIPS for \zxhlp{DETIME} seconds
\zxhlp{DELBOX}	Verb to delete boxes with TV cursor & graphics display.
\zxhlp{DFILEBOX}	Verb to delete Clean boxes with TV cursor & write to file
\zxhlp{DISMOUNT}	disables a magnetic tape and dismounts it from the tape drive
\zxhlp{DRAWBOX}	Verb to draw Clean boxes on the display
\zxhlp{DUMP}	displays portions of the POPS symbol table in all formats
\zxhlp{EGETHHEAD}	returns parameter value from image header and error code

```

\zxhlp{EGETNAME} fills in input name adverbs by catalog slot number, w error
\zxhlp{EHEX} converts decimal to extended hex
\zxhlp{END} marks end of block (\zxhlp{FOR}, \zxhlp{WHILE}, IF) of POPS code
\zxhlp{EPOCONV} Convert between J2000 and B1950 coordinates
\zxhlp{EPOSWTCH} Switches between B1950 and J2000 coordinates in header
\zxhlp{EVLA} puts the list of eVLA antennas in the current file on stack
\zxhlp{EXIT} ends an AIPS batch or interactive session
\zxhlp{EXP} returns the exponential of the argument
\zxhlp{EXPLAIN} displays help + extended information describing a task/symbol
\zxhlp{EXTDEST} deletes one or more extension files
\zxhlp{EXTLIST} lists detailed information about contents of extension files
\zxhlp{FILEBOX} Verb to reset Clean boxes with TV cursor & write to file
\zxhlp{FILEZAP} Delete an external file
\zxhlp{FLOOR} returns largest integer <= argument
\zxhlp{FOR} starts an iterative sequence of operations in POPS language
\zxhlp{FREESPAC} displays available disk space for AIPS in local system
\zxhlp{GAMMASET} changes the gamma-correction exponent used in the TV OFM
\zxhlp{GET2NAME} fills 2nd input image name parameters by catalog slot number
\zxhlp{GET3NAME} fills 3rd input image name parameters by catalog slot number
\zxhlp{GET4NAME} fills 4th input image name parameters by catalog slot number
\zxhlp{GET5NAME} fills 5th input image name parameters by catalog slot number
\zxhlp{GETDATE} Convert the current date and time to a string
\zxhlp{GETHEAD} returns parameter value from image header
\zxhlp{GETNAME} fills 1st input image name parameters by catalog slot number
\zxhlp{GETONAME} fills 1st output image name parameters by catalog slot number
\zxhlp{GETPOPSN} Verb to return the pops number on the stack
\zxhlp{GETTHEAD} returns keyword and other values value from a table header
\zxhlp{GETVERS} finds maximum version number of an extension file
\zxhlp{GO} starts a task, detaching it from AIPS or \zxhlp{AIPSB}
\zxhlp{GRBLINK} Verb which blinks 2 TV graphics planes
\zxhlp{GRCLEAR} clears the contents of the specified TV graphics channels
\zxhlp{GRDROP} deletes the specified gripe entry
\zxhlp{GREAD} reads the colors of the specified TV graphics channel
\zxhlp{GRINDEX} lists users and time of all gripe entries
\zxhlp{GRIPE} enter a suggestion or bug report for the AIPS programmers
\zxhlp{GRLIST} lists contents of specified gripe entry
\zxhlp{GROFF} turns off specified TV graphics channels
\zxhlp{GRON} turns on specified TV graphics channels
\zxhlp{GWRITE} reads the colors of the specified TV graphics channel
\zxhlp{HELP} displays information on tasks, verbs, adverbs
\zxhlp{HINOTE} adds user-generated lines to the history extension file
\zxhlp{HITEXT} writes lines from history extension file to text file
\zxhlp{HSA} puts the list of \zxhlp{HSA} antennas in the current file on stack
\zxhlp{HUEEDGE} Show a wedge on the TV suitable for \zxhlp{TVHUEINT} displays
\zxhlp{IM2HEAD} displays the image 2 header contents to terminal, message file
\zxhlp{IM2TV} Verb to convert pixel coordinates to TV pixels
\zxhlp{IM3HEAD} displays the image 3 header contents to terminal, message file
\zxhlp{IM4HEAD} displays the image 4 header contents to terminal, message file
\zxhlp{IM5HEAD} displays the image 5 header contents to terminal, message file
\zxhlp{IMCENTER} returns pixel position of sub-image centroid
\zxhlp{IMDIST} determines spherical distance between two pixels
\zxhlp{IMERASE} replaces an image portion of the TV screen with zeros
\zxhlp{IMHEADER} displays the image header contents to terminal, message file
\zxhlp{IMOHEAD} displays the output image header contents
\zxhlp{IMPOS} displays celestial coordinates selected by the TV cursor
\zxhlp{IMSTAT} returns statistics of a sub-image
\zxhlp{IMVAL} returns image intensity and coordinate at specified pixel
\zxhlp{IMWEDGE} load step wedge of full range of image values to TV

```

13.40. VERB

13. Current AIPS Software

```

\zxhlp{IMXY}           returns pixel coordinates selected by the TV cursor
\zxhlp{INP}             displays adverb values for task, verb, or proc - quick form
\zxhlp{INPUTS}          displays adverb values for task, verb, or proc - to msg file
\zxhlp{JOBLIST}         lists contents of a submitted and pending batch job
\zxhlp{KLEENEX}         ends an AIPS interactive session wiping the slate klean
\zxhlp{LENGTH}          returns length of string to last non-blank character
\zxhlp{LN}              returns the natural logarithm of the argument
\zxhlp{LOG}             returns the base-10 logarithm of the argument
\zxhlp{M2CAT}            displays images in the user's catalog directory for \zxhlp{IN2DISK}
\zxhlp{M3CAT}            displays images in the user's catalog directory for \zxhlp{IN3DISK}
\zxhlp{M4CAT}            displays images in the user's catalog directory for \zxhlp{IN4DISK}
\zxhlp{M5CAT}            displays images in the user's catalog directory for \zxhlp{IN5DISK}
\zxhlp{MAXFIT}           returns pixel position and image intensity at a maximum
\zxhlp{MAX}              returns the maximum of its two arguments
\zxhlp{MCAT}             lists images in the user's catalog directory on disk \zxhlp{INDISK}
\zxhlp{MFITSET}          gets adverbs for running \zxhlp{IMFIT} and \zxhlp{JMFIT}
\zxhlp{MIN}              returns the minimum of its two arguments
\zxhlp{MOCAT}            displays images in the user's catalog directory for \zxhlp{OUTDISK}
\zxhlp{MOD}               returns remainder after division of 1st argument by 2nd
\zxhlp{MODULUS}          returns square root of sum of squares of its two arguments
\zxhlp{MOUNT}             makes a tape drive available to user's AIPS and tasks
\zxhlp{NAMEGET}           fills 1st input image name parameters by default matching
\zxhlp{OBITIMAG}          Access to \zxhlp{OBIT} task Imager without self-cal or peeling
\zxhlp{OBITMAP}           Simplified access to \zxhlp{OBIT} task Imager
\zxhlp{OBITPEEL}          Access to \zxhlp{OBIT} task Imager with self-cal and peeling
\zxhlp{OBITSCL}           Access to \zxhlp{OBIT} task Imager with self-cal, NOT peeling
\zxhlp{OFFHUINT}           Proc which restores TV functions to normal after \zxhlp{TVHUE}
\zxhlp{OFFPSEUD}          Verb which deactivates all pseudo-color displays
\zxhlp{OFFSCROL}          Verb which deactivates scroll of an image
\zxhlp{OFFTRAN}           Verb which restores transfer function to normal
\zxhlp{OFFZOOM}            Verb which returns the hardware IIS zoom to normal
\zxhlp{OFMADJUS}          interactive linear adjustment of current TV OFM lookup tables
\zxhlp{OFMCONT}            creates/modifies TV color OFMs with level or wedged contours
\zxhlp{OFMDIR}             lists names of the user's and system's OFM files from \zxhlp{OFMFIL}
\zxhlp{OFMGET}              loads TV \zxhlp{OFMS} from an OFM save file
\zxhlp{OFMLIST}            lists the current TV OFM table(s) on the terminal or printer
\zxhlp{OFMSAVE}             saves the TV's current OFM lookup table in a text file
\zxhlp{OFMTWEAK}           interactive modification of current TV OFM lookup tables
\zxhlp{OFMZAP}              deletes an OFM lookup table save file
\zxhlp{OUTPUTS}             displays adverb values returned from task, verb, or proc
\zxhlp{PARALLEL}            Verb to set or show degree of parallelism
\zxhlp{PASSWORD}           Verb to change the current password for the login user
\zxhlp{PCAT}                Verb to list entries in the user's catalog (no log file).
\zxhlp{PLGET}                gets the adverbs used to make a particular plot file
\zxhlp{PRINTER}              Verb to set or show the printer(s) used
\zxhlp{PRINT}                 Print the value of an expression
\zxhlp{PROCEDUR}            Define a POPS procedure using procedure editor
\zxhlp{PROC}                  Define a POPS procedure using procedure editor.
\zxhlp{PRTHI}                 prints selected contents of the history extension file
\zxhlp{PRTMSG}                 prints selected contents of the user's message file
\zxhlp{PSEUDOVB}              Declares a name to be a symbol of type pseudoverb
\zxhlp{PUTHEAD}                 Verb to modify image header parameters.
\zxhlp{PUTTHEAD}              inserts a given value into a table keyword/value pair
\zxhlp{PUTVALUE}                Verb to store a pixel value at specified position
\zxhlp{Q2HEADER}                 Verb to summarize the image 2 header: positions at center
\zxhlp{Q3HEADER}                 Verb to summarize the image 3 header: positions at center
\zxhlp{Q4HEADER}                 Verb to summarize the image 4 header: positions at center
\zxhlp{Q5HEADER}                 Verb to summarize the image 5 header: positions at center

```

```

\zxhlp{QGETVERS} finds maximum version number of an extension file quietly
\zxhlp{QHEADER} Verb to summarize the image header: positions at center
\zxhlp{QIMVAL} returns image intensity and coordinate at specified pixel
\zxhlp{QINP} displays adverb values for task, verb, or proc - restart form
\zxhlp{QOHEADER} Verb to summarize the output image header: center positions
\zxhlp{QUEUES} Verb to list all submitted jobs in the job queue
\zxhlp{RANDOM} Compute a random number from 0 to 1
\zxhlp{READ} Read a value from the users terminal
\zxhlp{REBOX} Verb to reset boxes with TV cursor & graphics display.
\zxhlp{RECAT} Verb to compress the entries in a catalog file
\zxhlp{REHEX} converts extended hex string to decimal
\zxhlp{REMDISK} removes a computer's disks from the current AIPS session
\zxhlp{REMOVIE} Verb to rerun a previously loaded (\zxhlp{TVMOVIE}) movie
\zxhlp{RENAME} Rename a file (UV or Image)
\zxhlp{RENUMBER} Verb to change the catalog number of an image.
\zxhlp{REROAM} Verb to use previous roam image mode, then does roam
\zxhlp{RESCALE} Verb to modify image scale factor and offset
\zxhlp{RESTART} Verb to trim the message log file and restart AIPS
\zxhlp{RESTORE} Read POPS memory file from a common area.
\zxhlp{RETURN} Exit a procedure allowing a higher level proc to continue.
\zxhlp{REVERSN} checks disk for presence of extension files
\zxhlp{REWIND} Verb to rewind a tape
\zxhlp{ROAM} Roam around an image too large for the display.
\zxhlp{ROAMOFF} Verb to recover image from roam display in simple display mode
\zxhlp{RUN} Pseudoverb to read an external \zxhlp{RUN} files into AIPS.
\zxhlp{SAVDEST} Verb to destroy all save files of a user.
\zxhlp{SAVE} Pseudoverb to save full POPS environment in named file
\zxhlp{SCALAR} Declares a variable to be a scalar in a procedure
\zxhlp{SCRATCH} delete a procedure from the symbol table.
\zxhlp{SCRDEST} Verb to destroy scratch files left by bombed tasks.
\zxhlp{SET1DG} Verb to set 1D gaussian fitting initial guesses.
\zxhlp{SETDEBUG} Verb to set the debug print and execution level
\zxhlp{SETMAXAP} Examines/alters system parameter limiting dynamic pseudo-AP
\zxhlp{SETROAM} Verb use to set roam image mode, then do roam. OBSOLETE
\zxhlp{SETSILICE} Set slice endpoints on the TV interactively
\zxhlp{SG2RUN} Verb copies the K area to a text file suitable for \zxhlp{RUN}
\zxhlp{SGDESTR} Verb-like to destroy named POPS environment save file
\zxhlp{SGINDEX} Verb lists \zxhlp{SAVE} areas by name and time of last \zxhlp{SAVE}.
\zxhlp{SHOW} Verblike to display the \zxhlp{TELL} adverbs of a task.
\zxhlp{SIN} Compute the sine of a value
\zxhlp{SIZEFILE} return file size plus estimate of \zxhlp{IMAGR} work file size
\zxhlp{SPY} Verb to determine the execution status of all AIPS tasks
\zxhlp{SQRT} Square root function
\zxhlp{STALIN} revises history by deleting lines from history extension file
\zxhlp{STQUEUE} Verb to list pending \zxhlp{TELL} operations
\zxhlp{STRING} Declare a symbol to be a string variable in POPS
\zxhlp{SUBMIT} Verb which submits a batch work file to the job queue
\zxhlp{SUBSTR} Function verb to specify a portion of a \zxhlp{STRING} variable
\zxhlp{SYSTEM} Verb to send a command to the operating system
\zxhlp{T1VERB} Temporary verb for testing (also T2VERB...T9VERB)
\zxhlp{TABGET} returns table entry for specified row, column and subscript.
\zxhlp{TAPPUT} replaces table entry for specified row, column and subscript.
\zxhlp{TAN} Tangent function
\zxhlp{TAPES} Verb to show the \zxhlp{TAPES}(s) available
\zxhlp{TGET} Verb-like gets adverbs from last \zxhlp{GO} of a task
\zxhlp{TGINDEX} Verb lists those tasks for which \zxhlp{TGET} will work.
\zxhlp{THEN} Specified the action if an IF test is true
\zxhlp{TIMDEST} Verb to destroy all files which are too old

```

13.40. VERB

13. Current AIPS Software

```

\zxhlp{TK1SET}           Verb to reset 1D gaussian fitting initial guess.
\zxhlp{TKAGUESS}         Verb to re-plot slice model guess directly on TEK
\zxhlp{TKAMODEL}          Verb to add slice model display directly on TEK
\zxhlp{TKARESID}          Verb to add slice model residuals directly on TEK
\zxhlp{TKASLICE}          Verb to add a slice display on TEK from slice file
\zxhlp{TKERASE}            Erase the graphics screen or window
\zxhlp{TKGUESS}            Verb to display slice model guess directly on TEK
\zxhlp{TKMODEL}             Verb to display slice model directly on TEK
\zxhlp{TKPOS}               Read a position from the graphics screen or window
\zxhlp{TKRESID}             Verb to display slice model residuals directly on TEK
\zxhlp{TKSET}                Verb to set 1D gaussian fitting initial guesses.
\zxhlp{TKSLICE}              Verb to display slice file directly on TEK
\zxhlp{TKVAL}                 Verb to obtain value under cursor from a slice
\zxhlp{TKXY}                  Verb to obtain pixel value under cursor
\zxhlp{TO}                     Specifies upper limit of a \zxhlp{FOR} loop
\zxhlp{TPHEAD}                Verb to list image header from FITS or IBM-CV tape
\zxhlp{TPUT}                     Verb-like puts adverbs from a task in file for TGETs
\zxhlp{TV1SET}                Verb to reset 1D gaussian fitting initial guess on TV plot.
\zxhlp{TV3COLOR}              Verb to initiate 3-color display using 3 TV channels
\zxhlp{TVACOMPS}              Verb to add slice model components directly on TV graphics
\zxhlp{TVAGUESS}              Verb to re-plot slice model guess directly on TV graphics
\zxhlp{TVAMODEL}                Verb to add slice model display directly on TV graphics
\zxhlp{TVANOT}                  Verb to load annotation to the TV image or graphics
\zxhlp{TVARESID}                Verb to add slice model residuals directly on TV graphics
\zxhlp{TVASLICE}                 Verb to add a slice display on TV graphics from slice file
\zxhlp{TVBLINK}                  Verb which blinks 2 TV planes, can do enhancement also
\zxhlp{TVBOX}                     Verb to set boxes with TV cursor & graphics display.
\zxhlp{TVCLEAR}                  Verb to clear image from TV channel(s)
\zxhlp{TVCOMPS}                Verb to display slice model components directly on TV graphics
\zxhlp{TCUBE}                     Verb to load a cube into tv channel(s) & run a movie
\zxhlp{TVFIDDLE}                  Verb enhances B/W or color TV image with zooms
\zxhlp{TVGUESS}                  Verb to display slice model guess directly on TV graphics
\zxhlp{TVHELIX}                  Verb to activate a helical hue-intensity TV pseudo-coloring
\zxhlp{TVHUEINT}                  Verb to make hue/intensity display from 2 TV channels
\zxhlp{TVILINE}                    Verb to draw a straight line on an image on the TV
\zxhlp{TVINIT}                     Verb to return TV display to a virgin state
\zxhlp{TVLABEL}                   Verb to label the (map) image on the TV
\zxhlp{TVLAYOUT}                  Verb to label the holography image on the TV with panel layout
\zxhlp{TVLINE}                     Verb to load a straight line to the TV image or graphics
\zxhlp{TVLOD}                      Verb to load an image into a TV channel
\zxhlp{TVLUT}                      Verb which modifies the transfer function of the image
\zxhlp{TVMBLINK}                  Verb which blinks 2 TV planes either auto or manually
\zxhlp{TVMLUT}                      Verb which modifies the transfer function of the image
\zxhlp{TVMODEL}                     Verb to display slice model directly on TV graphics
\zxhlp{TVMOVIE}                     Verb to load a cube into tv channel(s) & run a movie
\zxhlp{TVNAME}                      Verb to fill image name of that under cursor
\zxhlp{TOFF}                        Verb which turns off TV channel(s).
\zxhlp{TVON}                         Turns on one or all TV image planes
\zxhlp{TPHLAME}                     Verb to activate "flame-like" pseudo-color displays
\zxhlp{TVPOS}                        Read a TV screen position using cursor
\zxhlp{TPSEUDO}                     Verb to activate three types of pseudo-color displays
\zxhlp{TVRESID}                     Verb to display slice model residuals directly on TV graphics
\zxhlp{TVROAM}                       Load up to 16 TV image planes and roam a subset thereof
\zxhlp{TVSCROL}                     Shift position of image on the TV screen
\zxhlp{TVSET}                        Verb to set slice Gaussian fitting initial guesses from TV plot
\zxhlp{TVSLICE}                      Verb to display slice file directly on TV
\zxhlp{TVSPLIT}                     Compare two TV image planes, showing halves
\zxhlp{TVSTAR}                      Verb to plot star positions on top of a TV image

```

```

\zxhlp{TVSTAT}      Find the mean and RMS in a blotch region on the TV
\zxhlp{TVTRANSF}    Interactively alters the TV image plane transfer function
\zxhlp{TVWEDGE}     Show a linear wedge on the TV
\zxhlp{TVWINDOW}    Set a window on the TV with the cursor
\zxhlp{TVWLABEL}    Put a label on the wedge that you just put on the TV
\zxhlp{TVZOOM}      Activate the TV zoom
\zxhlp{TYPE}        Type the value of an expression
\zxhlp{U2CAT}       list a user's UV and scratch files on disk \zxhlp{IN2DISK}
\zxhlp{U3CAT}       list a user's UV and scratch files on disk \zxhlp{IN3DISK}
\zxhlp{U4CAT}       list a user's UV and scratch files on disk \zxhlp{IN4DISK}
\zxhlp{U5CAT}       list a user's UV and scratch files on disk \zxhlp{IN5DISK}
\zxhlp{UCAT}        list a user's UV and scratch files on disk \zxhlp{INDISK}
\zxhlp{UNQUE}       remove a given job from the job queue
\zxhlp{UOCAT}       list a user's UV and scratch files on disk \zxhlp{OUTDISK}
\zxhlp{USAVE}       Pseudoverb to save full POPs environment in named file
\zxhlp{VALUE}       Convert a string to a numeric value
\zxhlp{VERB}        Declares a name to be a symbol of type verb
\zxhlp{VGET}        Verb-like gets adverbs from version task parameter save area
\zxhlp{VGINDEX}    Verb lists those tasks for which \zxhlp{VGET} will work.
\zxhlp{VLA}         puts the list of \zxhlp{VLA} antennas in the current file on stack
VLBA      puts the list of VLBA antennas in the current file on stack
\zxhlp{VPUT}        Verb-like puts adverbs from a task in files for VGETs
\zxhlp{WAITTASK}   halt AIPS until specified task is finished
\zxhlp{WEDERASE}   Load a wedge portion of the TV with zeros
\zxhlp{XHELP}      Accesses hypertext help system
\zxhlp{ZAP}         Delete a catalog entry and its extension files

```

13.41 VLA

```

\zxhlp{APGPS}      Apply GPS-derived ionospheric corrections
\zxhlp{ASDMFILE}   Full path to \zxhlp{EVLA} \zxhlp{ASDM}/BDF directory
\zxhlp{BDF2AIPS}   Read \zxhlp{EVLA} \zxhlp{ASDM}/BDF data into AIPS
\zxhlp{BDFLIST}    List contents of \zxhlp{EVLA} ASD data file
\zxhlp{BPWAY}     Determines channel-dependent relative weights
\zxhlp{BREAK}      procedure to \zxhlp{TELL} \zxhlp{FILLM} to break all current uv files, start new
\zxhlp{CALDIR}    lists calibrator source models available as AIPS FITS files
\zxhlp{CALIN}      specifies name of input disk file usually with calibration data
\zxhlp{CLCOR}      applies user-selected corrections to the calibration CL table
\zxhlp{CONFIG}    Configuration ID number within an \zxhlp{EVLA} \zxhlp{ASDM}/BDF data set
\zxhlp{CVEL}       shifts spectral-line UV data to a given velocity
\zxhlp{DAYFX}     Fixes day number problems left by \zxhlp{FILLM}
\zxhlp{DFCOR}     applies user-selected corrections to the calibration CL table
\zxhlp{DOOSRO}   calibrating amplitude and phase, and imaging \zxhlp{VLA} data
\zxhlp{DOVLAMP}   Produces amp calibration file for phased-\zxhlp{VLA} VLBI data
\zxhlp{FARAD}     add ionospheric Faraday rotation to CL table
\zxhlp{FEW}        procedure to \zxhlp{TELL} \zxhlp{FILLM} to append incoming data to existing uv files
\zxhlp{FILLM}     reads \zxhlp{VLA} on-line/archive format uv data tapes (post Jan 88)
\zxhlp{FIXAL}     least squares fit aliasing function and remove
\zxhlp{FLOPM}     reverses the spectral order of UV data, can fix \zxhlp{VLA} error
\zxhlp{FRMAP}     Task to build a map using fringe rate spectra
\zxhlp{FXALIAS}   least squares fit aliasing function and remove
\zxhlp{GCPLT}     Plots gain curves from text files
\zxhlp{GPSDL}     Calculate ionospheric delay and Faraday rotation corrections
\zxhlp{LDGPS}     load GPS data from an ASCII file
\zxhlp{MANY}       procedure to \zxhlp{TELL} \zxhlp{FILLM} to start new uv files on each scan
\zxhlp{MAPBM}    Map \zxhlp{VLA} beam polarization

```

\zxhlp{MORIF}	Combines IFs or breaks spectral windows into multiple windows (IFs)
\zxhlp{NCHAN}	Number of spectral channels in each spectral window
\zxhlp{NIF}	Number of IFs (spectral windows) in a data set
\zxhlp{PANEL}	Convert \zxhlp{HOLGR} output to panel adjustment table
\zxhlp{PCVEL}	shifts spectral-line UV data to a given velocity: planet version
\zxhlp{PIPEAIPS}	calibrating amplitude and phase, and imaging \zxhlp{VLA} data
\zxhlp{PRTSY}	Task to print statistics from the SY table
\zxhlp{QUIT}	procedure to \zxhlp{TELL} \zxhlp{FILLM} to stop at the end of the current scan
\zxhlp{REIFS}	Breaks spectral windows into multiple spectral windows (IFs)
\zxhlp{REWAY}	computes weights based in rms in spectra
\zxhlp{RFLAG}	Flags data set based on time and freq rms in fringe visibilities
\zxhlp{STOP}	procedure to \zxhlp{TELL} \zxhlp{FILLM} to break all current uv files and stop
\zxhlp{SYSOL}	undoes and re-does nominal sensitivity application for Solar data
\zxhlp{TECOR}	Calculate ionospheric delay and Faraday rotation corrections
\zxhlp{TELFLM}	procedure to \zxhlp{TELL} real-time \zxhlp{FILLM} a new \zxhlp{APARM}(1) value
\zxhlp{TYAPL}	undoes and re-does nominal sensitivity application
\zxhlp{TYSMO}	smooths and filters a calibration TY or SY table
\zxhlp{USERLIST}	Alphabetic and numeric list of \zxhlp{VLA} users, points to real list
\zxhlp{VLABP}	\zxhlp{VLA} antenna beam polarization correction for snapshot images
\zxhlp{VLAMP}	Makes \zxhlp{ANTAB} file for phased \zxhlp{VLA} used in VLBI observations
\zxhlp{VLANT}	applies \zxhlp{VLA}/\zxhlp{EVLA} antenna position corrections from OPs files
\zxhlp{VLAPROCS}	Procedures to simplify the reduction of VLBA data
\zxhlp{VLARUN}	calibrating amplitude and phase, and imaging \zxhlp{VLA} data
\zxhlp{VLATECR}	Calculate ionospheric delay and Faraday rotation corrections

13.42 VLBI

\zxhlp{ACCOR}	Corrects cross amplitudes using auto correlation measurements
\zxhlp{ACFIT}	Determine antenna gains from autocorrelations
\zxhlp{ACSCL}	Corrects cross amplitudes using auto correlation measurements
\zxhlp{AFILE}	sorts and edits MkIII correlator A-file.
\zxhlp{ALIAS}	adverb to alias antenna numbers to one another
\zxhlp{ANBPL}	plots and prints uv data converted to antenna based values
\zxhlp{ANCAL}	Places antenna-based Tsys and gain corrections in CL table
\zxhlp{ANTAB}	Read amplitude calibration information into AIPS
\zxhlp{ANTNAME}	A list of antenna (station) names
\zxhlp{APCAL}	Apply TY and GC tables to generate an SN table
\zxhlp{ASTROMET}	Describes the process of astrometric/geodetic reduction in AIPS
\zxhlp{BANDPOL}	specifies polarizations of individual IFs
\zxhlp{BLING}	find residual rate and delay on individual baselines
\zxhlp{BSCAN}	seeks best scan to use for phase cal, fringe search, ..
\zxhlp{BSPRT}	print BS tables
\zxhlp{CALIN}	specifies name of input disk file usually with calibration data
\zxhlp{CAPLT}	plots closure amplitude and model from CC file
\zxhlp{CL2HF}	Convert CL table to HF table
\zxhlp{CLCOR}	applies user-selected corrections to the calibration CL table
\zxhlp{CLPLT}	plots closure phase and model from CC file
\zxhlp{CLVLB}	Corrects CL table gains for pointing offsets in VLBI data
\zxhlp{CROSSPOL}	Procedure to make complex poln. images and beam.
\zxhlp{CRSFRING}	Procedure to calibrate cross pol. delay and phase offsets
\zxhlp{CVEL}	shifts spectral-line UV data to a given velocity
\zxhlp{CXPOLN}	Procedure to make complex poln. images and beam.
\zxhlp{DFCOR}	applies user-selected corrections to the calibration CL table
\zxhlp{DOVLAMP}	Produces amp calibration file for phased-\zxhlp{VLA} VLBI data
\zxhlp{DTSIM}	Generate fake UV data
\zxhlp{EDITR}	Interactive baseline-oriented visibility editor using the TV
\zxhlp{FRING}	fringe fit data to determine antenna calibration, delay, rate

```

\zxhlp{FRMAP} Task to build a map using fringe rate spectra
\zxhlp{FRPLT} Task to plot fringe rate spectra
\zxhlp{FXAVG} Procedure to enable VLBA delay de-correlation corrections
\zxhlp{FXPOL} Corrects VLBA polarization assignments
\zxhlp{HF2SV} convert HF tables from \zxhlp{FRING}/\zxhlp{MBDLY} to form used by Calc/Solve
\zxhlp{HFPRT} write HF tables from \zxhlp{CL2HF}

HLPEDIPC Interactive PC table editor \zxhlp{PCEDT} - run-time help
HLPIBLED Interactive Baseline based visibility Editor - run-time help
HLPSCIMG Full-featured image plus self-cal loops, editing - run-time help
HLPSCMAP Imaging plus self-cal and editing \zxhlp{SCMAP} - run-time help
\zxhlp{HYB} \zxhlp{RUN} to set parameters for HYBRID (\zxhlp{CALIB}/\zxhlp{MX}) self-cal imaging
\zxhlp{IBLED} Interactive BaseLine based visibility EDitor
\zxhlp{KRING} fringe fit data to determine antenna calibration, delay, rate
\zxhlp{M3STAR} translate Haystack MKIII VLBI format "A" TAR's into AIPS
\zxhlp{MATCH} changes antenna, source, FQ numbers to match a data set
\zxhlp{MBDLY} Fits multiband delays from IF phases, updates SN table
\zxhlp{MERGECAL} Procedure to merge calibration records after concatenation
\zxhlp{MK3IN} translate Haystack MKIII VLBI format "A" tapes into AIPS
\zxhlp{MK3TX} extract text files from a MKIII VLBI archive tape
\zxhlp{OBPLT} Plot columns of an OB table.
\zxhlp{OOFRING} fringe fit data to determine antenna calibration, delay, rate
\zxhlp{PCASS} Finds amplitude bandpass shape from pulse-cal table data
\zxhlp{PCCOR} Corrects phases using \zxhlp{PCAL} tones data from PC table
\zxhlp{PCEDT} Interactive TV task to edit PC tables
\zxhlp{PCFIT} Finds delays and phases using a PC table
\zxhlp{PCFLG} interactive flagging of Pulse-cal data in channel-TB using the TV
\zxhlp{PCLOD} Reads ascii file containing pulse-cal info to PC table.
\zxhlp{PCPLT} Plots pulse-cal tables in 2 dimensions as function of time
\zxhlp{PCVEL} shifts spectral-line UV data to a given velocity: planet version
\zxhlp{PHSLIMIT} gives a phase value in degrees
\zxhlp{POLSN} Make a SN table from cross polarized fringe fit
\zxhlp{RESEQ} Renumber antennas
\zxhlp{RLDLY} fringe fit data to determine antenna R-L delay difference
\zxhlp{SEARCH} Ordered list of antennas for fring searches
\zxhlp{SNEDT} Interactive SN/CL/TY/SY table editor using the TV
\zxhlp{TAUO} Opacities by antenna number
\zxhlp{TECOR} Calculate ionospheric delay and Faraday rotation corrections
\zxhlp{TFILE} sorts and edits MkIII correlator UNIX-based A-file.
\zxhlp{TRECVR} Receiver temperatures by polarization and antenna
\zxhlp{UVPOL} modifies UV data to make complex image and beam
\zxhlp{VBCAL} Scale visibility amplitudes by antenna based constants
\zxhlp{VBGLU} Glues together data from multiple passes thru the VLBA corr.
\zxhlp{VBMRG} Merge VLBI data, eliminate duplicate correlations
\zxhlp{VLAMP} Makes \zxhlp{ANTAB} file for phased \zxhlp{VLA} used in VLBI observations
\zxhlp{VLASUMM} Plots selected contents of SN or CL files
\zxhlp{VLBAAMP} applies a-priori amplitude corrections to VLBA data
\zxhlp{VLBAARCH} Procedure to archive VLBA correlator data
\zxhlp{VLBACALA} applies a-priori amplitude corrections to VLBA data
\zxhlp{VLBACCOR} applies a-priori amplitude corrections to VLBA data
\zxhlp{VLBACPOL} Procedure to calibrate cross-polarization delays
\zxhlp{VLBACRPL} Plots crosscorrelations
\zxhlp{VLBAEOPS} Corrects Earth orientation parameters
\zxhlp{VLBAFIX} Procedure that fixes VLBA data, if necessary
\zxhlp{VLBAFPOL} Checks and corrects polarization labels for VLBA data
\zxhlp{VLBAFQS} Copies different FQIDS to separate files
\zxhlp{VLBAFRGP} Fringe fit phase referenced data and apply calibration
\zxhlp{VLBAFRNG} Fringe fit data and apply calibration
\zxhlp{VLBAIT} Procedure to read and process VLBA data (Phil Diamond)

```

\zxhlp{VLBAKRG}	Fringe fit phase referenced data and apply calibration
\zxhlp{VLBAKRNG}	Fringe fit data and apply calibration
\zxhlp{VLBALOAD}	Loads VLBA data
\zxhlp{VLBAMCAL}	Merges redundant calibration data
\zxhlp{VLBAMPCL}	Calculates and applies manual instrumental phase calibration
\zxhlp{VLBAPANG}	Corrects for parallactic angle
\zxhlp{VLBAPCOR}	Calculates and applies instrumental phase calibration
\zxhlp{VLBAPIPE}	applies amplitude and phase calibration procs to VLBA data
\zxhlp{VLBARUN}	applies amplitude and phase calibration procs to VLBA data
\zxhlp{VLBASNPL}	Plots selected contents of SN or CL files
\zxhlp{VLBASRT}	Sorts VLBA data, if necessary
\zxhlp{VLBASUBS}	looks for subarrays in VLBA data
\zxhlp{VLBASUMM}	Prints a summary of a VLBI experiment
\zxhlp{VLBATECR}	Calculate ionospheric delay and Faraday rotation corrections
\zxhlp{VLBAUTIL}	Procedures to simplify the reduction of VLBA data
\zxhlp{VLBIN}	Task to read VLBI data from an NRAO/MPI MkII correlator
\zxhlp{VLBINPRM}	Control parameters to read data from NRAO/MPI MkII correlators
\zxhlp{VLOG}	Pre-process external VLBA calibration files
\zxhlp{VPLOT}	plots uv data and model from CC file
\zxhlp{WEIGHTIT}	Controls modification of weights before gain/fringe solutions

13.43 Additional recipes

13.43.1 Sautéed sole tobago with bananas, pecans and lime

1. Preheat 1/2 cup **vegetable oil** in a heavy sauce pan over medium-high heat.
2. Dredge 8 filets of **sole** or **flounder** lightly in **flour**.
3. Sauté until golden brown, about 3 minutes each side. Remove to warm platter.
4. Pour off excess oil and wipe down sauce pan. Place pan back on stove over high head; add 1/4 cup **butter**.
5. When foamy and just starting to brown, add 2 cups diagonally sliced **bananas** (1/2" slices) and 1 cup **pecan** halves. Toss and cook for 1 minute.
6. Add 1/2 cup fresh **lime juice** and 1 cup dry white **wine** (or light stock). Cook for another 2 minutes.
7. Add 1/4 cup **fresh herbs** (mint, parsley, coriander, basil or tarragon).
8. Pour sauce and bananas over fish. Garnish with additional banana slices and lime wedges.

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

13.43.2 Mexican bananas

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

13.43.3 Banana sweet potato puff casserole

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

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

13.43.4 Cream of banana soup

1. Cook 1 quart green **banana** pulp, 1 1/2 quarts **chicken stock**, 1 small **celery stalk**, 1/2 **onion**, 1 **carrot**, 1 small **bay leaf**, 5 **peppercorns**, and **salt** to taste together for about 30 minutes until the mixture thickens.
2. Strain over 1/4 cup **flour** and 1/4 cup **butter** which have been combined as for a white sauce. Cook until thickened.
3. Just before serving, add 2 cups **cream** or **milk** and heat.
4. Serve with a slice of lemon on each plate as a garnish.

13.43.5 Hawaiian banana cream pie

1. Preheat oven to 375° F.
2. In a bowl, combine 1 cup chopped cashew or macadamia **nuts**, 1/2 cup flaked **coconut**, and 2 tablespoons **brown sugar**.
3. Beat 1 **egg white** until stiff; fold into nut mixture.
4. Press mixture evenly into an 8-inch pie plate, building up the sides slightly. Bake for 7 minutes or until crust is lightly browned. Crust will tighten as it cools (use a rack).
5. In a medium-sized saucepan, beat 3 **egg yolks**. Mix in 5 tablespoons **cornstarch** and 3/4 cup granulated **sugar**. Stir in 1.5 cups **milk**, 1/4 teaspoon **salt**, and 1 tablespoon unsalted **butter**.
6. Cook mixture slowly over medium heat, stirring constantly, for 5 to 7 minutes. Filling should be bubbling and thick.
7. Remove from heat and stir in 1 teaspoon **vanilla extract**. Transfer this custard to a glass bowl, cover with plastic wrap, and refrigerate for 2 hours.
8. Two hours before serving, whip 1/2 cup heavy **whipping cream** to stiff peaks and fold into custard. Peel and slice one **banana**, arranging evenly on bottom of crust. Spoon custard filling into crust. Cover again with plastic wrap and chill for 2 more hours.
9. Sprinkle 1/2 cup finely chopped cashew or macadamia **nuts** evenly over the filling. Peel, slice and arrange a second **banana** in a circular fashion around the outside top of the pie, placing a few slices decoratively in the center.

A Easy Continuum UV-Data Calibration and Imaging

This appendix contains a step-by-step guide to calibrating simple continuum data from interferometers using *AIPS*. This older guide is now preceded with information about a new “pipeline” of *AIPS* procedures which can *automagically* calibrate, image, and even self-calibrate many multi-source continuum (and line) data sets. Although this pipeline is unlikely to produce immediately publishable results in most cases, it does produce a data set with preliminary calibration and editing which can then be edited further “by hand.” Following the additional editing, the pipeline may be re-run to produce improved images. When good single-source data sets are produced, self-calibration can also be performed using the methods described in § 5.4.

A.1 VLARUN

The user should read *AIPS* Memo 112, “Capabilities of the **VLA** pipeline in AIPS,” by Lorant O. Sjouwerman dated March 19, 2007. This memo goes into details and advice that are beyond the scope of this appendix. The memo may be obtained from the *AIPS* web set <http://www.aips.nrao.edu/>.

To use the pipeline, you must first load the **VLA** Archive or other multi-source visibility data into *AIPS*; see § 4.1.1. To acquire the pipeline, including all procedures and special adverbs, enter

> **RUN VLARUN C_R** to define the procedures and adverbs

Do this only once. The procedures will be remembered in your LASTEXIT **SAVE/GET** file.

The pipeline has three stages: (1) calibration and editing, (2) basic imaging, and (3) self-calibration. Study the recommendations and other advice in **EXPLAIN VLARUN C_R**. Note that **VLARUN** can also handle spectral-line data. Then, the inputs for the calibration stage include

- | | |
|--|--|
| > TASK 'VLARUN' ; INP C_R | to review the inputs needed. |
| > WORKDISK <i>n</i> C_R | to specify the disk containing the <i>uv</i> data. |
| > CATNUM <i>catn</i> C_R | to specify the catalog number of the <i>uv</i> data; INNAME et al. may also be used. |
| > FASTSW 1 C_R | to correct source name peculiarities induced by fast-switching observations. |
| > AUTOFLAG 2 C_R | to use FLAGR on all data and QUACK on all but high frequency data sets |
| > PHAINT 1 ; AMPINT 5 C_R | to set the phase calibration interval to 1 minute and the amplitude calibration interval to 5 minutes. |
| > DOMODEL 1 C_R | to use standard flux calibrator models where available. |
| > NOPAUSE -1 C_R | to have the pipeline pause after GETJY to allow you to evaluate whether it is okay to proceed. |
| > AUTOPLOT -1 C_R | to make no diagnostic plot files. |
| > DOIMAGES -1 C_R | to do calibration and editing only. |
| > INP ; TPUT C_R | to double check the inputs and save them. |
| > VLARUN C_R | to calibrate and edit your data. |

The pipeline can do imaging and even self-calibration, although the latter is not for the faint at heart. To

include this in your pipeline run, enter

- > DOIMAGES 1 C_R to request imaging.
- > ARRYSIZE 0 C_R to let the procedure find the array dimensions.
- > IMSIZE -1 C_R to image the target source over the full primary beam, with smaller areas for calibrators.
- > NITER *nit*; CUTOFF *f* C_R to set the number of iterations *nit* and Clean flux limit *f*. NITER must be > 0 for the image to be Cleaned. Use conservative values to begin with and higher limits in subsequent runs.
- > ALLIMG 1 C_R to make images of calibrators as well as target sources.
- > SLFCAL 0 C_R to avoid self-calibration initially.
- > INP ; TPUT C_R to double check the inputs and save them.
- > VLARUN C_R to calibrate, edit, and image your data.

In the first run of VLARUN, it is recommended to avoid self-calibration although the making of images will help you evaluate the quality of the auto-editing. If your data are not “perfect” (*i.e.*, good enough), then consider one or more of the interactive editing tasks such as TVFLG (§ 4.4.3), SPFLG (§ 10.2.2), EDITA (§ 4.4.2), and EDITR (§ 5.5.2) along with the advice given on editing in § 4.3.1, § 4.4, § 4.4.1, and § 5.5. Then

- > TGET VLARUN C_R to restore the inputs.
- > AUTOFLAG -1 C_R to do no more automatic flagging.
- Consider raising NITER and/or lowering CUTOFF in this second pass, plus
- > SLFCAL *sni* C_R to also do self-calibration with interactivity on the TV (*sni* > 0) or in a batch-like process (*sni* < 0) for $|sni|$ self-cal iterations.
- > INP ; TPUT C_R to double check the inputs and save them.
- > VLARUN C_R to calibrate, edit, and image your data.

Repeat as needed. Note that VLARUN has been used on many of the data sets in the VLA archive, with the results archived at <http://archive.nrao.edu/>. These results include both images and calibrated *uv* data sets.

A.2 Basic calibration

From VLA Archive Tape to a UV FITS Tape
AIPS Memo No. 76 Updated
Glen Langston

The Gentle User enters the Computer room with a VLA archive tape containing a scientific breakthrough. The user’s sources are named SOURCE1 and SOURCE2. The interferometer phase is calibrated by observations of CAL1 and CAL2. The flux density scale is calibrated by observing 3C48 (=0137+331) and polarization is calibrated with observations of 3C286 and/or 3C138. Mount the tape on drive number *n*, log in and start AIPS. Example input: AIPS NEW. Mount the tape: INTAPE=*n*; DENS=6250; MOUNT.

PRTTP Find out what is on the tape, get project number and bands. TASK='PRTTP'; PRTLEV=-2; NFILES=0; INP; GO; WAIT; REWIND.

FILLM Load your data from tape. Select only one band at a time to process. TASK='FILLM'; VLAOBS='?'; BAND='?'; NFILES=?; DOWEIGHT 1; INP; GO (Replace all ?’s with appropriate values.) FILLM will load your visibilities (*uv*-data) with weights suitable for calibration into a large file for each band. It also creates 6 AIPS tables each; these tables have two letter names which are:

HI Human readable history of things done to your data. Use PRTHI to read it.

- AN Antenna location and polarization tables. Antenna polarization calibration is placed here.
- NX Index into visibility file based source name and observation time. Not modified by calibration.
- SU Source table contains the list of sources observed and indexes into the frequency table. The flux densities of the calibration sources are entered into this table.
- FQ Frequencies of observation and bandwidth with index into visibility data. Not modified.
- CL Calibration table describing the antenna based gains. Version 1 should never be modified. The CL table contains entries at regular time intervals (*i.e.*, 2 minutes) for each antenna. **The ultimate goal of calibration is to create a good CL version 2.** Use [PRTAB](#) to read tables.

PRTAN Print out the antenna locations. **TASK='PRTAN'; PRTLEV=0; INP; GO.** Choose a good *Reference* antenna (called *R*) near the center of the array ([REFANT=R](#)). Check the [VLA](#) operator log to make sure the antenna was OK during the entire observation.

QUACK Flag the bad points at the beginning of each scan, even the ones with good amplitudes could have bad phases. Creates a Flag Table (FG). You want to use FG table version 1 for all tasks. **TASK='QUACK'; FLAGV=1; OPCOD=' ';** [APARM=0; SOUR=' ';](#) **INP; GO** deletes the first six seconds of each scan, which may not be enough.

- FG A flag table marks bad data. FG tables contain an index into the UV data based on time range, antenna number, frequency and IF number.

LISTR Lists your UV data in a variety of ways. Make a list of your observations. **TASK='LISTR'; OPTYP='SCAN'; DOCRT=-1; SOUR=' ';** [CALC='*' ; TIMER=0; INP; GO.](#) NOTE: If you have observed in a way as to create more than one FREQID, you must run through the entire calibration once for EACH FREQID. For new users, it is better to use [UVCOP](#) to copy each FREQID into separate files and calibrate each file separately. This is required if you are doing polarization calibration.

UVCOP Skip this step if your data consists of only one FREQID. Copy different FREQIDs into separate files. **TASK='UVCOP'; FREQID=?; CLRON; OUTDI=INDI; INP; GO.** The result will be a [???.UVCOP](#) file.

SETJY Sets the flux of your flux calibration source in the SU table. **TASK='SETJY'; SOUR='3C48',' ';** [OPTYP='CALC'; FREQID=1; INP; GO.](#) Adjust flux density for partial resolution following the rules in the [VLA](#) Calibration Source Manual or the *AIPS CookBook*.

TASAV As insurance, make a copy of all your tables. **TASK='TASAV'; CLRON; OUTDI=INDI; INP; GO.**

CALIB **CALIB** is the heart of the *AIPS* calibration package. **RUN VLAPROCS**, an *AIPS* *runfile*, to create procedures [VLACALIB](#), [VLACLCAL](#) and [VLARESET](#). The procedure [VLACALIB](#) runs **CALIB**. Set [UVRANGE](#) and [ANTENNA](#) to zero to allow use of models for 3C48. For L, C and X band 5% and 5 degree errors are OK; for other bands the limits are higher. **CALIB** places antenna amplitude and phase corrections into an SN table for the time of observation of phase calibration sources.

- SN Solution table contains antenna based amplitude and phase corrections for the time of observations of the calibration sources. These SN table results are latter interpolated for all times of observation and placed in a CL table. Only the CL table corrections will be applied to the program sources.

TASK='VLACAL'; CALS='3C48',' '; [CALCODE='*' ; REFANT=R; UVRA=0; SNVER=1; DOCALIB=-1; DOPRINT=1; MINAMP=10; MINPH=10; INP; VLACAL.](#) **VLACALIB** will load and use a source model for 3C48, making [UVRANGE](#) unnecessary. The task **CALIB** lists antenna pairs which deviate significantly from the solution. If you have lots of errors, then carefully examine your data using [TVFLG](#) or [LISTR](#). (See *AIPS CookBook* for a lengthy discussion on flagging.) If one antenna is bad over a limited time range, use [UVFLG](#) to flag that antenna for the time from just after the previous good CAL observation to before the next good CAL observation.

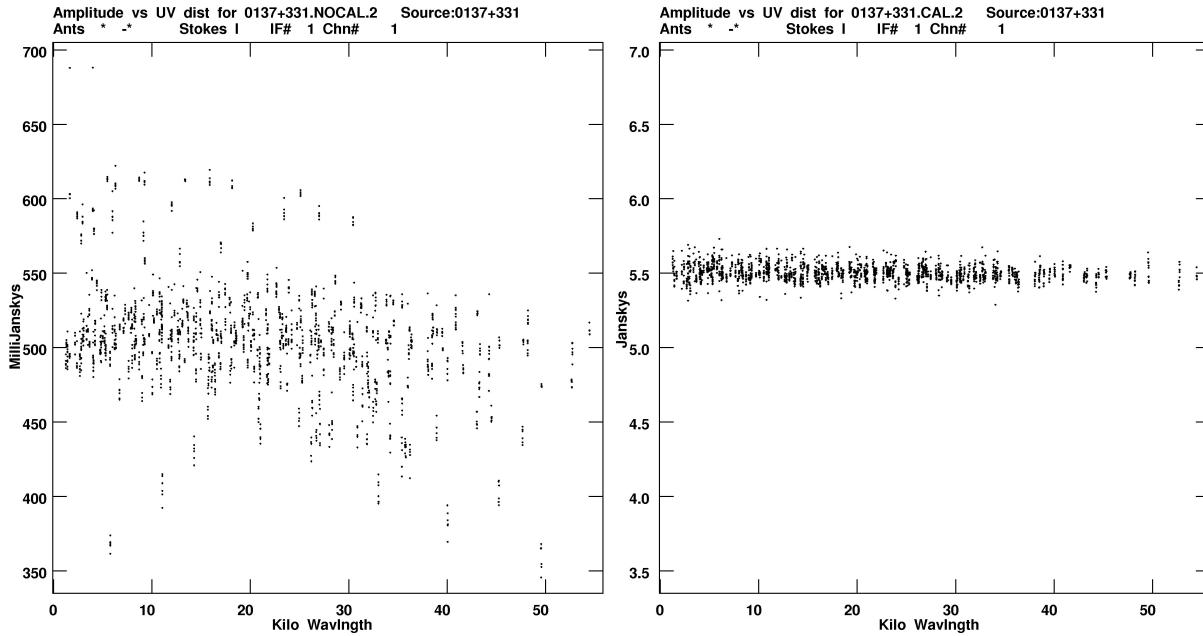


Figure A.1: (left) Un-calibrated *uv*-data and (right) calibrated *uv*-data from a C-band snapshot of 3C48. Default VLA gains are a tenth of the actual gains and can show significant scatter. Only wild *uv* points ~50% greater than the average can be detected before calibration.

UVFLG Flag bad UV-data. **TASK='UVFLG';**
ANTEN=?;0; BASELI=?;0; TIMER=?; OUTFGVER=1; SOUR=' '; **OPCOD=' ';** **INP; GO.** If in doubt about any data, FLAG THEM! If you have flagged the primary calibrator, return to **CALIB** above and try again.

CALIB Now calibrate the antenna gain based on the rest of the cal sources. Look in the Calibrator manual for UV limits; if there are limits, **VLACAL** must be run separately for these sources. **TGET VLACAL; CALS='CAL1','CAL2',' ';** **ANTEN=0; BASELI=0; UVRANGE=?;?; INP; VLACAL.** Flag bad antennas listed. Each execution of **CALIB** replaces previous corrections in the SN table or appends new corrections. If unsatisfied with a **VLACAL** execution, all effects of it are removed by running **VLACAL** again for the same sources (but different ADVERBS or after flagging bad data).

GETJY Sets the flux of phase calibration sources in the SU table. **TASK 'GETJY';** **SOUR='CAL1,'CAL2',' ';** **CALS='3C48',' ';** **BIF=0; EIF=0; INP; GO.** **GETJY** over-writes existing SU table entries, and is not affected by previous executions.

TASAV Good time to save your tables. **TGET TASAV; INP; GO.**

CLCAL Read the antenna amplitude and phase corrections from the SN table and interpolate the corrections into a new CL table. **CLCAL** applies calibration source corrections to the program sources. Each execution of **CLCAL** adds to output CL table version 2. **CLCAL** is run using the procedure **VLACLCAL.** **TASK='VLACLC'; SOUR='SOURCE1','CAL1',' ';** **CALS='CAL1',' ';** **OPCODE='CAL1'; TIMER=0; INTERP='2PT'; INP; VLACLC.** Run **CLCAL** for the second source using the second calibrator. **TGET VLACLC; SOUR='SOURCE2','CAL2',' ';** **CALS='CAL2',' ';** **INP; VLACLC.** Move the SN table corrections for 3C48 into the CL table. **TGET VLACLC; SOUR='3C48',' ';** **CALS='3C48',' ';** **INP; VLACLC.** (3C48 could also be calibrated with CAL1 or CAL2.)

LISTR Make a matrix listing of the Amplitude and RMS of calibration sources with calibration applied. Look for wild points. `TASK='LISTR'; OPTYP='MATX'; SOUR='CAL1','CAL2',''; DOCAL=2; DOCRT=-1; DPARM=3,1,0; UVRA=0; ANTEN=0; BASELI=0; BIF=1; INP; GO`. If only a few points are bad, flag them and continue. If too many are bad, delete CL table 2 and the SN tables using `VLARESET`. Then return to the first `CALIB` step. If the data look good, run `LISTR` again for IF two. `TGET LISTR; BIF=2; INP; GO`

UVPLT Plot the *uv*-data in a variety of ways. Make a Flux versus Time plot first. Choose `XINC` so the plot will have no more than 1000 points. `TASK='UVPLT'; SOUR='SOURCE1',''; XINC=10; BPARM(1)=11; DOCAL=2; BIF=1; INP; GO`. Look at the plot with `LWPLA`, `TKPL`, `TVPL` or `TXPL`. Plot other IF . Flag wild points. Plot Flux versus baseline. `TGET UVPLT; BPARM=0; INP; GO`.

Calibration is now complete for continuum, un-polarized observations. Write the calibrated data to tape with `FITTP` if you don't want to calibrate the polarization. To create images from the *uv*-data use `SPLIT` to calibrate the multi-source data and create a single source *uv*-data set. (`FITTP` and `SPLIT` are described at the end of the polarization calibration process.)

A.3 Polarization calibration

For polarization observations, the following steps are required. For 21cm or longer wavelength observations, ionospheric Faraday rotation corrections may be needed. See `FARAD` in the *AIPS CookBook*, but don't expect much help anymore.

TASAV As added insurance, save your tables again. `TGET TASAV; INP; GO`.

LISTR Print the parallactic angles of the calibration sources. `TGET LISTR; SOUR=''; CALC='*'; OPTYP='GAIN'; DPARM=9,0; INP; GO`

PCAL Intrinsic antenna polarization calculation. `PCAL` will be successful only if cal. sources are observed at several parallactic angles. `PCAL` will modify the AN and SU tables. `TASK='PCAL'; CALS='CAL1','CAL2',''; BIF=1; EIF=2; DOCAL=2; REFANT=R; INP; GO`

RLDIF Now determine the absolute linear polarization angle. Make a matrix listing of the angle of 3C286. `TASK 'RLDIF'; SOUR='3C286',''; DOCAL=2; BIF=1; EIF=0; DOPOL=1; GAINUSE=2; DOCRT=-1; INP; GO`. The observed angles are different for each frequency and IF . This task returns the average angles for all IFs in `CLCORPRM`.

CLCOR Now apply the angle corrections to CL table 2. The relative phase of Left and Right circular polarization produces the linear polarization angle and the phase correction is applied to L. The phase difference (twice the angle of linear polarization) for 3C286 is 66° and for 3C138, $\phi = -18^\circ$ at L band, perhaps -24° at higher frequencies. Change `RLDIF`'s results to this form with `FOR I = 1:20; CLCOR(I)=66-CLCOR(I); END`. Then `TASK='CLCOR'; STOKES='L'; SOUR=''; OPCOD='POLR'; BIF=1; EIF=2; GAINVER=2; GAINUSE=0; INP; GO`. Run `RLDIF` again to check the phases. `TGET RLDIF; INP; GO`. Note that `CLCOR` copies the CL table version 2 to 3 while applying the phase correction. If the phases are wrong, delete version 3, return to `PCAL` and `RLDIF` and then do another `CLCOR`.

A.4 Backup and imaging

FITTP Writes the output *uv*-data to tape. `DISMOUNT` your archive; `MOUNT` your output tape. `TASK='FITTP'; DOEOT=1; OUTTAP=INTAP; INP; GO`. Use `DOEOT=-1` when at the beginning of a new tape.

SPLIT The *AIPS* calibration process only modifies the tables associated with the multi-source *uv*-data set. **SPLIT** selects individual sources, reads the CL table and multiplies the visibilities by the corrections to produce a calibrated single-source *uv*-data set. **TASK='SPLIT'; SOUR=''; CALC=''; UVRA=0; TIMER=0; DOCAL=2; FLAGVER=1; GAINUSE=3; DOPOL=1; DOBAN=-1; BIF=0; EIF=0; STOKES=''; BLVER=-1; APARM=0; DOUVCOM=1; ICHANSEL=0; INP; GO**

Mapping Use your favorite Fourier Transform task (*e.g.*, **IMAGR**) to produce images from the calibrated data. Procedure **MAPPR** provides a simplified interface to **IMAGR**.

A.5 Additional recipes

A.5.1 Banana colada

1. Peel and slice 1 ripe **banana**.
2. Place sliced banana in blender along with 6 ounces **pineapple juice** (or crushed tinned pineapple in its own juice) and 1 ounce **rum** plus 1 ounce **coconut rum** or 2 ounce **rum** plus 1 teaspoon **Coco Lopez**.
3. Optionally add 1 ounce **banana liqueur**.
4. Blend until smooth.
5. Add crushed ice, if so desired.
6. If the mixture is too thick, add more juice (or more rum if you prefer!); if too thin, add more banana. This is a really easy recipe to adjust to one's taste.

A.5.2 Breaded chicken and bananas

1. In food processor, blend 1 can **condensed milk**, 1/3 cup **milk**, 1/2 cup flaked **coconut**, and 1/4 cup **lemon juice** until smooth. Pour into a bowl.
2. Prepare 3 cups **corn flake crumbs** in another bowl or plate.
3. Cut 6 very firm **bananas** lengthwise, dip in milk mixture, roll in corn flakes, and set aside.
4. Cut 2 **chickens** into pieces, dip in milk mixture, roll in corn flakes, and place in greased baking pans (2 13x9 pans may be required).
5. Sprinkle chicken with 1/2 cup melted **butter** and bake as 350° F for one hour.
6. Arrange bananas over the chicken. Sprinkle with 1/4 cup melted **butter**. Bake 15 minutes longer or until chicken juices run clear.
7. Garnish with sliced star and/or kiwi fruits if desired.

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

A.5.3 Banana-pineapple bread

1. Mix together 1 cup chopped **nuts**, 2-1/2 cups **sugar**, 5 cups **flour**, 1 teaspoon **salt**, 1 teaspoon **baking powder**, and 1 teaspoon **cinnamon**.
2. Mix together 1-1/2 cups **vegetable oil**, 3 **eggs**, 3 mashed **bananas**, 1 teaspoon **lemon juice**, and 1 can **crushed pineapple** (drained).
3. Combine. Bake at 350° F for one hour.

Thanks to Tim D. Culey, Baton Rouge, La. (tsculey@bigfoot.com).

B A Step-by-Step Guide to Spectral-Line Data Analysis in *AIPS*

Initially contributed by Andrea L. Cox and Daniel Puche

In this guide, we assume that the reader is familiar with the basic tools of *AIPS*; e.g., [MCAT](#) (§ 3.3), [GETN](#) (§ 3.3.1), [IMHEAD](#) (§ 3.3.4) and other *AIPS* tools involving the manipulation of the data catalog are not mentioned. This guide contains three main sections covering editing and calibration of spectral-line data, making and Cleaning of map cubes, and moment analysis and rotation curves of galaxies. *It is assumed through these sections that all sources in the data set were observed at the same frequency; the final section of this guide describes what you should do before beginning data reduction if this is not the case.*

This is an outline of a typical reduction procedure for spectral-line data from the [VLA](#); different users may use slightly different approaches. This guide is a supplement to the *AIPS EXPLAIN* and [ABOUT](#) verbs (§ 3.8) and the *AIPS CookBook*. Some of the less obvious or more important parameters for each task will be mentioned, but the user should *always* check to ensure that the rest of the parameters are specified correctly. When in doubt, the defaults are usually fairly safe. Words in boldface or typewriter fonts represent *AIPS* tasks and their inputs. When you see a phrase enclosed in brackets, replace the phrase and the brackets with the correct input. For example, to specify the source 0134+329

SOURCE '*<source_name>*' would be typed as

SOURCE '0134+329' The text below is in a three-column format, showing a step number on the left, a descriptive paragraph in the center, the name of the *AIPS* task or verb on the right.

B.1 Editing and calibrating spectral-line data

B.1.1 Loading the data

- (1) Go to the [VLA](#) archive web page at

<http://archive.cv.nrao.edu/>

and locate the desired data set using the basic retrieval tool. Use the observing date, program code, or observer's name to locate the data. Give the retrieval code provided to the PI to unlock data within the proprietary period (usually 6 months). Select the data file(s) desired and ask for them with *AIPS*-friendly names. Wait for the e-mail notification that the files are ready and then follow the supplied advice to copy the data files to your computer.

- (2) Before starting *AIPS*, create an environment variable pointing at the disk directory containing the data files. Use `cd < data_area >` followed by `setenv MYDATA 'pwd'` if you use C-shell or its variants or `export MYDATA='pwd'` if you use the bash shell. (Note the back tick marks surrounding the `pwd`.) Then start *AIPS*.

- (3) Load the data from the archive file(s) to AIPS data areas. Use the defaults to load the data without further averaging. The m disk files will be named $<program_name>_i$ for $i = n + 1, \dots, n + m$.

```
VLAOBS '<program_name>' ; DOWEIGHT 1
NFILES n ; NCOUNT m
DATAIN 'MYDATA:<program_name>_'
```

The data loaded to disk will normally be in two parts: One will have the class “CH0” and the other will have the class “LINE.” The names of the files are thus

```
DATE.CH0
DATE.LINE
```

where “DATE” is the date the observations began. Each file is a multi-source file, containing observations for all your sources: flux calibrators, phase calibrators, and target sources. Your spectral-line data are contained in the LINE file, while the CH0 file is a “pseudo-continuum” file; it is the average of the inner 75% of the bandpass and will be used for gain and phase calibration.

- (4) List the “scan summaries” from the CH0 data. *Keep the output for future reference.* Note that the frequency in the header of a multi-source file is always the sky frequency in the center of the band of the first scan of the observation (see § B.4).

```
OPTYPE 'SCAN' ; DOCRT -1 ; OUTPRINT ''
```

to print on the line printer. To make a text file instead

```
OUTPRINT 'MYDATA:<filename_in_all_caps>'
```

- (5) Print the antenna configuration file. *Keep the output for future reference.*

PRTAN

B.1.2 Inspecting and editing the data

There are a number of different ways to isolate and edit bad uv points from your data set. The method described below is typical. Other tasks of interest can be found by typing **ABOUT EDITING** and **ABOUT UV** and by consulting § 4.3.1, § 4.4.3, § 4.4.2, § 5.5.2, and § 10.2.2 in the *AIPS CookBook*.

- (6) Plot amplitude versus baseline length for your flux and phase calibrators. Inspect each source, Stokes, and IF separately. Set **XINC** so there are only a few thousand visibilities on the output plot (the total number of visibilities is listed on the scan summary sheets from Step 4). Use the TV to save trees.

```
BPARM 0 ; DOTV 1
```

If there are anomalous amplitude points, continue to the next step. If your data points have a small scatter, you may not need to edit and can skip to calibration (Step 10).

- (7) Determine if the anomalous data points are from a particular baseline, antenna, Stokes, or IF, inspecting each Stokes and IF separately. The output of this task will be all points that have anomalous amplitudes, based on your selection criteria.

```
OPCODE 'CLIP' ; APARM(1) <max_flux>
APARM(3) <min_flux>
```

- (8) Once you have determined which data points to flag with **UVFND**, flag them with **UVFLG**. You can flag by time-range, baseline, or antenna and you can flag any or all of the Stokes parameters or IFs. To assist in undoing flags, set a **REASON**.

```
OPCODE 'FLAG' ; OUTFGVER 1
```

- (9) Examine the *uv* data for your calibrators on the TV to check for any obvious problems which you might have missed; see § 4.4.3. Check each IF and each Stokes separately and edit the data more carefully, if necessary, before continuing.

`DOCAL -1 ; CALCODE '*' ; FLAGVER 1`

TVFLG

B.1.3 Calibrating the data

Steps 10–15 should be applied to the CH0 data alone, not to the LINE data. To ensure that all inputs are set to their defaults before continuing, type

`RESTORE 0`

Then, when you are satisfied with your editing, type

`RUN VLAPROCS`

to set up VLA-specific parameters and procedures for calibration. You may turn all adverbs for a specific *< task >* to their default values with

`DEFAULT < task >`

If you have multi-frequency data, each frequency must be calibrated separately; this can be done by specifying the `FREQID` parameter in each task (see § B.4). More information on calibrating your data can be found by typing `ABOUT CALIBRAT` and `HELP CALIBRAT` and consulting Chapter 4.

- (10) Calculate the flux of the primary flux calibrator for the channel zero (CH0) data.

SETJY

`SOURCE '< flux_calibrator >' , , ; OPTYPE 'CALC'`

- (11) Calculate gain and phase solutions for *all* of the calibrators. In this case, you must run this procedure once for each source. Check for the presence of an appropriate model with `CALDIR`. For those sources having a model

`DOCALIB 1 ; UVRANGE 0 ; ANTENNAS 0
CALSOEUR '< flux_calibrator_1 >' , ,`

VLACALIB

For sources without a model, choose a `UVRANGE` according to the tables in the *VLA Calibration Manual*. You may do more than one source at a time, if they have the same `UVRANGE`. The output of this procedure is a solution (SN) extension table, which is printed automatically. Select a reference antenna (`REFANT`) which did not have any problems during the observing run and which is located near the center of the array.

`DOCALIB 1 ; UVRANGE < uv_min > , < uv_max >
CALSOEUR '< calibrator_1 >' , ... , '< calibrator_n >'
DOPRINT 1; OUTPRINT '< filename.in_all_caps >'`

The output from `VLACALIB` will include a list of closure errors. If there are too many large errors, edit your data carefully using `UVFND`, `TVFLG`, or `LISTR` as described above. *Destroy old SN tables with EXTDEST and then re-run VLACALIB until the solutions are satisfactory.* The output will include amplitudes and phases for each baseline; for each calibrator, the amplitudes should be approximately constant and the phases should vary smoothly over time.

- (12) Calculate the flux densities of the secondary (phase) calibrators from the primary (flux) calibrator, based upon the flux densities in the source (SU) table and the antenna gain solutions in the solution (SN) table. *Destroy bad or redundant versions of the SN tables before using this task.* Compare the computed fluxes with those listed in the *VLA Calibration Manual*.

GETJY

`SOURCES '< phase_cal_1 >' , ... , '< phase_cal_n >'
CALSOEUR '< flux_cal >' , ,`

- (13) This procedure interpolates the solutions derived from the calibrators into the calibration (CL) table for *all* sources. Run this procedure once for each phase calibrator (which may be used to calibrate multiple sources).

```
SOURCES '<phase_cal>' , '<source_1>' , ... , '<source_n>'  
CALSOR ' <phase_cal>' , ,  
OPCODE 'CALI' ; INTERPOL '2PT'  
OUTPRINT '<filename_in_all_caps>'
```

Note: if you are observing at low frequencies or there are gaps in your observations of phase calibrators, you may want to use **SAMPTYPE 'BOX'** and review the other inputs carefully. They have changed.

- (14) Apply the calibration to the phase calibrators and examine the amplitudes, which should be nearly constant, and the phases, which should be nearly zero.

```
SOURCES ' ' ; CALCODE '*'           ← print results for all calibrators  
OPTYPE 'MATX' ; DOCALIB 2 ; GAINUSE 2; DPARM 5, 1, 0  
UVRANGE 0 ; DOCRT -1 ; OUTPRINT '<filename_in_all_caps>'
```

- (15) Examine the *uv* data for your *sources* on the TV to check for any obvious problems which you may have missed. Re-edit the data (Steps 6–9) if necessary.

```
DOCAL 2 ; GAINUSE 2 ; CALCOD '-CAL'
```

If you have too many visibilities to fit on the TV screen, you may want to set **TIMERANGE**, **SOURCES** or **DPARM(6)** (the input averaging time) to limit the amount of data displayed. There are also interactive options to set the on-screen averaging time and the time range currently displayed.

- (16) To calibrate the spectral-line data, simply copy the calibration (CL) table from the CH0 to the LINE data.

```
INEXT 'CL' ; NCOUNT 1 ; INVERS 2
```

Also be sure to copy the flagging (FG) table

```
INEXT 'FG' ; INVERS 1
```

Steps 17 and 18 should be applied to the LINE data alone, not to the CH0 data

- (17) Calibrate the bandpass for the LINE data using the primary (flux) calibrator. The output from this task is a table (BP) of the bandpass spectrum.

```
GET3NAME <CH0data>  
CALSOR '<flux_cal>' , , ; CALCODE '*'  
DOCALIB -1 ; FLAGVER 1
```

- (18) Examine the bandpass for each of the antennas on the TV.

```
APARM(8) 2 ; DOTV 1 ; STOKES 'RR'  
ANTENNAS 0 ; NCOUNT 4           ← plot 4 antennas at a time
```

Then do the LL Stokes. After this, generate a plot (PL) file of the total bandpass for each Stokes.

```
NCOUNT 0 ; DOTV -1
```

Plot the bandpass on the laser printer; specify **PLVER** for each Stokes.

- (19) Now that the calibration is completed, write the calibrated CH0 and LINE data to tape.

Print the contents of the tape(s) for your data.

VLACLCAL

LISTR

TVFLG

TACOP

BPASS

POSSM

LWPLA

FITTP

PRTTP

- (20) Apply the calibration and editing tables, writing single-source *uv* files for imaging. **SPLIT**
- ```
CALCODE '-CAL' ←write uv for all non-calibrator sources
DOCALIB 2; GAINVER 2 ; DOBAND 1; BPVER 1
```

## B.2 Making and Cleaning image cubes

- (21) Determine the imaging parameters needed for a full-field image of the continuum. Set the **BPARMs** to cover the full single-dish beam area (or more at 21-cm wavelength). **SETFC**
- ```
IMSIZE 0 ; CELLSIZE 0
BOXFILE 'MYDATA:<source_name>.BOX'
```
- Delete the output **BOXFILE** and re-run the task with your chosen (more congenial or exact) **CELLSIZE**.
- (22) Make a **CHO** image with a large field (set by **IMSIZE** and **CELLSIZE** returned by **SETFC**) to look for strong continuum sources. Use uniform weighting (**UVWTFN** '') with **ROBUST** of 0 or -1 if resolution is more important than detecting significantly extended sources; set **ROBUST** to 2 or more if the converse is true. **IMAGR**
- ```
STOKES 'I' ; NITER 0
```
- Then make a dirty image cube of your **LINE** data in the center field only (unless more are needed for the line source).
- ```
NFIELD 1 ; BCHAN 1 ; ECHAN 0
```
- (23) Examine the cube on the TV to determine which channels are free of line emission. **TMMOVIE**
- (24) Calculate the noise in a few of the line-free channels.
- Display a channel on the TV **TVALL**
- ```
TBLC = 0 , 0 , <channel_number> ; TTRC 0
```
- Select a large window that contains no continuum sources. **TVWIN**
- Calculate the RMS noise inside the window. The verb **TVSTAT** is helpful if there are no large rectangular windows free of continuum emission. **IMSTAT**
- Having an initial guess of the signal-free RMS, you may attempt to refine that estimate (see message generated) and also produce a histogram plot. **IMEAN**
- ```
BLC 0 ; TRC 0; DOTV 1; DOHIST 1; NBOXES 256
```
- (25) Remove the continuum emission by fitting a baseline in the *uv* plane to the line-free channels; see **EXPLAIN UVLSF** and § 8.3. If you don't want this task for flagging, but only for continuum subtraction, the important parameters are **UVLSF**
- ```
ORDER 1 ; DOOUT 1 ; FLUX 0 ; CUTOFF 0
```
- Choose your line-free channels, normally avoiding the channels at the edges of your bandpass since they are usually quite noisy.
- ```
ICHANSEL <begin1>, <end1>, , 0, 0, <begin2>, <end2>
```
- This task will also write a new continuum data set which is a better estimate of the continuum than the **CHO** file which contains contributions from the line signals. If there is a bright continuum source far away from the phase center, you *will* want to use **SHIFT** to center it while running **UVLSF**. Multiple interfering sources might even require **UVSUB** with a good continuum model rather than **UVLSF**.

(26) Make a dirty cube containing only the line emission. Use the same parameters as in Step 22, including usually only one field — but consider multi-scale Clean. **IMAGR**

(27) Calculate the noise in the line-free channels as in Step 24.

(28) Make a contour plot of the beam.

Display a channel on the TV.

TBLC = 0, 0, <channel_number> ; **TTRC** 0

Select a window containing the source (or beam) you want to plot.

Display the contour plot on the TV. For a beam plot, good parameters are

PLEV 10 ; **LEVS** -3,-1,1,3,5,7,9 **DOTV** 1

When you are happy with the plot, generate a plot (PL) extension file.

DOTV -1

Send the plot file to the default printer.

Measure the beam diameter at half-power (FWHP) to get beam parameters (**BMAJ**, **BMIN**, **BPA**) for the Cleaning process. If you do not specify these, **IMAGR** will choose them automatically by fitting a Gaussian beam with an elliptical cross-section. This is usually fine for uniformly-weighted images (**ROBUST** around 0), but may not be desirable for naturally-weighted images for which the beam is often rather non-Gaussian.

(29) Select boxes containing all of the line emission in the cube. Use as many boxes as necessary (up to 50). The better that you constrain the locations of real emission, the faster the Cleaning process will go. Note that **IMAGR** allows you to set the boxes interactively when you run with **DOTV** 1, which is important when Cleaning reveals emission initially lost in the sidelobes of the stronger objects. **TVBOX**

(30) Make a Clean cube of the line emission. Start by Cleaning just one or two channels (set with **BCHAN** and **ECHAN**) to ensure that your inputs are set correctly. If you don't want to waste time, it is a good idea to Clean only those channels with emission that is ≥ 4 times the RMS noise (Step 20). Use **SUBIM** and **MCUBE** to construct a full cube of the images later. Use the same parameters as in previous runs of **IMAGR**, but specify the beam parameters (**BMAJ**, **BMIN**, **BPA**) and control the Cleaning depth with **FLUX** and **NITER**. **IMAGR**

(31) Examine the Cleaned cube and do progressively deeper cleaning with **IMAGR** until you are satisfied with the result. The **DOTV** 1 option in **IMAGR** will help you reach this state more quickly. You can examine the Clean components with **PRTCC**. You should Clean until the total Cleaned flux converges. **TMovie**

(32) Correct for attenuation away from the center of the primary beam. Use your Cleaned cube as the input, the default parameters should be adequate, and the output is a corrected cube. **PBCOR**

(33) Back up the Cleaned cubes and the cubes after **PBCOR** to tape.

Print the contents of the tapes for your records. **FITTP**
PRTTP

B.3 Moment analysis and rotation curve of galaxies

After correcting for primary beam attenuation with **PBCOR**, the noise in the images will depend upon position. Because of this, you should use the uncorrected line cube for moment analysis.

- (34) The frequency axis can be labeled in either frequency or velocity units. Make sure that the desired units are chosen; use [IMHEAD](#) to check. Velocity is recommended for moment analysis. **ALTSW**
- (35) Transpose the axes to VEL-RA-DEC (or [FREQ](#)-RA-DEC) order. **TRANS**
`TRANSCOD '312'`
- (36) Generate images of the total emission ([MOM0](#)), the velocity field ([MOM1](#)), and the line width ([MOM2](#)) using the transposed cube as input. Be sure to exclude the end channels as they generally are very noisy. Try various values for the flux cutoff [FLUX](#) and the width of the smoothing functions (set by [CELLSIZE](#)) until you are satisfied with the results. **MOMNT**
- (37) Correct for the attenuation away from the center of the primary beam in the [MOM0](#) image, as in Step 28. If you computed the moment images with a velocity axis, use [ALTSW](#) to change to a frequency axis before running [PBCOR](#). **PBCOR**
- (38) Make contour plots of the [MOM0](#) and [MOM1](#) images; see Step 24. Note that [KNTR](#) can superpose contour and grey-scale plots as in Figure 8.4. **KNTR**
- (39) The task [GAL](#) allows you to generate a “tilted-ring” model rotation curve of two types from a galaxy’s velocity field or to fit single-parameter rotation curves to annuli of a specified width. The [EXPLAIN](#) file for this task describes all of the parameters in detail and contains general advice on how to obtain an optimum fit. **GAL**
The task [CUBIT](#), written by Judith Irwin, allows you to fit a rotating galaxy model to the full Cleaned image cube. The [EXPLAIN](#) file for this task describes all of the parameters in detail and contains general advice on how to obtain an optimum fit. **CUBIT**

B.4 Multi-frequency observations

B.4.1 General frequency information

For any *uv* data file, the frequency listed in the header information is the sky frequency of the center of the band (LINE or CH0) during the first scan of the observation. This is true regardless of whether you observed at a single frequency or multiple frequencies. After you [SPLIT](#) a multi-source file into single-source files, the frequency in the header refers to the sky frequency at the center of the band during the first scan *on that source*.

Corrections for the Doppler shift due to the rotation of the Earth can be taken into account within *AIPS*, if the data were observed at fixed frequency. Task [CVEL](#) may be used, but it requires that the spectra be well sampled in frequency.

B.4.2 Multi-frequency *uv* files

A simple rule of thumb: If you want to calibrate sources together, load the data with the same value of [FREQID](#) in [FILLM](#). If you want to calibrate sources separately, give them different [FREQIDS](#). For multi-frequency files, you should be sure to assign a different qualifier ([QUAL](#)) to each observing frequency (or velocity) with the [OBSERVE](#) program *before taking the observations*. There are essentially two types of multi-frequency observations:

1. Standard multi-frequency observations in which you want to do the entire calibration process separately for each frequency. When reading in such data with **FILLM**, set **CPARM(7) = 0**. This sets a different value of **FREQID** to data that differ by more than the maximum Doppler shift in a source in a day. During calibration, you can control which data you process by choosing the appropriate values for **FREQID** and **QUAL**. After calibration of each source/frequency, you may destroy the SN table to avoid using it to calibrate sources with different **FREQIDs**. For each source/frequency, you should create a new version of the calibration (CL) and bandpass (BP) tables (e.g., for the second source/frequency, you will create version 3 of the CL table and version 2 of the BP table). (It principle, **FREQIDs** may co-exist in single tables without interference, but if they are in carefully specified separate tables they cannot interfere with each other.)
2. Observations of, or affected by, Galactic emission or absorption, in which you want to combine data at different frequencies to do the calibration. Normally, these are observations in which the calibrator sources themselves are absorbed by Galactic HI around 0 velocity. It is *extremely important* that you assign a different qualifier to each frequency with the **OBSERVE** program. Then load the data with **FILLM** forcing a single value of **FREQID** by setting **CPARM(7) = -1**. In this case, the information that two observations with the same **FREQID** have different frequencies will be contained only in the qualifiers. Whatever data are loaded with the same value of **FREQID** will have the same reference frequency; it should be possible to average over the observing frequencies using the appropriate programs in **AIPS** (**CLCAL** and **BPASS**).

B.5 Additional recipes

B.5.1 Banana breeze pie

1. In a small saucepan, melt 1/3 cup **butter** or **margarine**. Add 1/4 cup **sugar** and 1/2 teaspoon **cinnamon**. Stir constantly over low heat until bubbles form around the edges of pan.
2. Remove from heat, add 1 cup **cornflake cereal** crumbs and mix well. Press mixture evenly into a 9-inch pie pan to form crust. Chill.
3. Beat 8 ounces softened **cream cheese** until light and fluffy. Add 1 15-ounce can **condensed milk** and blend thoroughly. Add 1/3 cup **lemon juice** and 1 teaspoon **vanilla**. Stir until thickened.
4. Slice 3 ripe **bananas** and line crust. Pour filling into crust and refrigerate for 2–3 hours or until firm. Do not freeze.
5. Slice 2 ripe **bananas**, dip in lemon juice and arrange on top of pie. Note. for a change of pace, use lime juice.

B.5.2 Banana cutlets

1. Peel 6 medium-ripe **bananas** and halve them crosswise.
2. Dip them in 1/3 cup **lemon juice** and then roll in 1 cup crushed **cornflake crumbs**.
3. Saute them in 3 tablespoons **butter** until a golden brown.
4. Serve on lettuce.

C A Step-by-Step Recipe for VLBA Data Calibration in *AIPS*

This appendix provides a step-by-step guide to calibrating many types of VLBA and **HSA** (High Sensitivity Array or Effelsberg, Arecibo, GBT, and phased **VLA**) experiments. Continuum strong-source or phase-referencing observations are included, as are simple spectral-line observations. This appendix applies specifically to data sets with full calibration transfer. There is an addendum (§ C.10) describing issues with flagging for non-VLBA data sets and other matters for cases in which not all calibration data are loaded by **FITLD**. It may often be used (with some modifications in loading amplitude data) for data sets containing other antennas. Simple VLBA utilities that go all the way up to and including fringe-fitting are described.

C.1 Quick Guide

This section is meant as a reminder for people who have some idea how to reduce straightforward VLBI data and a familiarity with *AIPS*. For further details see the rest of this appendix and/or Chapter 9. This section does not include things like **CVEL** or **AVSPEC** that might be useful for spectral line observations. Also, this section refers to procedures that can be loaded by typing **RUN VLBAUTIL**. All these steps are also performed by the **VLBARUN** (section § C.5) pipeline procedure.

1. **VLBALOAD** — load the data
2. **VLBAFIX** — fix the data (separate frequencies, merge tables, create index table, etc.)
3. **VLBAEOPS** — correct earth orientation parameters
4. **VLBATECR** — (if frequency < 10 GHz) remove dispersive delay from ionosphere
5. **VLBACCOR** — correct sampler threshold errors from correlator
6. **VLBAPCOR** or **VLBAMPCL** — (if more than 1 IF) correct instrumental delays, use **POSSM** to inspect correction
7. **VLBAPSS** — calibrate bandpass, use **POSSM** to inspect bandpass solutions
8. **VLBAAMP** — corrects any auto-correlation departure from unity caused by previous steps and performs a-priori amplitude calibration, use **SNPLT** to check nothing has gone wrong in the amplitude calibration.
9. **VLBAPANG** — correct parallactic angle
10. **VLBAFRNG** or **VLBAFRGP** — fringe fit data
11. **SPLIT** — apply all calibration and possibly spectrally average data
12. **IMAGR** — image data

C.2 Table Philosophy

AIPS follows an incremental calibration process on multi-source data sets. Calibration solutions are written to SN (“Solution”) tables, which can be inspected in various ways. `CLCAL` is used to apply an SN table and write a new CL (“Calibration”) table, which stores the cumulative calibrations. If the procedure in `VLBAUTIL` is used, much of this is hidden, however most procedures will report the new SN andor CL that are produced. The actual visibilities are not altered until the final calibration is applied using `SPLIT` (or `SPLAT`), which produces single-source (or multi-source) data sets that can be imaged. With this philosophy, it is easy to back up a step or two if errors are made in processing. Users should keep track of which tables contain which solutions and calibrations as they go through the calibration process.

A key verb to be aware of is `EXTDEST`, which can delete any unwanted table. For example, to delete SN version 3 from the data set catalogued as data set 1 on disk 1, type `INDISK 1; GETNAME 1; INEXT 'SN'; INVER 3; INP EXTDEST; EXTDEST`. *Beware of the fact that once a table is deleted, there is no ‘undelete’ function.*

C.3 Data set assumed in this Appendix

This appendix assumes a VLBA-only data set observed at several frequency bands (*e.g.*, 1.6, 2.3, and 5.0 GHz). To include data from the non-VLBA antennas see §C.10. It is also assumed that phase-referencing programs have been observed according to the philosophy discussed in detail in VLBA Scientific Memo No. 24. The hypothetical observation considered here contains the following sources:

- ‘`CAL-BAND`’ — fringe-search and bandpass calibrator
- ‘`CAL-AMP`’ — amplitude-check source
- ‘`CAL-POL`’ — polarization position angle calibrator
- ‘`STRONG`’ — strong target source
- ‘`CAL-PHASE`’ — phase-reference source
- ‘`TARGET`’ — weak target source, to be calibrated with `CAL-PHASE`

In the text below, table versions, such as SN version 1, are referred to as SN 1.

C.4 VLBA Utilities

Note that there are simple VLBA procedures (“front ends” to standard tasks) that will take the user all the way from data loading up to and including fringe-fitting. These are tremendous labor-savers for those working with reasonably straightforward data sets. For spectral line, use the procedures to calibrate a lower spectral resolution version of the spectral line data and copy the final calibration to the line set. To access the utilities, type `RUN VLBAUTIL` from inside *AIPS*. The procedures that we will use in this appendix are:

- `VLBALOAD`: loads VLBA data with simplified inputs
- `VLBASUMM`: makes summary listings of your data set
- `VLBAFIX`: Fixes VLBA data
- `VLBATECR`: automatically downloads and applies ionospheric corrections
- `VLBAEOPS`: automatically downloads and applies corrections to the Earth Orientation Parameters used by the correlator

- **VLBACCOR**: determines amplitude corrections caused by errors in sampler thresholds
- **VLBAPANG**: determines phase corrections for parallactic angles
- **VLBAPCOR**: determines instrumental phase corrections using pulse cals
- **VLBAMPCL**: determines instrumental phase corrections using **FRING**
- **VLBACPOL**: calibrates cross polarization delays
- **VLBABPSS**: does bandpass calibration
- **VLBAAMP**: does adjustment due to normalization errors and a-priori amplitude calibration
- **VLBAFRNG**: does global fringe fit using **FRING**
- **VLBAFRGP**: does global fringe fit for phase referenced experiments using **FRING**
- **VLBASNPL**: plots the SN or CL tables versus time
- **VLBACRPL**: plots the cross-correlation spectrum

There are two additional procedures that can make life easier, called **ANTNUM** and **SCANTIME**. **ANTNUM** will return the antenna number of the antenna corresponding to a certain character string. For example, in many data sets, typing **REFANT = ANTNUM ('BR')** will be the equivalent of typing **REFANT = 1**. **SCANTIME** will return the time range of a given scan number, for use in various programs. Typing **TIMERANG = SCANTIME(4)** will fill the eight-element array **TIMERANG** with the start and stop times of the 4th scan of a given data set.

Note that all of the **VLBAUTIL** procedures have **HELP** files with good discussions about when to use the simple procedures and when to use the tasks directly. Also, note that the procedures do not include data editing, which should be performed at appropriate points in the calibration process. You only need to type **RUN VLBAUTIL** once to access all of the procedures. If you run it again for any reason, it is a good idea to type **COMPRESS** immediately afterward to avoid overflowing *AIPS* symbol memory.

C.5 VLBA Pipeline

VLBARUN is a procedure which uses the VLBA calibration procedures (from **VLBAUTIL**) and some logic to calibrate and image VLBA data. **VLBARUN** attempts to make intelligent decisions on defaults, so it can be run fairly automatically if the names of the sources are known. If desired, **VLBARUN** will produce diagnostic plot files and write them to disk, creating an HTML file to ease examination of these files. Images will be produced, but no self-cal is done, so the images should be considered diagnostic in nature.

VLBARUN does all the calibration steps described in § C.1 and the next few sections. It does not do polarization calibration or flagging. Although **VLBARUN** is intended for simple VLBI observations, the addition of non-VLBA antennas should work if all the non-VLBA system temperature and gain curve information are loaded into the first TY and GC tables.

Sample inputs for procedure **VLBARUN** are:

> RUN VLBAUTIL C _R	to acquire the procedures used by VLBARUN .
> RUN VLBARUN C _R	to acquire VLBARUN .
> INDISK n ; GETN ctn C _R	to specify the input data file.
> OPTYPE 'CONT'	to say this is a continuum data set.
> CLINT 0	to use default.
> CHREFANT 'FD'	to set reference antenna to Fort Davis.

> TIMERANG 0	to have VLBARUN determine a good instrumental delay calibration scan.
> INVER 0	PC table to use, -1 if you want to force manual phase cal.
> CALSOR 'CAL-BAND', 'CAL-PHASE', 'CAL-AMP'	to list calibrators, bandpass calibrator <i>must</i> be first.
> SOURCES 'CAL-PHASE', 'TARGET'	to list phase referencing and target pairs.
> INFIL "	set if you want to apply a zenith delay file produced by DELZN
> SOLINT 0	.
> IMSIZE 512	to use default.
> FACTOR 0	to make images and specify size of target images.
> DOPLOT 1	to make calibrator images 128x128.
> OUTFILE '/directory/name	to make some diagnostic plots.
> OUTTXT 'email@somewhere.edu	to specify directory for output plots. If this is set then plots are written out from <i>AIPS</i> and organized in an HTML file for easy viewing. Preferably, this directory should be empty before the procedure is used.
> BADDISK 0 C _R	to specify an e-mail address if the user wants to be notified when the pipeline is done.
> VLBARUN C _R	to specify which disks to avoid for scratch.
	to run the procedure.

VLBARUN will then run and produce the requested number of diagnostic plots. For details on the plots produced for each level of **DOPLOT** see **EXPLAIN VLBARUN**. If an e-mail address is specified then a **VLBARUN DONE** or **VLBARUN FAILED** message will be sent. However, the **VLBARUN FAILED** message will only be sent if **VLBARUN** failed because of problems with the inputs, if **VLBARUN** failed because a task it was running failed this message is not sent. It is highly recommended that the user read the explain file for **VLBARUN**

C.6 Data Loading and Inspection

1. Load the data using **VLBALOAD** (which is a very simplified **FITLD**). **CLINT** should be set so that there are several CL table entries for each self-calibration or fringe-fitting interval anticipated; this will minimize interpolation error during the calibration process. However, setting **CLINT** too short will result in a needlessly large table. Somewhere between **CLINT** = 0.25 and **CLINT** = 1.0 is about right. A **FITLD** parameter that is set automatically in **VLBALOAD** is **WTTHRESH** = 0.7, which results in irrevocable discarding of all data with playback weight less than 0.7. The only way around this is to use **FITLD** explicitly.
2. Correct data with **VLBAFIX**. *If necessary*, **VLBAFIX** sorts (with **MSORT**), splits into different frequencies (with **UVCOP**), fixes the polarization structure (with **FXPOL**), and indexes (with **INDXR**) the data. **VLBAFIX** will also correct for subarrays (with **USUBA**), but you must tell it to do so. There are only 2 inputs of interest in **VLBAFIX**, **CLINT**, the CL table interval, and **SUBARRAY** which should be set to 1 if there are subarrays and 0 if not. This is a very benign procedure, it can be run on every data set read into *AIPS* and will only perform the necessary fixes. Note that, if the data are split into different frequencies, the flag table is applied and deleted.
3. At this point it is a good idea to get a listing of the antennas and scans in your data by running **VLBASUMM**. **VLBASUMM** runs **PRTAN** over all antenna tables and **LISTR** with **OPTYPE='SCAN'** and gives a choice of writing a text file to disk or sending the listing to a printer.

4. Apply corrections to the Earth Orientation Parameters (EOPs). VLBI correlators must use measurements of the Earth Orientation Parameters (EOPs) to take them out of the observations. These change slowly with time and therefore the EOPs used by the correlator must be continually updated, and they are generally best two weeks or more after the observation. Since we try to correlate observations as quickly as possible, it is likely that the EOPs used are not the most accurate. Therefore it is recommended that all phase-referencing experiments be corrected for this possible error. The procedure `VLBAEOPS` will do this correction. `VLBAEOPS` automatically downloads a file with correct EOPs and runs `CLCOR` to correct the EOPs.
5. Apply ionospheric corrections, if desired, with or `VLBATECR`. This procedure automatically downloads Global Positioning System (GPS) models of the electron content in the ionosphere to and uses them to correct the dispersive delays caused by the ionosphere. It is particularly important for phase referencing experiments at low frequency. We recommend `VLBATECR` for all experiments at 8 GHz or lower. `VLBATCOR` is only as good as the ionospheric model, so it is a very good idea to compare the corrected and uncorrected phases using `VPLT`. To inspect the phases using `VPLT`, use options `BPARM = 0, 2; APARM=0; DOCAL=1; GAINUSE=highest CL table`. The phases should not wind as much (although they will probably not be completely flattened), when the corrected CL table is applied. To see the corrections themselves, use `SNPLT` on the new CL table setting `OPTYPE = 'DDLY'`.
6. For a simple spectral-line data set, or any data set with high spectral resolution, it is a very good idea to average the data set smaller number of channels before deriving the calibration parameters. Otherwise, the calibration tasks may take forever to run. It is recommended that you quickly inspect the channels of interest for your line data (e.g., with `UVPLT`) for high points. Remove obviously high amplitudes with `CLIP` (or e.g., `UVMLN`) before averaging. Inspect the full resolution data also for high delays and fringe rates. Spectral averaging in such cases may not be acceptable. Continue calibration on the averaged data set as if it were a continuum set. There is a better method to calibrate spectral line data described in § 9.5.6 and § 9.5.7.10, but the one used here is simpler and will usually give acceptable results. To reduce the data-set size, run the task `AVSPC` with `AVOPTION = 'SUBS'`. For example, to average IFs with N_{chan} down to 32 channels, set the adverb `CHANNEL = $N_{\text{chan}}/32$` (e.g., to average from 2048 to 32 channels, use `CHANNEL = 64`).

C.7 Amplitude and Delay Calibration

Amplitude calibration uses measured antenna gains and system temperatures (T_{sys}), as well as finding a correction for voltage offsets in the samplers. Even though this is substantially still the case, we now recommend a somewhat different amplitude calibration procedure than in the past based on VLBA Scientific Memo #37 (Walker (2015)).

Before amplitude calibration is done there must be information for all antennas in the gain curve (GC), system temperature (TY), and weather (WX) tables (weather tables are needed for the opacity correction in `APCAL`). Missing T_{sys} and gain curve information can usually be obtained in `ANTAB` format and loaded with the `AIPS` task `ANTAB`. § C.10 has information on including non-VLBA calibration information, if they are not already included in the data. § C.11 has details on how to incorporate the pre-`EVLA VLA` T_{sys} and gain curves. Otherwise consult § 9.5.2.3.

1. Correct sampler offsets and apply amplitude calibration by running `VLBACCOR`. The procedure `VLBACCOR` runs `ACCOR`, `SNSMO`, and `CLCAL`. `ACCOR` uses the autocorrelation to correct the sampler voltage offsets. After `ACCOR` creates an SN table, `SNSMO` smooths the table in order to remove any outlying points. Then the SN table is applied to the highest CL table using `CLCAL` (using `INTERPOL='2PT'`), and a new CL table is created.

2. Next, the instrumental delay residuals must be removed. These offsets or “instrumental single-band delays” are caused by the passage of the signal through the electronics of the VLBA baseband converters or MkIII/MkIV video converter units. There are two different methods to remove these instrumental delays, one for the case where you have pulse-cal information for some, but not necessarily all, of your antennas; and one for the case where you have no pulse-cal information at all. Note that the preferred method for continuum experiments is to use the pulse-cals, since they correct the instrumental delay over the whole experiment, rather than on a short scan. Spectral-line observers would have switched off the pulse-cals as they interfere with line observations, so they are forced to use the second (strong source) method. For VLBA continuum experiments before April 1999, you can load the pulse-cal data using [PCLOD](#); consult the § 9.5.2.

- For the case where you have some pulse-cal information, run [VLBAPCOR](#). [VLBAPCOR](#) is another procedure which runs quite a few tasks, [PCCOR](#), [CLCAL](#), [FRING](#) (sometimes) and [CLCAL](#) again (sometimes). [PCCOR](#) extracts pulse-cal information from the PC table and creates an SN table. Then [CLCAL](#) is run to apply that SN table to the highest CL table, creating a new CL table. If there are antennas that do not have information in the PC table, or their PC entries are wrong, then [VLBAPCOR](#) can run [FRING](#) on a short calibrator scan (input [TIMERANGE](#)). The SN table from [FRING](#) contains corrections for the antennas left out of the PC table, and is applied to CL table without corrections from [PCCOR](#), and added to the CL table with the PC corrections. For the simplest case of all VLBA antennas, the inputs for [VLBAPCOR](#) should be [TIMER=time range on CAL-BAND with good fringes for all baselines](#); [REFANT=n](#); [SUBARRAY=0](#); [CALSOUP='CAL-BAND'](#); [GAINUSE=0](#); [OPCODE=''](#); [ANTENNAS=0](#). For the case where you have the VLA (in this example antenna 11), which does not have pulse-cals, your inputs should be the same as above except [OPCODE='CALP'](#); [ANTENNAS=11,0](#). For the second case it is important that there are no “Failed” solutions from [FRING](#); if there are failed solutions, then you should delete the tables that were created and find another [TIMERANG](#) with good fringes to the [REFANT](#). Also see [EXPLAIN VLBAPCOR](#) for a detailed description of the steps involved with using pulse-cals and [FRING](#) without using [VLBAPCOR](#).
- The alternate method is to use solve for the phase cals manually with [VLBAMPCL](#). This method uses the fringes on a strong source to compute the delay and phase residuals for each antenna and IF. [VLBAMPCL](#) runs [FRING](#) to find the corrections and then [CLCAL](#) to apply them. If there is no calibrator scan that includes all antennas then there is an option to run [FRING](#) and [CLCAL](#) again on another source and/or time range in order to correct the antenna(s) not corrected by the first scan. For the simplest case where all antennas have strong fringes to CAL-BAND, set [TIMERANG = time range of scan on calibrator with strong fringes to all antennas](#); [REFANT = n](#); [CALSOUP = 'CAL-BAND'](#); [GAINUSE = highest CL table](#); [OPCODE = ''](#). There must be no “Failed” solutions from [FRING](#), if there are any failed solutions the data from that antenna/IF will be completely deleted from the data.
- Now you *must* check the results of correcting your instrumental delays using [VLBACRPL](#) or [POSSM](#). Set [GAINUSE=highest CL table](#), and plot cross-correlations ([VLBACRPL](#) will do this for you). The plotted cross-correlations should show the phase slope removed from each IF and there should no longer be a phase jump between IFs, although the phase does not have to be at 0°. If you do a “manual” instrumental delay correction (*i.e.*, you used [FRING](#), not [PCCOR](#)); then the phases far in time from the scan on which [FRING](#) was performed may have a small slope and a small phase jump between the IFs. Also non-zero phase slopes still may be seen at low elevations, where the atmosphere causes additional delay residuals, or for low-frequency observations where the ionospheric delay varies. If you see significant phase slopes, or phase jumps between IFs on *any* baseline, then the instrumental phase corrections have not worked and you need to figure out why and start again.

3. Next you must calibrate the bandpass shapes. To do this, run [VLBAPSS](#) on the bandpass calibrator, [CAL-BAND](#). Make sure that the spectral line data for the bandpass calibrator is clean and devoid of high points, using [UVPLT](#) or [SPFLG](#). Inputs for [VLBAPSS](#) are [CALSOUP = 'CAL-BAND'](#) and a model for your calibrator if you have one. Then you must examine the BP table using [POSSM](#) by setting [APARM\(8\)=2](#).

4. Now it is time to finish the amplitude calibration by running **VLBAAMP**. The procedure **VLBAAMP** runs several tasks, **ACSCL**, **SNSMO**, **CLCAL**, **APCAL** and **CLCAL**. After the above steps the calibrated autocorrelation amplitudes will be offset from unity, **ACSCL** corrects this offset. **ACSCL** creates an SN which is then smoothed by **SNSMO** and the **CLCAL** is run to apply the calibration to the next CL table. Then to finalize the amplitude calibration, **APCAL** is run on the highest TY and GC tables, and a new SN table is created. Adverb **DOFIT** controls whether **APCAL** also uses the weather tables to fit and correct for opacity. It is desirable to perform an atmospheric opacity correction at high frequencies, particularly if very accurate source fluxes are needed. See § 9.5.4.6 for a more detailed discussion of **APCAL**. Lastly, **VLBAAMP** runs **CLCAL** to apply the amplitude calibration SN table to the CL created by the last run on **CLCAL**. **VLBAAMP** will print messages telling you about the new tables it has created. To keep track of your tables, it is important to copy these messages.

5. At this point it is a *very* good idea to examine your calibration.

- Run the task **SNPLT** or procedure **VLBASNPL** (which is a very simplified **SNPLT**) to examine the tables created by **ACCOR**. Use **INEXT='CL'**; **OPTYPE= 'AMP'**; **INVERS=CL-table-with-sampler-offsets**; **DOTV=1** (to display to the TV; for a hardcopy use **DOTV=-1** and **LWPLA** to print the plot files). The solutions that **SNPLT** plots should be close to 1000 milligain or 1 gain.
- Run **SNPLT** or **VLBASNPL** to examine the amplitude calibration. This time look at the SN table that **APCAL** created. Use **INEXT='SN'**; **OPTYPE= 'AMP'**; **INVERS=highest SN table**; **DOTV=1** for **VLBASNPL**; or to inspect IF m , use **SNPLT** and **BIF = m** ; **EIF = m** ; **OPTYP = 'AMP'**; **INVER = 1**; **INEXT = 'SN'**; **OPCODE = ''**; **NPLOT = 10**; **DOTV = 1**; **GO SNPLT**. For a hardcopy, use **DOTV = -1**; **GO SNPLT**; **GO LWPLA**. Plotted amplitudes are the square-roots of the system-equivalent flux densities (SEFDs), in Jansky, where the SEFD is the flux density of a source that would double the system temperature. (Low numbers are good!) At centimeter wavelengths, VLBA antennas have SEFDs near 300 Jy, so gains above 30° elevation should be near 17–18 and should vary slowly and smoothly with time (*i.e.*, change in elevation) for an individual source. To look at the input system temperatures, run **SNPLT** with **OPTYP = 'TSYS'**; **INEXT = 'TY'**; **INVER = 0**. On rare occasions, you might find clearly discrepant points that have leaked in from a different frequency band. In that case, you can use task **SNEDT**, or the clipping option of **SNSMO**, to get rid of the bad points. You may notice that at low elevations the gains on individual antennas are high. All data below a given elevation can be flagged by running **UVFLG**; *e.g.*, to flag all data below 10°, run **UVFLG** with **APARM(4)=0** and **APARM(5) = 10**. Note that FG tables are not applied to tables, so flagged data still may have points plotted by **SNPLT**. The T_{sys} measurements are also a very good diagnostic of bad data from poor weather, equipment failures, *etc..* If there are time ranges of unusually high or low T_{sys} you may consider flagging those time ranges using **UVFLG**. Be particularly suspicious of patches of unusual gains at only one IF or **STOKES** of an antenna. Remember, one of the best things you can do for your final result is to get rid of bad data.
- You may wish to use your favorite method of inspecting data for flagging (*e.g.*, **EDITR**, **TVFLG**, **IBLED**). On-line flags are already included in FG 1 unless they were applied as the data were split into separate frequencies. For example, run **EDITR** with inputs **SOURCES= 'CAL-BAND', ''** (do each source separately); **DOCAL=1**; **GAINUSE=highest CL table**; **FLAGVER=0**; **OUTFGVER=0**; **DOTWO=1**; **ANTUSE=1,2,3,4,5,6,7,8,9,10**. Once you gain experience you might want to set **CROWDED=1** which allows plots of all polarizations and IFs in one plot; this can speed up editing significantly. Look for anomalously high or low amplitudes, remember there can be a slow change in amplitude with time due to source structure. Some people do no additional flagging at this stage, but later use the results of fringe-fitting and visibility plots of calibrated data to point the way to bad data.

6. For spectral-line experiments needing velocity accuracy better than 1 km/s, a Doppler correction should be performed. Use **CVEL**; see § 9.5.4.5 and § 9.5.5 for details.
7. This is a useful time to run **TASAV** to save all your ancillary tables to another file. If you foul up the calibration, the relevant tables can be copied back using **TACOP**.

C.8 Rate and Phase Calibration

Now that the data have calibrated amplitudes, the next step is to do the calibration of the antenna rates and phases. This section describes that process.

1. Correct the antenna parallactic angles, if desired, using [VLBAPANG](#). The RCP and LCP feeds on alt-az antennas will rotate in position angle with respect to the source during the course of the observation (all VLBA and [VLA](#) antennas are alt-az). Since this rotation is a simple geometric effect, it can be corrected by adjusting the phases without looking at the data. You *must* do this correction for polarization experiments and phase referencing experiments. Parallactic angles are important for phase referencing because the parallactic angle difference between calibrator and target is different at different stations which leads to an extra phase error which can be corrected. [VLBAPANG](#) copies the highest numbered CL table with [TACOP](#) and then runs [CLCOR \(OPCODE = 'PANG'; CLCORPRM = 1,0\)](#). [VLBAPANG](#) has no inputs that require discussion. Be sure to correct the parallactic angles before any of the following steps. Again keep track of which CL tables add which correction.
2. Now you must remove global frequency- and time-dependent phase errors using [FRING](#) or one of the procedures which use this task, [VLBAFRNG](#) or [VLBAFRGP](#). This cannot be done simply for spectral-line sources, so the practice here is to determine delay and rate solutions from the (continuum) phase-reference sources and interpolate them over the spectral line observations. The procedures run either [FRING](#) along with [CLCAL](#). [VLBAFRNG](#) and [VLBAFRGP](#) use [FRING](#), with [VLBAFRGP](#) specifically for phase referencing. For all these procedures, if the [SOURCES](#) adverb is set, then [CLCAL](#) is run once for each source in [SOURCES](#). For the phase-referencing procedure ([VLBAFRGP](#)), any source that is in the [SOURCES](#) list that is *not* in the [CALSOUR](#) list will be phase referenced to the *first* source in the [CALSOUR](#) list. These procedures will produce new (highest numbered) SN and CL tables. Since it is probably best to run [CLCAL](#) on each source separately, [SOURCES](#) should always be set. To use [VLBAFRGP](#) for a simple phase referencing experiment (remember that [CAL-PHASE](#) is the phase reference calibrator), set [CALSOUR='CAL-PHASE', 'CAL-BAND', 'CAL-AMP', 'CAL-POL', 'STRONG'; GAINUSE=highest CL table; REFANT=n; SEARCH 9 4 1 3 5 6 7 8 10; SOLINT=coherence time; DPARM\(7\)=1 \(if a polarization experiment\); SOURCES='CAL-PHASE', 'CAL-BAND', 'CAL-AMP', 'CAL-POL', 'STRONG', 'TARGET'; INTERPOL='SIMP'](#). For this example, [FRING](#) will be run on the sources in [CALSOUR](#) and then [CLCAL](#) will be run 6 times, with all of the sources except TARGET referenced to themselves and TARGET referenced to [CAL-PHASE](#), using interpolation method SIMP. For a non-phase-referencing experiment you would use [VLBAFRNG](#) with inputs the same as above except for [SOURCES](#), which would not contain TARGET. The results will be the highest SN and CL tables. The [INTERPOL](#) to use is a personal preference. You might want to restrict the channel range slightly using [BCHAN](#) and [ECHAN](#), since the channels at the high end of each IF will have lower SNR, due to the cutoffs in the bandpass filters. For a data set with 16 channels per IF, numbered from 1 to 16, setting [ECHAN](#) to 14 or 15 may be worth trying. Note that some people like to run [CALIB](#) rather than [FRING](#) or [KRING](#) for this stage of phase-referencing observations, but fringe fitting is recommended, as it solves for rates.

The above fringe fit may take a bit of time, depending on the computer and the spectral resolution. Then, use [SNPLT](#) or [VLBASNPL](#) to inspect the solutions in the SN table. It's not totally out of the question that some data will be found that need flagging, which can be done with [UVFLG](#). In that case, it's a good idea to delete the last SN and CL table and re-run [VLBAFRGP](#) or [VLBAFRNG](#).

This fringe-fitting stage is the most likely place where things can go wrong, for reasons that are not immediately apparent to the observer. Below, a few common examples are listed.

- **Many solutions failed.** The source may be too weak, or the coherence time too short. Try increasing or decreasing [SOLINT](#). Or narrow the search window. For most VLBA data, [DPARM\(2\) = 400](#) and [DPARM\(3\) = 60](#) should be a good first step, though the rate window specified in [DPARM\(3\)](#) is proportional to the observing frequency, and may need to be larger at 22 GHz and above. For more options you could try running [FRING](#) and reduce the SNR threshold with

[APARM\(7\)](#) or averaging the RR and LL ([APARM\(3\)=1](#)). One last thing to try is to abandon [FRING](#) and solve for the phases in [CALIB](#), obviously not ideal since you will not get rates or delays, but it is a hit worth taking if the the calibration can be salvaged.

- **Some antenna has low SNR, and may cause an entire set of solutions to go bad.** This typically happens because an antenna should have been flagged. A common cause is when OV is looking at the White Mountains, and neither the on-line system nor the astronomer has flagged the data. Then, you need to run [UVFLG](#) and re-run [VLBAFRGP](#) or [VLBAFRNG](#).
 - **There are discrepant delay/rate solutions.** Look at the solutions you believe, and try [VLBAFRGP](#) or [VLBAFRNG](#) again with [DPARM\(2\)](#) and [DPARM\(3\)](#) specified appropriately. Full widths are specified, so if the good solutions fall between +15 mHz and -15 mHz, use [DPARM\(3\) = 30](#). (Actually, you should use a value somewhat larger to allow some margin.) It may be that an antenna is suffering from radio-frequency interference, so some channels and/or IFs will need to be flagged.
 - **Some solutions are outside the specified delay/rate range.** This can happen because the initial coarse fringe search uses the range specified by [DPARM\(2\)](#) and [DPARM\(3\)](#), but the least-squares solution can take off from there and go elsewhere.
 - **Delays and rates for some station change rapidly near the beginning or end of the observation.** This may be caused by low elevation at the relevant station. Depending on how desperate you are to include low-SNR data, you may wish to flag some time range, or flag all data at elevations below 5°, 10° or even 20° (particularly at high frequencies or for phase referencing) with [UVFLG](#).
 - **Phases wrap rapidly, particularly on the phase-reference source, CAL-PHASE.** There may not be a lot you can do about this initially, because it's possible that the tropospheric delay just changed too fast for the cycle time used in the observation, especially at low elevation. However, you may wish to note the times and antennas when the phase connection is best (typically the southwestern antennas near transit). Later, when imaging the program source, it can be helpful to image with a subset of antennas and time ranges, then use that initial image to self-calibrate the rest of the data.
3. Use [SNPLT](#) or [VLBASNPL](#) to inspect the interpolation of the phases in the CL. When you inspect the CL table notice any phase wraps that seem out of place. The human eye is better at pattern matching than a computer and these phases may be in error. If so you might want to run [CLCAL](#) independently and try another interpolation method or you might want to edit the CL table. Remember that this is your last calibration table; you want to get rid of any bad calibration now before applying it to the data. Getting rid of spurious wraps in the final CL table ([SNEDT](#)) or flagging the data associated with fast changing phases ([SNFLG](#)) will improve your final image more consistently than anything else, *particularly* for phase referencing.

C.9 Final Calibration Steps

1. If you used [AVSPC](#) to reduce the size of the data set used in determining calibration, you must copy your final calibration tables back to the full-size data set. This can be done with task [TACOP](#). For bookkeeping purposes, it may be best to copy over all the CL tables with the same table numbers in both the averaged and un-averaged data sets. Copy the FG table as well, since any data which are bad in the averaged dataset will be bad in the full resolution dataset. After inspecting the data with [UVPLT](#) or [VPLOT](#), run [EDITR](#), [TVFLG](#), [CLIP](#), [SPFLG](#), or other data editor to edit the bad data from the calibrated spectral-line dataset.
2. After you have corrected the bandpass for spectral line data, you may want to correct for the change in frequency by the motion of the antennas with respect to the Sun *etc..* This is done with [CVEL](#), after the source velocities are entered in the SU table with [SETJY](#). For a detailed description see § 9.5.5.

3. Polarization calibration still remains, if desired, and if all the appropriate calibration sources were observed. This can be done in a variety of ways; see § 9.5.7 for details.
4. Finally, apply the calibration to the visibility data and make single-source data sets using **SPLIT**. (Some people might wish to use **SPLAT** to average over time as well as spectral channel.) Inputs for a continuum observation are **SOURCES** = ' ' ; **BIF** = 0; **EIF** = 0; **DOPOL** = -1 (or 1 if polarization calibration was attempted); **DOBAND** = 1; **DOUVCOMP** = 1; **NCHAV** = 0; **APARM** = 2,0; **DOCALIB** = 1; **GAINUSE** = *highest CL table*. For a spectral-line observation, set **APARM** = 0, because you don't want to average over frequency. Use **OUTDISK** and **OUTCLASS** as appropriate for your computer and record-keeping purposes.

The single-source data sets are now ready for imaging and possible self-calibration. At this point, it is a good idea to look at the amplitude check source 'CAL-AMP' using tasks such as **UVPLT**, **WIPER** or **VPLOT** in order to see if there are any antenna gain calibrations that must be adjusted. Doing a **WIPER** for each target source is a good idea also, because there may be discrepant amplitude points due to interference or poor fringe fits (among other things). The task **WIPER** makes **UVPLT** like plots but allows flagging.

C.10 Incorporating non-VLBA antennas

Many non-VLBA telescopes have their amplitude calibration incorporated in the tables loaded by **FITLD**. We retain the following sections for the case where some telescopes are not included in calibration tables for whatever reason.

The phased-**EVLA** came on-line in February 2013. Its calibration is very different from the old **VLA**. With the phased-**EVLA** either an **ANTAB** style file was provided to the observer or the calibration information is attached to the data. This calibration is generated from the switched power measurements attached to the **EVLA** only data. There is no need for special calibration steps. However, the observer may wish to improve the calibration by editing the switched power and recreating the **ANTAB** style file. More information on this can be found in § 9.5.2.4. There is a separate section (§ C.11) dealing with pre-**EVLA VLA** data.

C.10.1 Loading T_{sys} and Gain Curves

Most telescopes that do not have T_{sys} and gain curve information attached to the data will provide the user with a text **ANTAB** format file that can be loaded with **ANTAB**. Please see **EXPLAIN ANTAB** for more information on the **ANTAB** format. Follow the steps below to load calibration information and amplitude calibrate the data.

1. To prevent any chance of having to re-run **FITLD**, first save the VLBA TY and GC (and all your other) tables by running **TASAV**. Then run **ANTAB** with **CALIN** set to the **ANTAB** file and setting **TYVER** and **GCVER** to the highest numbered tables (usually 1). You don't want to leave **TYVER** and **GCVER** equal to 0 because that causes **ANTAB** to create new TY and GC tables with only the antennas in the file you are loading. If **ANTAB** fails, it is most likely caused by the input file not being formatted correctly. Perhaps you needed to add an INDEX entry or the file wasn't in true **ANTAB** format. Also, if **ANTAB** fails check that the TY and GC tables have not been corrupted. If they have you should delete the tables and copy the original tables from the **TASAV**'ed file with **TACOP**.
2. Now, run **VLBAAMP** as described in § C.7 to combine the gain and system temperature information for all antennas into the appropriate SN and CL tables. Then, use **SNPLT** or **VLBASNPL** as described in § C.7 to make sure that the resulting SN now contains amplitude calibration for all the antennas and IFs included in the project.

C.10.2 Pointing Flags

The **VLA**, GBT, AR and possibly other telescope's on-line systems produce only recorder-related flags, not pointing flags. Thus, for example, there are no flags for when the telescope is not on-source. This can lead to a large amount of bad data especially if you are changing source frequently. However, the VLBI scheduling program **SCHED** creates a `*.flag` file which contains estimates of how long it takes these antennas to slew. The `*.flag` file will also contain flags for all antennas in the experiment, so it is best to remove the flags that pertain to the VLBA antennas. The flag file can be read in with **UVFLG** using the **INTEXT** adverb.

C.11 Pre-EVLA VLA data

The method described in this section is only meant for VLA data from before February 2013 (before the phased-EVLA came on-line). The observation being calibrated may have incorporated either a single **VLA** antenna or the phased **VLA**, but the amplitude calibration parameters for the **VLA** were not transferred automatically. You will need to obtain an input text file for the **VLA** then run **ANTAB** before **APCAL**. Before February 2013, the gains and system temperatures for this file, in an appropriate format, were supplied in a file called `xxxxxcal.y.gz`, where 'xxxxx' is the observation code (e.g., 'bm120'), located at <http://www.vlba.nrao.edu/astro/VOBS/astronomy/mmmyy/yyyy/>. That file contains instructions on editing the file to get correct inputs. For a phased array or a 1.3-cm observation in which 3 antennas are used, follow the instructions in § C.11.2; for a single antenna, use § C.11.1.

C.11.1 Single VLA Antenna

The method described in this section is only meant for VLA data from before February 2013. Depending on the age of your observation, you may have to add an INDEX entry within the TSYS card (**Do not separate the INDEX entry from the TSYS entry by a "/" !!!**), un-comment the **GAIN** line for your particular observing frequency, and un-comment the TSYS line. There are examples of INDEX entries in the comments at the head of the file.

Beginning in June 2003, the INDEX, **GAIN**, and TSYS information in this table are reformatted to be directly acceptable to **AIPS**. You should check the times in the text file to make sure that your observation has been properly described. Only a few special cases will require editing of the file; in most cases you are able to invoke **ANTAB** with no editing. Once you are satisfied with the **ANTAB** file, load the data following the directions in § C.10.1.

C.11.2 Phased VLA

The method described in this section is only meant for phased-VLA data from before February 2013. The **VLA** may be phased on a program source ('STRONG'), or may be phased on a phase-reference source ('CAL-PHASE'), with the resulting solutions applied to the program source ('TARGET'). Rather than recording a system temperature, the **VLA** system will record a ratio of antenna temperature to system temperature, which will vary as the array phases up. In order to convert the ratio of antenna and system temperatures to a usable gain, the flux density of some source will be needed.

1. Load and calibrate the **VLA** data by standard means (see Chapter 4). Determine the flux density of a relevant strong source, usually either 'STRONG' or 'CAL-PHASE'. Then, on the VLBI data set,

insert the flux density of this source into the SU table using **SETJY**. For example, if the source is ‘CAL-PHASE’ and its flux density is 0.432 Jy, run **SETJY** with **SOURCES** = ‘CAL-PHASE’; **BIF** = 0; **EIF** = 0; **ZEROSP** = 0.432,0; **OPTYPE** = ‘ ’.

2. Edit the input file as indicated above for a single **VLA** antenna. Again, an INDEX line, a GAIN line, and a TSYS line must be checked (after June 2003) or be created or un-commented. The GAIN line is independent of observing band (the source flux is used to determine the gain), and the TSYS line should include the parameter ‘SRC/SYS’, indicating that the ratio of antenna temperature to system temperature is being supplied.
3. Run **ANTAB** to read in the input file of amplitude calibration parameters. Then run **VLBACALA** to put this in an SN table. Both steps are essentially the same as for a single **VLA** antenna (see § C.11.1). The most likely problem is that **APCAL** in **VLBACALA** will fail because you forgot to enter a source flux density using **SETJY**, although the error message may not always make this obvious.
4. Run **VLBASNPL** or **SNPLT** to inspect the resulting SN table, as for the single **VLA** antenna. In this instance, you should see that the phased **VLA** is very sensitive. If the phasing worked well at centimeter wavelengths, the amplitude should be near 4 or 5 instead of the value of 17 or 18 seen for a single VLBA antenna. At the start of scans where the **VLA** is being phased, you may see a rapid change in the amplitude gain (toward smaller numbers) as the antenna phases are brought into alignment. The SN table should be inspected very carefully, because there may be data that should be flagged when the **VLA** phasing did not work well. Three possible reasons for poor phasing are (1) the source is too weak; (2) the troposphere is misbehaving; or (3) there was radio-frequency interference at the **VLA**.

C.12 Summary for non-VLBA antennas

Following the insertion of the amplitude solutions for the non-VLBA antennas, you can return to follow the standard path for calibration of VLBA data. Although the procedures from here on are identical to the VLBA-only case, the observer may wish to pay attention to several issues.

1. Many non-VLBA antennas are more sensitive than a single VLBA antenna, it can be a good idea to use the most sensitive antenna as the reference antenna for fringe-fitting.
2. Non-VLBA antennas may have a larger delay offset than VLBA antennas. The user should pay close attention to the fringe fits, and be aware of the possibility that non-VLBA antennas may have larger residual delays and rates than a VLBA antenna.
3. Many non-VLBA antennas do not slew as rapidly as the VLBA. The FG table supplied by calibration transfer may not include all the on-line flags, and may not incorporate information about the pointing of non-VLBA antennas, and when they arrive on source. Therefore, some judicious flagging by the user may be necessary. See § C.10.2 for a discussion of applying flags produced by the scheduling software SCHED.
4. The elevation limit for non-VLBA antennas is generally different and usually higher than the VLBA’s limit ($> 2^\circ$) which may cause sources to rise and set at different antennas at different times. For example, the **VLA** has an elevation limit of 8° , this means that a source will rise later and set earlier at the **VLA** than it does at Pie Town or Los Alamos.

C.13 Some Useful References

1. Chatterjee, S., "Recipes for Low Frequency VLBI Phase-referencing and GPS Ionospheric Correction," VLBA Scientific Memo No. 22, May 1999.
<http://www.vlba.nrao.edu/memos/sci/>
2. Ulvestad, Jim, "VLBA Calibration Transfer with External Telescopes, Version 1.1," VLBA Operations Memo No. 34, July 30, 1999. <http://www.vlba.nrao.edu/memos/vlba/vba.oper.txt>
3. Ulvestad, Jim, "A Step-By-Step Recipe for VLBA Data Calibration in AIPS Version 1.3" VLBA Scientific Memo No. 25 (the basis of this appendix), January 2, 2001.
<http://www.vlba.nrao.edu/memos/sci/>
4. Ulvestad, Jim, Greisen, Eric W., Mioduszewski, Amy "AIPS Procedures for Initial VLBA Data Reduction, Version 2.0" AIPS Memo No. 105 April 26, 2001.
[# MEMOS](http://www.aips.nrao.edu/aipsdoc.html)
5. Wrobel, J. M., Walker, R. C., Benson, J. M., & Beasley, A. J., "Strategies for Phase Referencing with the VLBA," VLBA Scientific Memo No. 24, June 2000.
<http://www.vlba.nrao.edu/memos/sci/>

C.14 Additional recipes

C.14.1 Golden mousse

1. Combine 1 cup mashed ripe **bananas**, 2 tablespoons **orange juice**, 1/4 cup shredded **coconut**, 3 tablespoons **brown sugar**, a few grains **salt**, and 1/8 teaspoon grated **orange rind**.
2. Whip until stiff 1 cup **heavy cream**.
3. Fold whipped cream into fruit mixture and turn into freezing tray. Freeze rapidly without stirring until firm.

C.14.2 Mexican chicken vegetable soup with bananas

1. In large, covered kettle, over medium-low heat, simmer 4 pounds cut up **stewing chicken**, 1/c cup coarsely chopped **onion**, 1 teaspoon **salt**, and 4 cups of hot **water** for 2 hours or until chicken is tender.
2. Remove chicken to cutting board; cut meat from bones into chunks; discard bones. Skim any fat from surface of broth.
3. Add chicken, 1/2 cup chopped **celery**, 1 12-ounce can whole-kernel **corn** and 1 16-ounce can **tomatoes** to soup. Continue simmering, covered for 10 minutes. Season to taste.
4. Five minutes before serving, peel 4 firm (green-tipped) **bananas**, slice diagonally into 1-inch slices.
5. Add sliced bananas to soup, continue cooking just until bananas are tender. Serve immediately.

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

C.14.3 Orange baked bananas

1. Mix in a saucepan 1/2 cup firmly packed **brown sugar**, 1 tablespoon **cornstarch**, 1/8 teaspoon **cinnamon**, and a few grains **salt**.
2. Add gradually, blending in 3/4 cup boiling water.
3. Bring rapidly to boiling and cook about 5 minutes or until sauce is thickened, stirring constantly.
4. Remove from heat and blend in 1½ teaspoons grated **orange peel**, 1/4 cup **orange juice**, 1 teaspoon **lemon juice**, and 2 tablespoons **butter**.
5. Peel and cut into halves lengthwise 6 **bananas** with all-yellow or green-tipped peel.
6. Arrange halves cut side down in baking dish and brush with about 2 tablespoons melted **butter**.
7. Sprinkle 1/2 teaspoon **salt** over bananas and then pour the orange sauce over bananas.
8. Bake at 375° F for 10 to 20 minutes.

C.14.4 Delightful banana cheesecake

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

Thanks to Ralston Purina Company.

C.14.5 Chicken salad with banana mayonnaise and grapes

1. Place 3 medium **bananas** cut in chunks, 2 teaspoons chopped **garlic**, 3/4 cup non-fat **plain yogurt**, 1 tablespoon **honey**, 2 teaspoons **lemon juice**, and 1/4 teaspoon **salt** in a blender or food processor. Blend until creamy.
2. Arrange 12 cups mixed **lettuces** on six plates.
3. Toss 6 **chicken breasts** cooked and cubed with banana mayo; divide onto salads.
4. Sprinkle with 2 bunchs (≈ 48) halved **grapes** and 1/2 cup **walnut** or **pecan** halves.

Thanks to Chiquita Bananas. See <http://www.jaetzel.de/tim/chiquit.htm>.

D Hints for Reducing High-Frequency VLA Data in *AIPS*

High-frequency data (22 or 43 GHz) from the [VLA](#) may be reduced occasionally with the standard centimeter-wavelength recipe given in this *CookBook*, particularly in the smaller arrays. However, quite frequently, the standard recipe will be inadequate for such data, particularly in the larger (A and B) array configurations. Nevertheless, [VLA](#) data taken at these high-frequencies in the largest array configurations can be calibrated in almost all cases with only a few minor adjustments to the centimeter wavelength recipe.

One reason for more complicated calibration is the high resolution, which resolves the standard flux density calibrators, particularly 3C48. However, most of the problems are caused by the atmosphere, where the troposphere introduces rapid phase fluctuations between the antenna elements of the interferometer. Both effects scale with baseline length expressed in units of wavelength, but the latter also heavily depends on the current weather; phases are sometimes observed to wind on time scales of less than a minute. This causes decorrelation during your calibrator and target source scans, and requires you to determine phase-only calibration, before the flux density (*i.e.*, gain) calibration should be attempted.

In this appendix, an approach to reducing high-frequency [VLA](#) data in *AIPS* is described which should help to overcome the most common problems. It is assumed that the reader has some experience with reducing data in *AIPS*, and is familiar with the “standard recipe” (*e.g.*, Appendix A), tools to examine the data, to apply self-cal, and if appropriate, to deal with spectral-line and polarization calibration issues. If not, you should read the *AIPS CookBook* first (in particular Chapter 4).

High-frequency calibration begins when loading the data, requiring specific parameters in [FILLM](#) to be set (ideally one should use these [FILLM](#) inputs for all frequencies).

Run [FILLM](#) with:

- | | |
|--|--|
| > DOWEIGHT 1 C _R | to apply T _{sys} weights for each individual IF and polarization. |
| > DOUVCOMP -1 C _R | to store the data without compression, which discards individual IF weights. |
| > CPARM 0 ; CPARM (8) 0.05 C _R | to use a short time interval in the CL table entries (in min); 0.05min = 3s. |
| > BPARM 0 ; BPARM (10) 0.75 C _R | to apply opacity and gain curve corrections with zenith opacities weighted 75% by the measured weather and 25% by seasonal averages. |

This creates a CL table that can be interpolated over very short intervals, hopefully short enough to cover the atmospheric phase fluctuations accurately. The default CL table interval is 5 minutes, which may be fine for centimeter wavelengths, but is much too long for proper interpolation of high-frequency phases. Also, you have “nominal sensitivity” weights for individual IF/Pol entries, which reflect sensitivity differences between the receivers, IFs, etc. To retain this “nominal sensitivity” weighting you are required to set [DOCALIB=1](#) (actually $0 < \text{DOCALIB} \leq 99$ and a non-negative value for [GAINUSE](#)) in all the calibration tasks during the remainder of the data calibration.

The importance of the CL table interval is illustrated in Figure D.1 and Figure D.2. On the large scale, the phases look beyond redemption. But, on a relatively short time scale, the phases are relatively well behaved and may be calibrated easily.

After loading your data, check your CL table entries, *e.g.*, [LISTR](#) with [OPTYPE](#) ‘[GAIN](#)’, [PRTAB](#) with [DOHMS](#) 1, or [SNPLT](#) on a short (few minutes) time range with [OPTYPE](#) ‘[AMP](#)’. Make sure the entries are at the

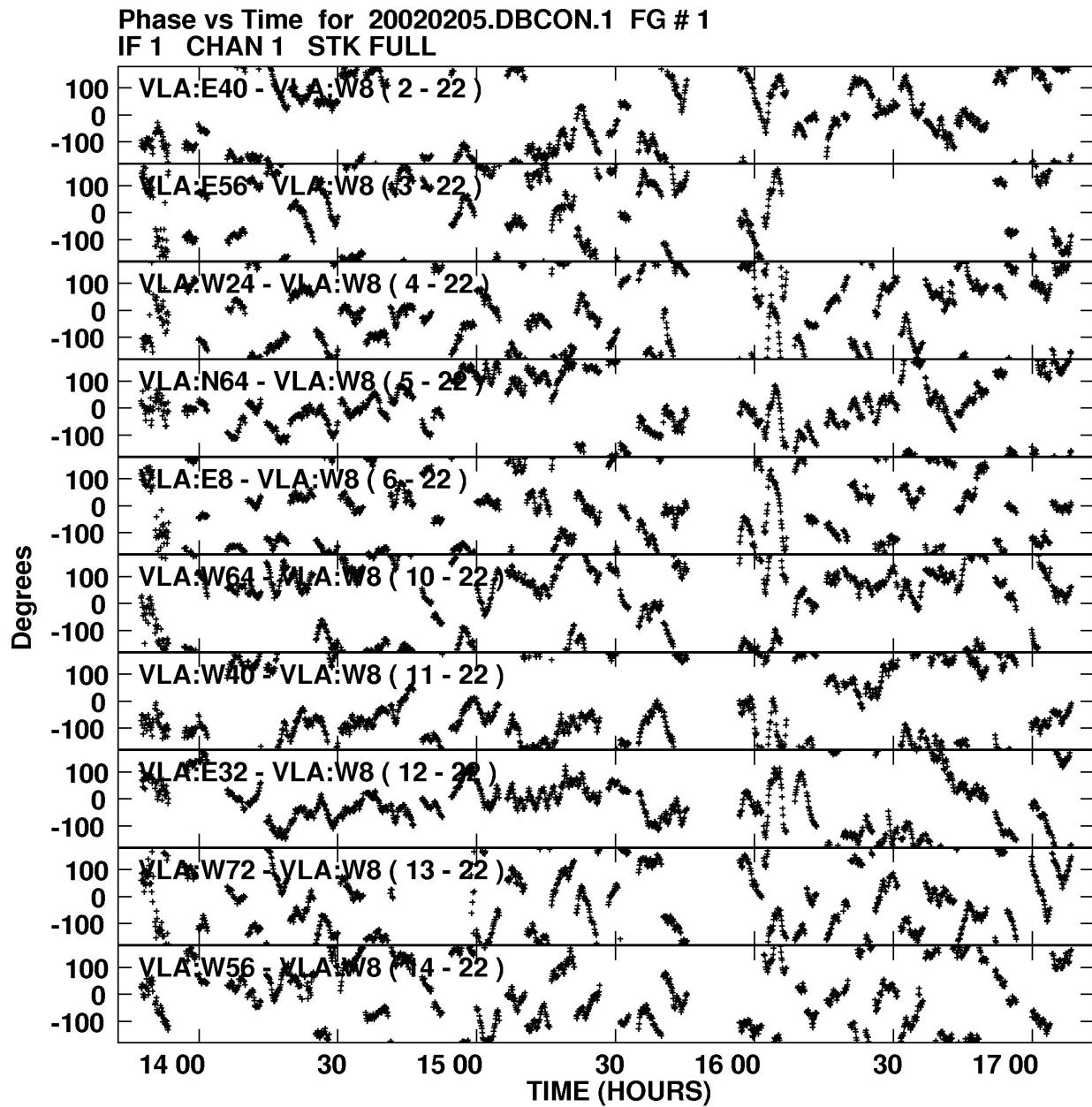


Figure D.1: Uncalibrated VLA A-array 43 GHz phases to the reference antenna in the center of the array (22) over 3 hours on a strong source (frequency switched every 6 minutes) as a function of time plotted by [VPLT](#) look very volatile. Around 16:00 hours they even wrap 360 degrees within one minute. They cannot be calibrated using the default CL table interval of 5 minutes.

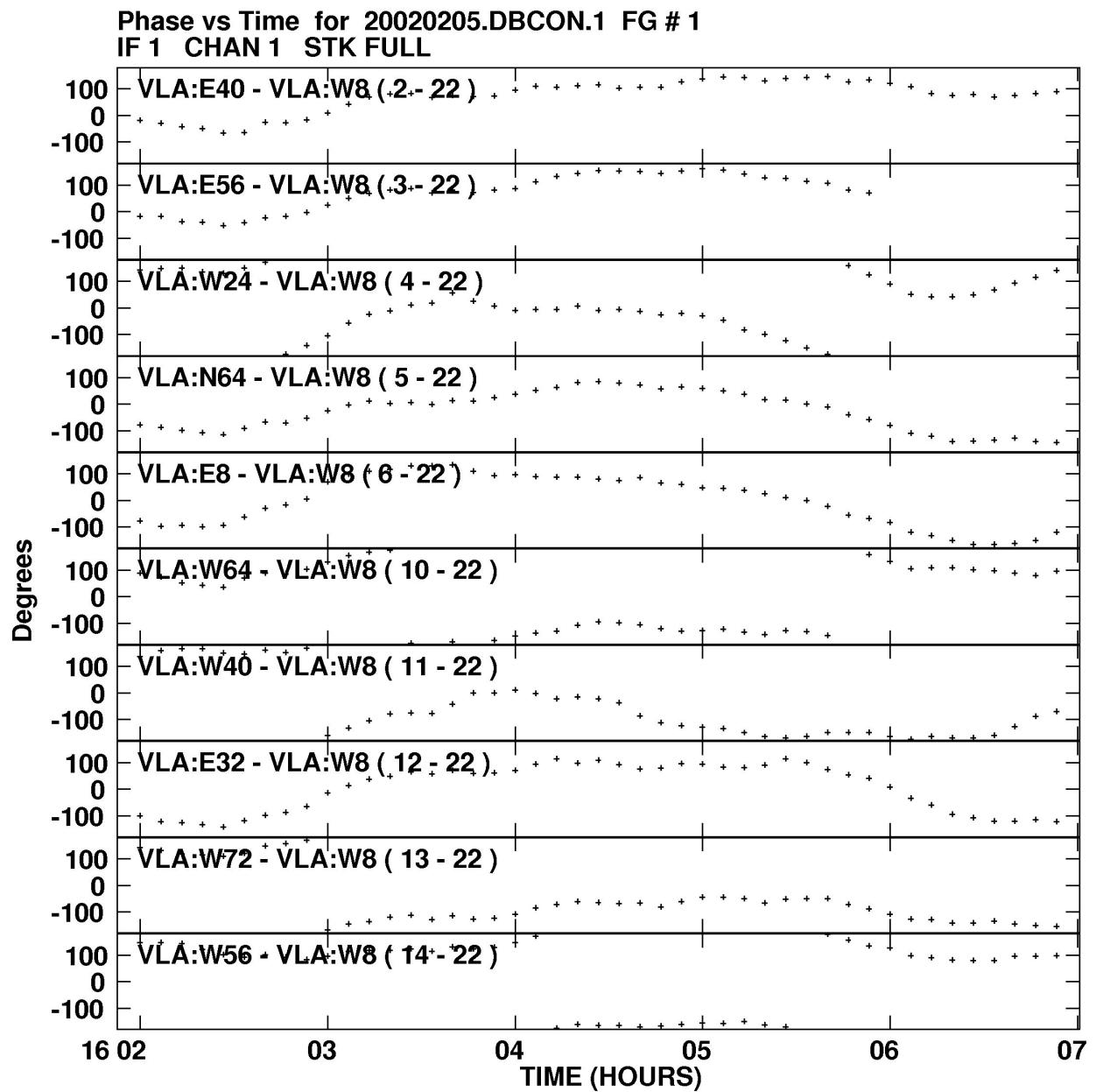


Figure D.2: A blow up from [VPLOT](#) of the time range 16:02 to 16:07 from Figure D.1. The uncalibrated visibility phases are seen to be well behaved, albeit on a short time scale. They can be calibrated if the SN table solutions are found on these typical short time scales and are interpolated with a CL table that has sufficiently short spacing between entries to allow for interpolation of the rapid phase fluctuations. An interval of 20 seconds would be okay here.

interval you expect (much less than a minute) and that the opacity and gain curve corrections have been applied (gains deviating from one by a few percent). Inspect your continuum or “channel 0” data (gains, system temperatures), and flag bad data. For example, you also may wish to flag antenna 1, which is known to have bad optics at 43 GHz, and the antennas without a 43 GHz receiver (currently in December 2002, antennas 9 and 15, but for earlier observations you may want to check the receiver status page (<http://www.vla.nrao.edu/astro/guides/highfreq/>), or your observation log — these antennas may have been left present in your data when you first do a pointing scan in X-band). Standard tools for data inspection and flagging are described in Chapter 4 of the *AIPS CookBook* ([LISTR](#), [UVPRT](#), [VPLT](#), [SNPLT](#), [UVFLG](#), [TVFLG](#), [EDITR](#), [EDITA](#), and many more). Make sure that at least your calibrators are “clean.” Run [VPLT](#) on your calibrators with a reference antenna close to the center of the array (determined by using [PRTAN](#)) to get an indication how rapidly your phases fluctuate; use [ANTENNA reference_antenna 0; SOLINT 0; BPARM 0 2 0](#) (for phase only). If your program source is too weak to allow self-calibration and the phase change from one scan on your calibrator to the next is of the order of 180°, you probably want to flag the source data in between the calibrator scans. Task [SNFLG](#) may be useful for doing this flagging.

Note that fast-switching, when used, will have changed the source names you used in making the observe schedule file. Your sources will have been renamed to their J2000 positions, making it difficult to recognize the calibrator and target scans when you run [LISTR \(OPTYPE 'SCAN'\)](#).

Run [SETJY](#) on your absolute flux density calibrator: 3C286 = J1331+305 = B1328+307, or 3C48 = J0137+331 = B0134+329. And maybe it is a good idea to make a copy of your correct CL table number one (actually all tables) with [TASAV](#) before continuing, so in case of accidents, you have CL table one with the opacity/gain corrections applied. ([INDXR](#) may be used to re-create a CL table, including the opacity and gain corrections made by [FILLM](#).

Run [VLANT](#) to correct phases for improved estimates of the antenna positions. Note that this task requires your computer to be connected to the Internet if your data are recent (within past 18 months or so). Otherwise, apply baseline corrections following advice at www.vla.nrao.edu/astro/archive/baselines/.

Run [CALIB](#), at this stage to correct for phase only, with a small solution interval (depending on your signal to noise, e.g., 20 seconds) on all your calibrator sources. You should use the Clean-components models for 3C286 or 3C48 provided with *AIPS*. See § 4.3.3.1 for information on [CALDIR](#) and [CALRD](#). Then run [CALIB](#) on these sources separately using the appropriate model. There are also models for 3C147 and 3C138.

Inputs to the first pass of [CALIB](#):

- | | |
|---|---|
| > CALSOR 'cal1', 'cal2', ... C_R | to define your calibrators; all but those for which you plan to use a model, e.g., 3C48. |
| > DOCALIB 1 C_R | to apply nominal sensitivities, essential that $0 < DOCALIB \leq 99$. |
| > GAINUSE 0 C_R | apply latest CL table (is version 2 after VLANT). |
| > REFANT <i>reference_antenna</i> C_R | to pick a well behaved antenna in the array center. |
| > SOLINT 20/60 C_R | to solve every 20 seconds; may have to try some values. |
| > SOLMODE 'P' C_R | to do phase calibration only at this stage. |
| > SNVER 1 C_R | to collect all solutions in SN table one. |

And, if you have 3C48, 3C138, 3C147, and/or 3C286 as absolute flux density calibrator(s), you should re-run [CALIB](#), one calibrator source at a time, with the previous/above values plus:

- | | |
|--|--|
| > CALSOR 'ssss', '' C_R | to specify the name you have used for the calibrator source. |
| > IN2DISK <i>d2</i> C_R | to specify the disk with the source model. |
| > GET2NAME <i>ctn2</i> C_R | to specify the CC model to be used by its catalog number. |
| > INVERS 0 C_R | to use the model's latest CC-version (<i>i.e.</i> , one). |
| > NCOMP 0 C_R | to use all the CC-components of the model. |

This may work, but there is no guarantee. Some tricks to apply, in no particular order, in your data set or `CALIB` to obtain a larger relative portion of good versus bad solutions would be:

- Flag some more bad data points on your calibrator sources.
- Discard antennas with uncertain baseline positions (see observing log file).
- Choose a different reference antenna (the one you have might be misbehaving).
- Decrease the `UVRANGE` to weight short baselines (centrally located antennas) more in the solution.
- Use `SOLTYPE 'L1'` to be less sensitive to outlying points.
- Use `FRING` instead of `CALIB` with a larger `SOLINT` to solve for the phase rates, switching off the delay search with `DPARM(2) = -1`.
- Increase or decrease `SOLINT`; increase for weak, decrease for strong sources.
- Decrease the SNR cutoff `APARM(7)` (default 5) to include more noisy but possibly valid solutions.
- Decrease the number of antennas required for a solution (`APARM(1)`, default 6) to require fewer antennas
- Recreate 'CH 0' from 'LINE' to get up to 25% more bandwidth on calibrators.

Note that at 43 GHz in A-array the unprojected *uv*-distance between the outer two antennas on one arm is 0.5 Mega-wavelengths, and the outer 6 antennas — the default for `APARM(1)` — require good solutions out to 2 Mega-wavelengths for `CALIB` to accept the solution for your outermost antenna. Hence, it is a good idea to set `APARM(1)` to *e.g.*, four (or three, if you're willing to check the output `SN` table carefully).

Check the resulting `SN` table number one with `LISTR (OPTYPE 'GAIN', DPARM 1 0)` or `SNPLT (INEXT 'SN', OPTYPE 'PHAS')`, and judge whether you have enough solutions and whether you believe the phases shown are likely to reflect the variation caused by the troposphere. If not, fiddle around with your data and/or parameters in `CALIB` as suggested above and try again. In case the majority of solutions are fine, you may want to edit spurious points in your `SN` table with *e.g.*, `SNEDT`, `SNCOR`, or `SNSMO`.

Once you are satisfied with the phases in your `SN` table, you want to apply phase corrections to minimize decorrelation in your calibrator scans before you determine the absolute flux density scale. To insert the corrections, run `CLCAL` with:

- | | |
|--|--|
| > <code>SOURCES '' C_R</code> | to correct phases for all sources. |
| > <code>CALSOUR 'cal1', 'cal2', ... C_R</code> | to include all your calibrators. |
| > <code>INTERPOL '2PT' C_R</code> | to interpolate between solutions ('SIMP' will average phases over a scan). |
| > <code>SNVER 1; GAINVER 1; GAINUSE 2 C_R</code> | to apply <code>SN#1</code> to <code>CL#1</code> , creating <code>CL#2</code> . |
| > <code>REFANT reference_antenna C_R</code> | to select the same antenna as used in <code>CALIB</code> . |

In less straightforward observations you may not be able to run `CLCAL` only once, *e.g.*, when you are switching frequencies. If in doubt, consult Chapter 4 of the *AIPS CookBook*. It is however very simple to run `CLCAL` multiple times. Inspect your new `CL` table two (three after `VLANT`) for unexpected dubious interpolations and extrapolations (`LISTR` with `OPTYPE 'GAIN'`, `DPARM 1 0`, or `SNPLT` with `INEXT 'CL'`, `OPTYPE 'PHAS'`) and backtrack possible problems.

Now re-run `CALIB` with the corrected phases to obtain the flux density scale. Begin with those sources not requiring models:

- | | |
|--|---|
| > <code>CALSOUR 'cal1', 'cal2', ... C_R</code> | to identify your calibrators; all, except 3C48 and 3C286. |
|--|---|

- > **DOCALIB** 1 C_R to apply antenna gain-opacity and antenna location corrections to the data and theirweights.
- > **GAINUSE** 0 C_R to apply latest CL table (is 3 here).
- > **REFANT** *reference_antenna* C_R to pick the same antenna as used before.
- > **SOLINT** 0 C_R to average over the full scan; remember that phase variations are corrected by **DOCALIB**.
- > **SOLMODE** 'A&P' C_R to do full calibration to get flux densities and residual phases.
- > **SNVER** 2 C_R to collect solutions in a new SN table (two).

Then run **CALIB** again for the absolute flux density calibrator 3C286, or 3C48, using a model and the previous/above values plus:

- > **CALSOUP** 'sssss', '' C_R to specify the name you have for the source.
- > **UVRANGE** 0 C_R to use the full uv range without restrictions.
- > **IN2DISK** *d2* C_R to specify the disk with the source model.
- > **GET2NAME** *ctm2* C_R to specify the model to be used by its catalog number.
- > **INVERS** 0 C_R to use the model's latest CC-version (*i.e.*, one).
- > **NCOMP** 0 C_R to use all the CC-components of the model.

The same tricks may be used as for phase-only to improve the ratio of good to bad solutions. Check your SN table 2 thoroughly; the phases **must** be zero or very close to zero (therefore **INTERPOL** '2PT' is preferred over **INTERPOL** 'SIMP' in **CLCAL**), and you want to make sure the gains of your reference antenna do not scatter too much for individual sources. Before **GETJY** the flux density scale is not fixed so the average gain will depend on source. **GETJY** corrects this so that the gains for each antenna should be similar for all sources. If you can identify misbehaving antennas, flag them, delete SN table 2 (as it does not overwrite values for which data have been deleted), and re-run **CALIB** as many times as needed to re-create SN table two. Cautious users will start from the beginning.

Run **GETJY** to obtain the secondary calibrator flux densities:

- > **SOURCES** 'cal1', 'cal2', ... C_R to specify the unknown sources; *not* the primary calibrator(s).
- > **CALSOUP** '3C48', '' C_R to specify the source name(s) you have used in **SETJY**.
- > **SNVER** 2 C_R to point to the flux density/gain solution table.

If you used a model for the flux density calibrator, your flux scale is tied to the flux density of that calibrator in the SU table. The model Clean components are scaled to match that flux.

Carefully note the flux densities reported by **GETJY** and do not trust these values blindly. **LISTR** or **SNPLT** may point out problematic antenna solutions, requiring you to flag some more data and start over. If you flag data, it is best to delete SN table #2. Solutions at the times of deleted data will not be overwritten. It is helpful to know what flux you expect for your secondary calibrators. See the full source list at aips2.nrao.edu/vla/calflux.html. It is particularly helpful to use one or more of the sources regularly monitored by NRAO staff; see <http://www.aoc.nrao.edu/~smyers/calibration/> for total flux as well as polarization information. You want to check the values, because sometimes the flux densities deviate considerably from the expected values and make no sense. This could be the case if the pointing solutions that were determined prior to your primary calibrator scan are inappropriate for this particular primary calibrator scan *e.g.*, when it is windy or when the cloud cover on your single primary calibrator scan differs from the cloud cover on the secondary calibrators, or, even worse, when these combine. In some cases you may be forced to approximate the flux density scale by entering a (recent) flux density for one of your secondary calibrators, ignoring the primary calibrator scan and accepting an introduced flux density uncertainty. If you decide you have to restart, do not forget to delete your SN and CL tables (except for CL table 1 and 2 if you used **VLANT**) and to reset the flux densities of all your calibrators with **SETJY** (**OPTYPE** 'REJY'), before entering a **ZEROSP** for a new flux density calibrator source (also with **SETJY**). Re-iterate until you are happy with the flux-density scale.

The final flux density calibration table is obtained by running [CLCAL](#) again:

- > [SOURCES](#) '' C_R to calibrate flux densities for all sources.
- > [CALSOUR](#) 'cal1', 'cal2', ... C_R to include calibrators to use for your targets.
- > [INTERPOL](#) '2PT', or 'SIMP' C_R (to specify the interpolation method: no real difference for per-scan solutions).
- > [SNVER](#) 2; [GAINVER](#) 2; [GAINUSE](#) 3 C_R to apply SN#2 to CL#2, creating CL#3.
- > [REFANT](#) reference_antenna C_R to select the same antenna as used in [CALIB](#).

From here you are almost ready to follow the usual “standard recipe,” *i.e.*, polarization and bandpass calibration if appropriate, and splitting into single-source data sets. However, remember to set [DOCALIB](#) = 1 in all these tasks as long as you are working on the multi-source data set and haven’t applied initial phase, flux density (including polarization, bandpass) and “nominal sensitivity” calibration with [SPLIT](#). After [SPLIT](#), the individual weights will have been entered in the data, properly scaled by the latest CL table you’ve made. Using your single source calibrated data, set [DOCALIB](#) = -1 in your subsequent imaging and analysis tasks, unless you do self-calibration.

If you anticipated checking your fast-switching calibration by including a “check source” (a moderately strong source observed a few times with the same fast-switching parameters at about the same distance from your fast-switching source as your target source, but not necessarily in the same direction), you can now assess a snapshot of your calibration by imaging this source. If the fast-switching has worked perfectly, your check source has the expected morphology, flux density, and position. Any position error on the check source should indicate the accuracy of the astrometry on your target source. If you did not include a check source, all but the astrometry and spatial dependence of the calibration can be inferred from your fast-switching source by imaging a scan (use a modified SN/CL table by skipping the calibration on this scan) with the calibration derived from the two neighboring scans.

D.1 Additional recipes

D.1.1 Banana bran muffins

1. Preheat oven to 400° F.
2. Grease 12 2.75-inch muffin cups.
3. In bowl, combine 1/2 cup crushed **cereal** (1.5 cups un-crushed Multi-Bran Chex recommended), 1.5 cups all-purpose **flour**, 1/2 cup **sugar**, 1/3 cup chopped **nuts** (optional), 2.5 teaspoons **baking powder**, and 1/2 teaspoon **baking soda**.
4. In a separate bowl, combine 3 large mashed **bananas** (1.5 cups), 1 **egg** slightly beaten, 1/4 cup vegetable **oil**, 2 tablespoons **water**, and 1 teaspoon **vanilla extract**.
5. Add to cereal mixture and stir just until moistened. Do not over-mix.
6. Divide evenly among muffin cups.
7. Bake 18–20 minutes, or until tester inserted in center comes out clean.

Thanks to Ralston Purina Company.

D.1.2 Banana-pineapple rum bread

1. Place 1/2 cup **white rum** and 1/2 cup diced **dried pineapple** in a bowl, cover, and let sit for at least one hour.
2. In a mixing bowl, beat together 4 tablespoon **butter** or margarine and 3/4 cup **sugar**. Add 1 extra large **egg** and continue beating until light and fluffy.
3. Add 2 large mashed ripe **bananas** and mix well. Beat in 1/3 cup plain **yogurt** — curdling of the mixture is normal.
4. In another mixing bowl, combine 2 cups **all-purpose flour**, 1/2 tablespoon **baking soda**, 1 teaspoon ground **cinnamon**, 1 teaspoon ground **nutmeg**, 1 teaspoon ground **allspice**, and 1/2 teaspoon **salt**.
5. Add the wet ingredients and mix until well blended. Drain the pineapple and add. Fold in 1/2 cup coarsely chopped **pecans**.
6. Pour into liberally greased 9-inch loaf pan. Bake at 350° F for 45 to 55 minutes or until the bread passes the toothpick test. Remove the pan from the oven and let it sit for 10 minutes, before turning out on a rack to cool.

Thanks to Tim D. Culey, Baton Rouge, La. (tsculey@bigfoot.com).

D.1.3 Dulce Zacatecano

1. Peel 3 large not-too-ripe **bananas** and slice lengthwise. Saute in 5 tablespoons **butter** until golden brown. Drain on paper, place in a shallow baking dish, and sprinkle with a little **sugar**.
2. Whip 1/2 cup **heavy sweet cream**. Add 1/4 cup **sugar**, 1/4 cup **dry sherry wine**, and 1 teaspoon **vanilla**. Pour over bananas covering them completely. Chill and serve very cold.

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

D.1.4 Virginia's instant banana pie

1. Mix 1 cup **sour cream**, 1 cup **milk**, and 1 small package **instant vanilla pudding** until mixture thickens.
2. Slice 3 medium **bananas** into the bottom of a 9-inch **graham cracker pie crust**.
3. Pour the pudding over the bananas and refrigerate at least 2 hours.

D.1.5 Chocolate chip banana bread

1. Blend 2 cups mashed **bananas**, 1 tablespoon grated **orange peel**, and 1/3 cup **orange juice** in a bowl. Beat in 3 **eggs**. Stir in 1 cup packed **brown sugar** and 1/3 cup **vegetable oil**.
2. Combine 2-1/2 cups **all-purpose flour**, 1 cup **chocolate chips** 2 teaspoons **baking powder**, 1/2 teaspoon **baking soda**, 1/2 teaspoon **salt**, and 1/2 teaspoon **nutmeg**.
3. Stir dry ingredients into banana mixture just until blended. Pour into 4 greased 5-3/4 x 3-1/4-inch loaf pans.
4. Bake in 350° F oven for 45 to 55 minutes or until tester inserted comes out clean. Let cool in pans on rack for 10 minutes. Remove from pan and let cool completely on rack.

Thanks to Tim D. Culey, Baton Rouge, La. (tsculey@bigfoot.com).

E Special Considerations for EVLA Data Calibration and Imaging in *AIPS*

The old [VLA](#) with its once state of the art, but now dated, correlator and electronics has been turned off. The new electronics and correlator of the [EVLA](#)¹ have been turned on and made available to users. Observing is now on the customary terms although a “Resident shared risk” program is still available for certain esoteric, not fully commissioned, capabilities. Data may have many thousands of spectral channels and up to 8 GHz of bandwidth per polarization. *AIPS* software is capable of handling these data.

The [EVLA](#) has already produced amazing scientific results although the large data volume has created difficulties. Delays are sometimes not set accurately, causing the data analysis to begin with the “VLBI” task [FRING](#) needed to correct large slopes in phase across the bandpass. The flagging information known to the on-line system (telescope off source and the like) is now available in the data format and so can reach *AIPS* either as an initial flag table or as already flagged data. Substantial flagging effort may therefore still be required, particularly to deal with [RFI](#) which inevitably occurs in the wide bandwidths. System temperatures and gains are now transferred, but should be applied with caution to scale the visibilities and to compute data weights. The data weights without this adjustment reflect only integration time. The weather table is available with the data so that reasonable opacities may be determined. However, the “over-the-top”, table which is used in determining antenna positions, the frequency offset table, used in managing Doppler tracking, and the [CQ](#) table, used to correct amplitudes for spectral averaging in the presence of non-zero delays, are not available if the data come to *AIPS* via CASA. The [OT](#) table is now provided when the data are read using the [OBIT](#) package, but astrometric data used to compute accurate projected spacings are just now becoming available.

The following guide will not assume that all issues have been fixed. Steps that can be omitted or simplified when they are will be described. This appendix is written with the assumption that the reader is moderately familiar with *AIPS* as described in the preceding chapters. It is also written with the assumption that you are using current versions of the 31DEC11 or later releases of the software.

E.1 Getting your data into *AIPS*

Your [EVLA](#) data are stored as an “ALMA Science Data Model” ([ASDM](#)) format file in “SDMBDF” (Science Data Model Binary Data Format) in the NRAO archive. They may be read out of the archive in that format, a CASA measurement set format, or in an *AIPS*-friendly uvfits format. This last is produced by the CASA uvfits writing software. Go the the web page

<http://archive.cv.nrao.edu/>

and select the Advanced Query Tool. Fill out enough of the form to describe your data and submit the query. If the data are not yet public, you will need the Locked Project Access Key which may be obtained from the NRAO data analysts. To avoid the need for this key, you may log in to [my.nrao.edu](#) after which it will know if you are entitled to access particular locked projects. The query will return a list of the data sets which meet your specifications. (*Users logged in to an NRAO Socorro computer should use a variant of these instructions; see below.*) On this form, enter your e-mail address, choose AIPS Friendly names (almost certainly does not work), AIPS FITS under the [EVLA-WIDAR](#) section and choose the desired time averaging. If the delays are not accurately known, spectral averaging can be damaging to the data amplitudes. However, the data are often recorded at one-second intervals which is rather short, making the data voluminous. Judicious averaging can help with data set size and processing times without compromising the science. Choose the

¹Now officially called the Karl G. Jansky Very Large Array, but called the [EVLA](#) here.

data set(s) you wish to receive and submit the request. You will be told an estimate of the output data set sizes and the amount of time you will need to wait for the format translation to occur. A 19 Gbyte SDMBDF file run as a test with no averaging was estimated to produce a UVFITS file of 30.26 Gbytes and to take 103 minutes to prepare for download. That time assumes that your download job is the only one being performed. If your download fails, you will probably be told erroneously by e-mail that it worked. The output file will however be missing or incomplete. Try again before contacting NRAO for help.

If you are logged into an NRAO Socorro computer — and perhaps if you are not, — you may find a better route to acquire your data with useful additional information not available via CASA. Try requesting your data in SDMBDF format instead of uvfits and also uncheck the “Create tar file” box *only if you are on a Socorro computer*. (Remote users will need to have the archive load the SDMBDF file into “tar” form in the public ftp area for copying to their home machine.) If you have enough disk space, direct the data to a directory on your machine, *after you have made that directory world writable* (`chmod 777directory-name`). The downloading of the SDMBDF file is quite efficient compared to having the archive computer do all the file translation. When you are told that the SDMBDF file is ready, you may run AIPS and load the data via the verbs **BDFLIST** and **BDF2AIPS**. These verbs run programs in the **OBIT** software package to load your data directly into *AIPS* including flag (FG), index (NX), calibration (CL), over-the-top (OT), SysPower (SY), and CalDevice (CD) tables which you will not get from CASA. If you use this approach, you may skip the **UVLOD** and **INDXR** steps described below. Note that these verbs require that **OBIT** be installed on your computer — as it is in Socorro — and that **ObitTalk** be in your \$PATH. **OBIT** is relatively easy to install and may be obtained from `bcotton@nrao.edu`.

Unlocked files will be downloaded to the NRAO public ftp site

`ftp://ftp.aoc.nrao.edu/e2earchive/`

by default and you may then use **ftp** to copy the file to your computer. Locked files will go to a protected ftp site and you must use **ftp** to download those, even within NRAO. The instructions for downloading will be e-mailed to you. Be sure to specify *binary* for the copy. If you are located in the AOC in Socorro, you may set an environment variable to the archive location, *e.g.*,

`export E2E=/home/ftp/pub/e2earchive CR`

for bash shells

`setenv E2E /home/ftp/pub/e2earchive CR`

for C shells such as tcsh

and simply read unlocked data files directly from the public download area. Note that the file will be deleted automatically after 48 hours in both public and protected data areas.

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

- > **DEFAULT 'BDF2AIPS'; INP C_R** to initialize all relevant adverbs.
- > **DOWAIT 2 ; DOCRT 1 C_R** to wait for the verbs to finish and to display the log file on the terminal after the **OBIT** task finishes. **DOWAIT 1** displays the messages as they are generated but is insensitive to returned error conditions from the **OBIT** tasks. Be sure to set **DOWAIT -1** after using **BDF2AIPS**.
- > **ASDMF(1) = 'path_to_asdm_dir C_R** to set the full path name into the adverb. Note the lack of close quote so that case is preserved. If the name is too long (> 64 characters), put part of the name in **ASDMF(1)** and the rest in **ASDMF(2)**. Trailing blanks in **ASDMF(1)** will be ignored.
- > **BDFLIST C_R** to list the contents of the SDMBDF. Note particularly the “configuration” numbers.
- > **OUTNA 'myname' C_R** to set the *AIPS* name.
- > **OUTCL '' C_R** to take default (UVEVLA) class.
- > **OUTDI 3 C_R** to write the data to disk 3 (one with enough space).

- > **DOUVCOMP FALSE C_R** to write visibilities in uncompressed format. There are no weights at present, so there is no loss of information in compressed format, but the conversion from compressed format costs more than reading the larger data files.
- > **FOR CONFIG = 0:100 ; BDF2AIPS; END C_R** to load all of the configurations in your data, terminating with error messages on the first configuration number not present in your data (when **DOWAIT** is 2).

There are other adverbs — **NCHAN**, **NIF**, **BAND**, and **CALCODE** — available if needed to limit which data are read. **CONFIG** is frequently all that is needed to select data, but these others may be needed if more complicated modes of observing were used.

The uvfits data file may be read from disk into *AIPS* using **UVLOD** or **FITLD**, using:

- > **DEFAULT 'UVLOD'; INP C_R** to initialize and review the inputs needed.
- > **DATAIN 'E2E:filename' C_R** where *filename* is the disk file name in logical area E2E; (see § 3.10.3).
- > **OUTNA 'myname' C_R** to set the *AIPS* name.
- > **OUTCL '' C_R** to take default (UVDATA) class.
- > **OUTSEQ 0 C_R** to take next higher sequence #.
- > **OUTDI 3 C_R** to write the data to disk 3 (one with enough space).
- > **DOUVCOMP FALSE C_R** to write visibilities in uncompressed format. There are no weights at present, so there is no loss of information in compressed format, but the conversion from compressed format costs more than reading the larger data files.
- > **INP C_R** to review the inputs.
- > **GO C_R** to run the program when you're satisfied with inputs.

Watch the messages from **UVLOD** to see where your data set goes and whether the task ran properly. When it is finished, check the output header:

- > **INDI *n*; GETN *m* C_R** to select the data set on disk *n* and catalog number *m*.
- > **IMHEAD C_R** to examine the header.

Note that the header does not show the usual complement of *AIPS* extension files. CASA translates the on-line data into its internal format and then writes the uvfits file read by *AIPS*. Since CASA does not have files comparable to *AIPS* index and CL tables, it does not provide them. To build index and calibration tables, use;

- > **TASK 'INDXR'; INP C_R** to select the task and review its inputs.
- > **INFIL ' ' ; PRTLEV = 0 C_R** to be sure not to use an input text file and to avoid excess messages.
- > **CPARM = 0, 0, 1/2 C_R** to make a CL table 1 with a 30-second interval.
- > **BPARM τ , 0 C_R** to take default EVLA gains and a zenith opacity of τ . Set $\tau = -1$ for no opacity correction. You may set $\tau = 0$, which is now recommended, to get new default opacities. These are based on a detailed model predicting the opacity at any frequency from that at 22 GHz. The combination of weather and seasonal model long used by **FILLM** and **INDXR** is now used solely to estimate the 22 GHz opacity.
- > **GO C_R** to run the task after checking the inputs.

INDXR now uses new EVLA gains information which includes some frequency dependence within the new wide bands. **OBIT**'s application of weather and gains appears to have errors — re-create the CL table 1 and index table with **INDXR** even if you got your data via **OBIT**.

It is a good idea to list the structure of your data set and your antenna locations on the printer and to keep those listings next to your work station for reference:

- > **DEFAULT LISTR ; INP C_R** to initialize the **LISTR** inputs and review them.
- > **INDI n; GETN m C_R** to select the data set on disk *n* and catalog number *m*.
- > **OPTYPE 'SCAN' ; DOCRT -1 C_R** to choose a scan listing on the printer.
- > **GO ; GO PRTAN C_R** to print the scan listing and the antenna file contents.

Read these with care. There have sometimes been problems with antenna identifications, with the order of the IF frequencies, and even with identification of sources by scan. Task **SUFIX** may be used to correct the last problem and, if desired, **FLOPM** may be used to reverse the frequency order. Oddly ordered IFs may require **UVCOP** to split them apart followed by **VBGLU** to paste them back together. You may have to use **SETJY** or **TABPUT** to change the **CALCODE** of some sources if your calibration sources have a blank calibrator code or your target sources have a non-blank calibrator code.

E.2 Initial calibration — VLANT, FRING

As with the **VLA**, NRAO maintains text files describing any changes which are made to our estimate of the antenna locations. Users may wish to apply these changes if their data were taken between the time when antennas were moved to their current stations and the time that the corrections were entered into the on-line control data base. Task **VLANT** works for both **VLA** and **EVLA** data, reading these text files and performing the needed changes to the CL table, writing a new one. Thus

- > **DEFAULT VLANT ; INP C_R** to initialize the **VLANT** inputs and review them.
- > **INDI n; GETN m C_R** to select the data set on disk *n* and catalog number *m*.
- > **GO C_R** to run the task, writing CL table 2.

We have had difficulty setting all of the delays in the **EVLA** to values which are sufficiently accurate. If the delay is not set correctly, the interferometer phase will vary linearly with frequency, potentially wrapping through several turns of phase within a single spectral window ("IF band"). We hope that bad delays will not arise in future, allowing you to skip this section, but use **POSSM** to check for phase slopes. However, even very modest delay errors may matter over the very wide bandwidths of the **EVLA**. Note that **FRING** computes its values around the center of each spectral window but then writes a phase correction appropriate to the frequency reference pixel. Use **IMHEADER** to see what this is in your data set. Traditionally it has been 1.0, but $N_{\text{chan}}/2 + 1$ is a better choice. New task **CENTR**, or numerous other tasks with adverb **FQCENTER**, will convert your data set to this reference. Note that this conversion requires re-scaling the *u, v, w* values and adjusting some FQ table values; it is not just a simple change in the header.

Delay errors are problem familiar to VLBI users and *AIPS* has a well-tested method to correct the problem. Using your **LISTR** output, select a time range of about one minute *toward the end of a scan* on a strong point-source calibrator, usually your bandpass calibrator. Then

- > **DEFAULT FRING ; INP C_R** to initialize the **FRING** inputs and review them.
- > **INDI n; GETN m C_R** to select the data set on disk *n* and catalog number *m*.
- > **SOLINT 1.05 * x C_R** to set the averaging interval in minutes slightly longer than the data interval (*x*) selected.
- > **TIMERANG db , hb , mb , sb , de , he , me , se C_R** to specify the beginning day, hour, minute, and second and ending day, hour, minute, and second (wrt **REFDATE**) of the data to be included. Too much data will cause trouble.
- > **DPARM(9) = 1 C_R** to fit only delay, not rate. *This is very important.*

- > **DPARM(4) = t CR** to help the task out by telling it the integration time t in seconds. Oddities in data sample times may cause **FRING** to get a very wrong integration time otherwise.
- > **INP CR** to check the voluminous inputs.
- > **GO** to run the task, writing SN table 1 with delays for each antenna, IF, and polarization.

The different IFs in current **EVLA** data sets may come from different basebands and therefore have different residual delays. The option **APARM(5) = 3** to force the first $N_{if}/2$ IFs to have one delay solution while the second half of the IFs has another is strongly recommended, but only when the first half all come from one of the “AC” or “BD” basebands (hardware IFs) and the second half come from the other. The 3-bit data path of the **EVLA** actually has four hardware IFs, so **APARM(5) = 4** produces four delay solutions, dividing the IFs in quarters. Note that, at low frequencies, the phases may also be affected by dispersion (phase differences proportional to wavelength). **FRING** now offers **APARM(10)** to enable solving for a single delay plus dispersion from the fitted single-IF delays. This SN table will need to be applied to the main CL table created by **INDXR** or **OBIT**.

- > **TASK 'CLCAL' ; INP CR** to look at the necessary inputs.
- > **TIMERANG 0 CR** to reset the time range.
- > **GAINUSE 0 ; GAINVER 0 CR** to select the highest CL table as input and write one higher as output (version 2 and 3, resp. in this case).
- > **SNVER 1 ; INVER 1 CR** to use only the SN table just created.
- > **INP CR** to review the inputs.
- > **GO CR** to make an updated calibration table.

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

E.3 Initial editing

You should use the tools below to flag out obviously bad data. The tasks which automatically flag data for you, however, depend on meaningful amplitudes and flat spectral shapes. Therefore, flagging and calibration are an iterative process. Do the obvious flagging without spending a lot of time on it. Then do an initial calibration of bandpass, amplitude, and phase. Use that calibration to run **RFLAG** and/or other auto-flagging tasks. Then throw away all CL tables following **VLANT** and begin again with **FRING**, **BPASS**, **CALIB**, etc.

There will be data validity information prepared both by the on-line control software and by the WIDAR correlator and most of this information is available as an initial flag table. The tasks above will have applied this table for you by default since **FLAGVER 0**. On-line flags may already have caused data to be flagged within your data set (but *not* deleted) by CASA. Unfortunately, a flag table is present only via the **OBIT** route. **UVFLG** will be needed to add flags for shadowing (**APARM(5)=25** or so) to a flag table. We still need to look at the data to flag out whatever remains of the time off source not flagged using on-line flagging information. There have also been drop outs in which the visibility is pure zero, typically for all channels and IFs and a single integration. The drop outs should now be handled by **OBIT**, **UVLOD**, and **FITLD**. Note, however, that CASA and **FITLD** pass along all data samples, including those that are fully flagged. This makes the data set rather larger than one might wish. Use **UVCOP** (or **TYAPL** — see § E.6) to remove all fully flagged data samples. Before doing this, use **TVFLG** (§ 4.4.3) to look for any more data samples that might need to be flagged fully. Check especially samples at the beginnings and ends of scans. Try

- > **DEFAULT TVFLG ; INP** to reset all adverbs and choose the task.
- > **INDI n ; GETN m CR** to select the data set on disk n and catalog number m .
- > **DOCAL 1 ; DOBAND -1 CR** to apply the delay calibration. If a bandpass has been determined, use **DOBAND 3** or **1** to apply it.

> BCHAN <i>c1</i> ; ECHAN <i>c2</i> C_R	to average across a range of channels — not as flexible as ICHANSEL but probably okay here.
> NCHAV ECHAN-BCHAN+1 C_R	to average all the channels into one number.
> BIF <i>j</i> ; EIF BIF C_R	to edit one IF only, which will suffice for problems that are not IF dependent, such as drop outs, antenna not on source, etc. Choose an IF that is reasonably free of RFI .
> CALCODE <i>**</i> C_R	to do just calibrators for the moment.
> DPARM(6) Δt C_R	to do no time averaging in the work file set Δt to the data interval in seconds.
> GO C_R	to start the task.

The default smoothing time shown in the display will probably be some multiple of Δt . Change the smoothing time to one times the basic interval in order to edit in detail; **TVFLG** will break up your data into overlapping “groups” which you may edit in sequence. Remember to change the initial setup so that the flags apply to all channels and all IFs. See § 4.4.3 for more information. The **EVLA** has shown a tendency to produce periods of data which are too low in amplitude to be normal noise, but which are not zero. Use **TVFIDDLE** or **TVTRANSF** functions to enhance the brightness of the amplitudes to make sure that apparently black regions really are black (flagged already).

We note here that some users feel that the data need to be inspected more carefully than with just an average of most of the channels. **POSSM** (below) may be of use to find **RFI**. Avoiding the worst of that, you may still wish to run **TVFLG** to look at the average of a few channels at a time. Use **NCHAV** and **CHINC** appropriately. Task **SPFLG** (§ 10.2.2) is the ultimate weapon when looking for channel-dependent difficulties and is even capable of normalizing by source flux including spectral index. However, it is onerous when there are many baselines. **FTFLG** is similar to **SPFLG** but displays all baselines together. It would be a quick way to look for wide-spread **RFI**. These more onerous tools should probably not be used at this preliminary stage; use them after some of the auto-flagging tasks have been run.

E.4 Basic calibration

For data taken with the new low-band receivers (frequencies rather less than 1 GHz), spectral windows may cover a frequency range so wide that the sources vary quite a bit across the individual window. Such wide windows may be convenient for taking the data, but later data reduction will be more reliable with narrower spectral windows. The task **MORIF** in 31DEC12 may be used to increase the number of spectral windows by a user-specified factor. To avoid having to use **NOIFS** ahead of **MORIF**, task **REIFS** was written. It is more flexible in placing the new IFs, including allowing them to include data from more than one input IF.

For *both* continuum and line observations, we must begin by determining which spectral channels are reliable and which are affected by the inevitable loss of signal-to-noise at band edges or are degraded by radio-frequency interference (**RFI**). Use **POSSM** to display spectra from the shorter baselines on the TV:

> DEFAULT POSSM ; INP	to set the task name and clear the adverbs.
> INDI <i>n</i> ; GETN <i>m</i> C_R	to select the data set on disk <i>n</i> and catalog number <i>m</i> .
> SOURCE <i>'bandpass.cal'</i> C_R	to select the strong bandpass calibrator.
> DOTV 1 ; NPLOTS 1 C_R	to plot only on the TV, one baseline at a time.
> ANTEN <i>n1, n2, n3, n4</i> C_R	to select the antennas nearest the center of the array or the maintenance areas.
> BASELINE ANTEN C_R	and only them.

> **DOCAL** 1 ; **APARM** 0 C_R to apply the **FRING** solutions and display vector averaged spectra. Scalar averaged spectra will turn up at the edges reflecting the decreased signal to noise in the outer channels.

> **APARM**(9)=1 C_R to plot all IFs in a single plot.

> **GO** C_R to run the task. Make notes of the desirable channels IF by IF.

If there is no **RFI**, then you may be able to use the same channel range for all IFs. If the **RFI** is particularly pernicious, you may have to edit it out of your data before continuing; see § E.5. The first time through this section, you should accept but perhaps try to avoid the worst of the **RFI**. After the detailed editing, that **RFI** should be gone.

POSSM may reveal extensive ringing in your spectra due to narrow **RFI** signals. Try **SMOOTH**=1,0 to apply Hanning smoothing. If this proves beneficial, you should apply this **SMOOTH**, plus the initial flag table and calibration to the data once and for all with **SPLAT**. Using **SMOOTH** in all operations can produce errors in bandpass functions (if you forget it once in a while) and will produce especially strange results when you use the channel-dependent auto-flagging routines such as **RFLAG**.

For polarization calibration, it is assumed that the phase difference between the right and left polarizations of your calibration is stable with time. Thus, if polarization is important, it is *critical* to find a reference antenna with a stable right minus left phase. Use **CALIB** with **SOLMODE** 'P' and as short a time interval as possible on your strongest calibration sources. Use **SNPLT** with **STOKES** 'DIFF' and **OPTYPE** 'PHAS' to look at the right minus left phases in the SN table produced by **CALIB**. Find the one that is the most stable and use that as **REFANT** henceforth. To avoid later confusion, delete the SN used for this determination with **EXTDEST**.

The basic **EVLA** calibration is much like that described in detail in Chapter 4 except that bandpass calibration is now *required* rather than merely *recommended*. Having chosen those channels which may be reliably used to normalize the bandpass functions,

> **DEFAULT BPASS** ; **INP** to reset all adverbs and choose the task.
 > **INDI** n ; **GETN** m C_R to select the data set on disk n and catalog number m .
 > **DOCAL** 1 C_R to apply the delay calibration — *very important*.
 > **CALSOUR** 'bandpass_cal' C_R to select the strong bandpass calibrator.
 > **SOLINT** 0 C_R to find a bandpass solution for each scan on the BP calibrator.
 > **ICHANSEL** $c11, c12, 1, if1, c21, c22, 1, if2, c31, c32, 1, if3, \dots C_R$ to select the range(s) of channels which are reliable for averaging in each IF. Use the central 30% of the channels if your calibrators are all very strong or more like 90% if they are not. Remember these values — you will use them again.
 > **BPASSPRM**(5) 1 ; **BPASSPRM**(10) 3 C_R to normalize the results only after the solution is found using the channels selected by **ICHANSEL**. Use **BPASSP**(5)=-1 if your phases are not stable within each scan.
 > **GO** C_R to make a bandpass (BP) table.

Do not use spectral smoothing at this point unless you want to use the same smoothing forever after. Apply the flag table. A model for the calibrator may be used; see § 4.3.3.1.

BPASS now contains the adverbs **SPECINDEX** and **SPECURVE** through which the spectral index and its curvature (to higher order than is known for any source) may be entered. For the standard amplitude calibrators 3C286, 3C48, 3C147, and 3C138, these parameters are known and will be provided for you by **BPASS**. For other sources, you may provide these parameters, but **BPASS** will fit the fluxes in the SU table for a spectral index (including curvature optionally) if you do not. Note that, if no spectral index correction is applied, the spectral index of the calibration source will be frozen into the target source. Bandwidths on the **EVLA** are wide enough that this is a serious problem. If you do not know the spectral index of your calibration source, **BPASS** itself or the new task **SOUSP** may be used to determine the spectral indices from the SU table. Of course, that means that **GETJY** must already have been run. Since **BPASS** must usually be run before

CALIB and hence **GETJY**, this suggests that one may have to iterate this whole process at least once. **SOUSP** now offers the option of correcting one or more SN tables after it adjusts the source fluxes for the spectral index it determined. This may reduce the need for further iterations.

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

In 31DEC16, there is a new TV graphical editing task called **BPEDT**. It looks a lot like **EDITA**, displaying multiple antennas at a time, but the horizontal axis is spectral channel rather than time and the data displayed are the solutions from the BP table. This task is particularly suited to identifying and flagging residual RFI in the bandpass calibrator scans after the first pass(es) of **RFLAG**. Repeat **BPASS** if you generate any flags with **BPEDT**.

You now need to run **SETJY** with **OPTYPE 'CALC'** and **SOURCES** set to point at your primary flux calibration sources. You should load the models for these sources that apply to your data with **CALRD**; see § 4.3.3.1. Then run **CALIB** with the model once for each primary flux calibrator:

- > **DEFAULT CALIB ; INP** to reset all adverbs and choose the task.
- > **INDI n; GETN m C_R** to select the data set on disk *n* and catalog number *m*.
- > **IN2DI n2; GET2N m2 C_R** to select the model image on disk *n2* and catalog number *m2*.
- > **DOCAL 1 ; DOBAND 3 C_R** to apply the delay and bandpass calibration — very important.
- > **SOLINT 0 ; NMAPS 1 C_R** to compute a solution for each calibration scan and use the source model.
- > **CALSOUR 'flux_cal' C_R** to select the primary flux calibrator by whatever form of its name appears in your **LISTR** output.
- > **ICHANSEL c11,c12,1,if1,c21,c22,1,if2,c31,c32,1,if3,... C_R** to select the range(s) of channels which are reliable for averaging in each IF. These *must* be the same values that you used in **BPASS**.
- > **SNVER 2 C_R** to put all **CALIB** solutions in solution table 2.
- > **GO C_R** to find the complex gains for the flux calibrator.

Read the output closely. If solutions fail, examine your data closely for bad things. The primary flux calibrator should work without failure. After you have done each primary flux calibrator for which you have models, run **CALIB** on the remaining calibration sources:

- > **CALSOUR 'other_cal1', 'other_cal2' C_R** to select the secondary calibrators by whatever names appear in your **LISTR** output.
- > **CLR2NAME ; NMAPS 0 C_R** to do no models.
- > **GO C_R** to find the remaining complex gains.

Again, examine the output messages closely. There may be a few failures but there should not be many in a good data set. The **RUN** file procedure **VLACALIB** (see § 4.3.3.1) may be used but it does not offer the **ICHANSEL** option which may be required by your data. It also does a scalar averaging for the amplitudes. This averaging is now a vector average of the spectral channels followed by a scalar average over time.

Scalar averaging suffers from Ricean bias in the amplitudes and so should be used only when the calibration source is very strong or when the atmospheric phases are very unstable.

At this point it is necessary to calibrate the fluxes of the secondary calibration sources using your SN table:

- > **TASK 'GETJY' ; INP C_R** to set the task name without changing other adverbs.
- > **SOURCE CALSOUR C_R** to select the secondary sources by the list of name you just used.
- > **CALSOUR 'flux_cal' C_R** to select the primary flux calibrator by whatever form of its name appears in your **LISTR** output.
- > **INP C_R** to check the inputs closely; remember to do all times, IFs, etc. with **SNVER 2**.
- > **GO C_R** to adjust the gains in the SN table and the fluxes in the SU (source) table.

Look at the messages with care — the fluxes in the various IFs should be consistent and the error bars should be reasonably small (< 10% at high frequencies, smaller at low frequencies). If not, look at your SN table with **SNPLT** to see if there are bad solutions. If there are, delete SN table 2, do more flagging with **TVFLG** or **SPFLG**, and repeat the process.

Use the interactive TV task **EDITA** to examine the values in your SN table. There may be bad solutions which will require additional flagging of your calibration data. If there is a significant amount of flagging, you should repeat the calibration process to avoid the influence of bad data on the gains and **GETJY** results. Rather than edit each antenna/IF tediously, use **EDITA** or **SNPLT** to determine allowed ranges of amplitude and phase gains and then use **SNFLG** with **OPTYPE 'A&P'** to do the flagging.

Although it may be better to wait until after detailed flagging, you may wish to iterate at this point, determining the spectral indices of your bandpass calibrators with **SOUSP** and re-doing **BPASS**, **CALIB**, and **GETJY**. If the result is a seriously changed spectral index for your secondary sources you may have to iterate further.

If your calibration source consists of a single spectral line, you may use **CALIB** to determine the gains in that IF, limited to the appropriate channels. The new task **SNP2D** will convert this SN table from a valid phase in only one IF into delays and phases in all IFs on the assumption that the observed phase is in fact a delay.

Finally, apply the gain solutions to your calibration table:

- > **DEFAULT CLCAL ; INP C_R** to clear the adverbs.
- > **INDI n; GETN m C_R** to select the data set on disk *n* and catalog number *m*.
- > **CALCODE '*' C_R** to select all calibration sources.
- > **SNVER 2; INVERS SNVER C_R** to select your solution table from **CALIB**. Do *not* include the SN table from **FRING** a second time!
- > **GO C_R** to apply SN table 2 to CL table 3, creating CL table 4.

Check the result using **POSSM** and/or **VPLOT**.

E.5 Detailed flagging

The calibration you have done to this point has been degraded by **RFI** which has not yet been flagged. However, you need to do the above in order to bring all spectral channels and all antennas and sources into the same flux scale. Now automatic tasks may be used — and they are needed for the large volumes of data produced by the **EVLA**.

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

In detail, **RFLAG** is run using

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

This will produce plots and set cutoff levels in adverbs **NOISE** and **SCUTOFF**. Another run, with **DOPLOT** = 0 will apply these cutoffs and create a new flag table. Note that the flux cutoff levels may depend on the source flux, calling for different levels for strong calibrators, weak calibrators, and very weak target sources. Different cutoff levels for **STOKES**='**RRLL**' and **STOKES** = '**RLLR**' may also be needed. A strong, resolved target source may require different levels for different **UVRANGES**. If so, you will need to break up wide bandwidth data into separate files each containing only one IF so that **UVRANGE** is applied properly. **VBGLU** may be used later to put the IFs back together. **RFLAG** is a new task, so experiment a bit. Note that, if you set **DOFLAG**=1, the creation of a new flag table will happen after the plots in the same execution of **RFLAG**. If a channel is found bad at a time in any one polarization, all polarizations are flagged. If you have a significant spectral line signal in your data, use **DCHANSEL** to have the affected channels ignored throughout **RFLAG**.

There are a lot of adverbs to **RFLAG**. **FPARM**(5) allows you to speed up the spectral part of the flagging by testing more than just the central channel in the sliding median filter. **FPARM**(6) allows you to expand all flags to adjacent channels. **FPARM**(7), 8, 11, and 12 control the extending of flags to additional channels, baselines, or antennas if too large a fraction of channels, baselines, or baselines to an antenna are flagged in the basic time and spectral operations. Similar adverbs also occur in the new task **REFLG** whose job it is to compress the enormous flag tables generated by **RFLAG**. **REFLG** does not handle flags generated by **CLIP**, **TVFLG**, and **SPFLG** since they vary with polarization. **REFLG** can extend a flag to all times if too large a fraction of time is flagged for a given channel, baseline, etc. **REFLG** may not reduce your flag table enough, although it is inexpensive to run and so worth the effort. The application of 10 million flag entries to a data set repetitively is rather expensive. Copying the data, applying the flags once and for all, is the best

solution. [UVCOP](#) has been the traditional method to do this. However, [TYAPL](#) which needs to be run next and must make a new copy of the data has been given the option of applying a large flag table to avoid having to copy the data set twice. Task [FGCNT](#) lets you see how much of your data is flagged by any particular flag table.

A new tool which may help identify bad data at this early stage is the task [REWAV](#) described in § E.8. It checks the data for spectral windows with particularly low and particularly high rms levels. It must copy the selected data to a new file, so it is not particularly recommended at this point. Run it with no flagging of the output for bad values of the spectral rms. Then plot the weights with [VPLT](#) or [ANBPL](#) to look for weights that are seriously abnormal (high or low). Those data may need to be flagged. High weights mean that the data are of abnormally low amplitude, whilst low weights mean that the data are very noisy. [REWAV](#) uses robust methods to find the rms and so a few channels of [RFI](#) may not cause very low weights, but lots of [RFI](#) or receiver failures will make the weights abnormally low. [REWAV](#) now displays statistical information to help you assess what weights are “high” and “low”.

[POSSM](#) may be used again to see if serious [RFI](#) remains after [RFLAG](#) and it may be appropriate to run [TVFLG](#) to look at groups of a small number of spectral channels (or even every channel) on your calibration source. Task [FLGIT](#) (§ 8.1) is an older task that attempts to flag [RFI](#) that is both channel- and time-dependent in a non-interactive fashion. [SPFLG](#) (§ 10.2.2) is labor and time intensive but would be the most reliable method to deal with the problem. [FTFLG](#) is like [SPFLG](#) except that all baselines are added together in a single plot (per polarization). It probably is not good for much flagging, but will provide a quick and sensitive way to look for widely distributed [RFI](#). [CLIP](#) will flag particularly high amplitudes and is capable of running in a normalized fashion dividing by source flux including spectral index (and curvature if desired). ([UVPLT](#) also has the [DOSCALE](#) option, letting you see the normalized visibilities prior to flagging with [CLIP](#).) However, [RFLAG](#) should get the high fluxes in most realistic cases and even has a clip option ([FPARM](#)(13)).

The auto-editing task [FLAGR](#) is also of some use here. It averages the spectral channels to get an estimate of the mean and rms and uses those numbers evaluated over time and baseline in a variety of algorithms to further flag the data. [RFI](#) which is rather wide spectrally and long lived may be found in this way.

E.6 Calibration with the SysPower table

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

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

This table is accessible to [AIPS](#) users with a number of tasks. Begin with task [PRTSY](#) to view the table statistically over time on a per IF, per antenna basis or to view scan or source median averages of one or more of the SY table parameters. Then to examine its contents in more detail in various ways, use [SNPLT](#) with [OPTYPES](#) 'PDIF', 'PSUM', 'PGN', 'PON', 'POFF', 'PSYS', 'PDGN', or 'PSGN'. You could use [OPTYPE](#) = 'MULT' to examine more than one of these at one time, comparing any oddities in e.g., Psum and Pdif. Note that PSYS is especially interesting since $P_{sum}/P_{dif}/2 * T_{cal} = T_{sys}$, the system temperature. It should reflect changes in elevation and strength of the observed source, but should be immune to adjustments to the gain of the telescope. It determines data weights in [TYAPL](#) while $\sqrt{P_{dif_i} P_{dif_j}}$ divides into the visibilities. You

may use **EDITA** (§ 4.4.2) to edit your *uv* data on the basis of the contents of the SY table. Editing may be based on Psum, Pdif, Pgain, Tsys, and on the differences between these parameters and a running median of these parameters. One may also edit the SY table itself with **SNEDT**; the same parameters are available. **LISTR** can even display the SY table Psum, Pdif, system temperatures, and gain factors with **OPTYPE** 'GAIN' and **DPARM**(1) set to 17, 18, 15, or 16, respectively.

More importantly, the SY table can be used to do an initial calibration of the visibility data. Use the display programs to decide if your SY table is fine as is or needs editing. The tasks **TYSMO** and **TYAPL** (§ 4.1.1.1) may be used with **EVLA** data having an SY table. **TYSMO** flags SY samples on the basis of Pdif, Psum, Pgain, and Tsys and then smooths Psum, Pdif and Pgain to replace the flagged samples and/or reduce the noise. You may want to do this to remove outlying bad points and to reduce the jitter in these measurements. **TYSMO** even applies a flag table to the SY before its clipping and smoothing operations. Be sure to plot the results to make sure that the task did what you wanted. Then use **TYAPL** to remove a previously applied SY table (if any) and to apply the SY table you have prepared. The result should be data scaled nearly correctly in Jy and weights in $1/\text{Jy}^2$ in all IFs. The **CUTOFF** option allows you to use obviously good values from the SY table while passing the data from antennas with poor SY values along unchanged. If the SY values from some polarizations or IFs are bad due to **RFI** while others are good, you may copy the good values to replace the bad values using task **TYCOP**. The wide-band, 3-bit mode of the **EVLA** has serious non-linearities in the Pdif measurements which are not yet fully understood. For such data, use **OPTYPE='PGN'** to apply only the post-detection gains to the data. This will remove abrupt jumps due to changes in those gains (which will be more common with 3-bit data). Note that **TYSMO** and **TYAPL** also require a table of the Tcal values which **OBIT** provides in an *AIPS* CD table. Amplitude calibrations are not applied to **EVLA** data weights until they have been made meaningful by **TYAPL** or **REWAY**. Set **FLAGVER** in **TYAPL** if you want to apply your flag table once and for all.

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

Now return to § E.4 to repeat the bandpass and continuum calibrations with correctly scaled data with most of the **RFI** removed.

E.7 Polarization calibration — RLDLY, PCAL, and RLDIF

You may skip this section unless you have cross-hand polarization data and wish to make use of them. Although there have been major improvements in *AIPS* polarization routines, they still do not correct parallel hand visibilities for polarization leakage. Thus you need to calibrate polarization only if you wish to make images of target source Q and U Stokes parameters. Polarization calibration is discussed extensively in § 4.6; we will discuss changes made because of the wide bandwidths and other aspects of the **EVLA**.

Frequently, the delay difference between right- and left-hand polarizations must be determined even if **FRING** was not required for the parallel-hand data. Use **POSSM** to plot the RL and LR spectra to see if there are significant slopes in phase. If so, use a calibration source with significant polarization, although the **EVLA** D terms are often large enough to provide a usable signal in the absence of a real polarized signal. Note that 3C286 is significantly polarized and is likely to be the best source to use for this purpose. Then

- > **TASK 'RLDLY' ; INP C_R** to look at the necessary inputs.
- > **BCHAN c₁ ; ECHAN c₂ C_R** to select channels free of edge effects.
- > **DOCAL 1 ; GAINUSE 0 C_R** to apply the **FRING** results and all other current calibrations.

- > **REFANT** n_r C_R to select a reference antenna - only baselines to this antenna are used so select carefully. Alternatively, **REFANT** 0 will loop over all possible (not necessarily good) reference antennas, averaging the result.
- > **DOIFS** j C_R to set the adverb to the value of **APARM(5)** used in **FRING** (§ E.2). The IFs are done independently (≤ 0), all together ($= 1$), in halves ($= 2$), or more generally in N groups ($= N$).
- > **TIMERANG** $db, hb, mb, sb, de, he, me, se$ C_R to specify the beginning day, hour, minute, and second and ending day, hour, minute, and second (wrt **REFDATE**) of the data to be included. Use an interval not unlike the one you used in **FRING**.
- > **INP** C_R to check the inputs.
- > **GO** C_R to produce a new SN table with a suitable left polarization delay.

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

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

- > **DEFAULT PCAL ; INP** to reset all adverbs and choose the task.
- > **INDI** n ; **GETN** m C_R to select the data set on disk n and catalog number m .
- > **DOCAL** 1 ; **DOBAND** 3 C_R to apply the delay, complex gain, and bandpass calibration.
- > **CALSOUR** '*pol.cal1*', '*pol.cal2*' C_R to select the polarization calibrator(s) by whatever form of their names appears in your **LISTR** output. These sources must have I polarization fluxes in the source table.
- > **ICHANSEL** $c11, c12, 1, if1, c21, c22, 1, if2, c31, c32, 1, if3, \dots C_R$ to select the range(s) of channels which are reliable for averaging in each IF. These probably should be the same values that you used in **BPASS**.
- > **DOMODEL** -1; **SPECTRAL** -1 C_R to solve for source polarization in a continuum manner.
- > **PRTLEV** 1 C_R to see the answers and uncertainties on an antenna and IF basis.
- > **CPARM** 0,1 C_R to update the source table with the calibrator source Q and U found.
- > **INP** C_R to review the inputs.
- > **GO** C_R to find the antenna leakage terms and the source Q and U values on an IF-dependent basis.

PCAL will write the antenna leakage terms in the antenna file and the source Q and U terms in the source table (if **CPARM(2) > 0**). **DOMODEL** may be set to true only if the model has $Q = U = 0$ since **PCAL** cannot solve for the right minus left phase difference. If **SOLINT=0**, **PCAL** will break up a single scan into multiple intervals, attempting to get a solution even without a wide range of parallactic angles.

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

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

> INDI <i>n</i> ; GETN <i>m</i> C _R	to select the data set on disk <i>n</i> and catalog number <i>m</i> .
> DOCAL 1 ; DOBAND 3 C _R	to apply the delay, complex gain, and bandpass calibration.
> DOPOL 1 C _R	to apply the polarization calibration.
> BCHAN <i>c</i> ₁ ; ECHAN <i>c</i> ₂ C _R	to average data from channels <i>c</i> ₁ through <i>c</i> ₂ only.
> SOURC 'pol.cal1', 'pol.cal2' C _R	to select the polarization calibrator(s) by whatever form of their names appears in your LISTR output. These sources must have known polarization angles.
> SPECTRAL 0 C _R	to do the correction in continuum mode.
> DOAPPLY 1 C _R	to apply the solutions to a CL table (making a new modified one) and to the AN and SU tables, updating them in place.
> DOPRINT 0 C _R	to omit all the possible printing.
> INP C _R	to review the inputs.
> GO C _R	to determine and apply the corrections.

The **EVLA** polarizers appear to be very stable in time, but to have significant variation with frequency. See Figure E.1. Serious polarimetry with the **EVLA** will *require* solving for the antenna polarization leakage as a function of frequency. To compute a spectral solution, assuming you already did the process in the preceding paragraph:

> TGET PCAL C _R	to retrieve the PCAL adverbs.
> SPECTRAL 1 C _R	to do the channel-dependent mode.
> DOMODEL 0 C _R	to solve for Q and U as a function of frequency. Because PCAL does not solve for a right-left phase difference and that difference is a function of spectral channel, you must solve for a source polarization.
> INTPARM <i>p</i> ₁ , <i>p</i> ₂ , <i>p</i> ₃ C _R	to smooth the data after all calibration has been done while honoring ICHANSEL .
> SPECparm 0 C _R	to determine the calibration source I, Q, and U spectral indices from fluxes in the source table. If you use PMODEL you must provide spectral indices for the model that apply in the frequency range of the data (curvature cannot be specified).
> INP C _R	to review the inputs, the task will take a while to run.
> GO C _R	to run the task writing a PD table of spectral leakages ("D terms") and, if DOMODEL ≤ 0 , a CP table of source Q and U spectra.

If the combination of flagging, **ICHANSEL**, and **INTPARM** results in no solutions for some channels, the solutions from nearby channels will be interpolated or extrapolated so that all channels get solutions.

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

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

> DEFAULT RLDIF ; INP	to reset all adverbs and choose the task.
> INDI <i>n</i> ; GETN <i>m</i> C _R	to select the data set on disk <i>n</i> and catalog number <i>m</i> .
> DOCAL 1 ; DOBAND 3 C _R	to apply the delay, complex gain, and bandpass calibration.
> DOPOL 1 C _R	to apply the polarization calibration, spectral if present.

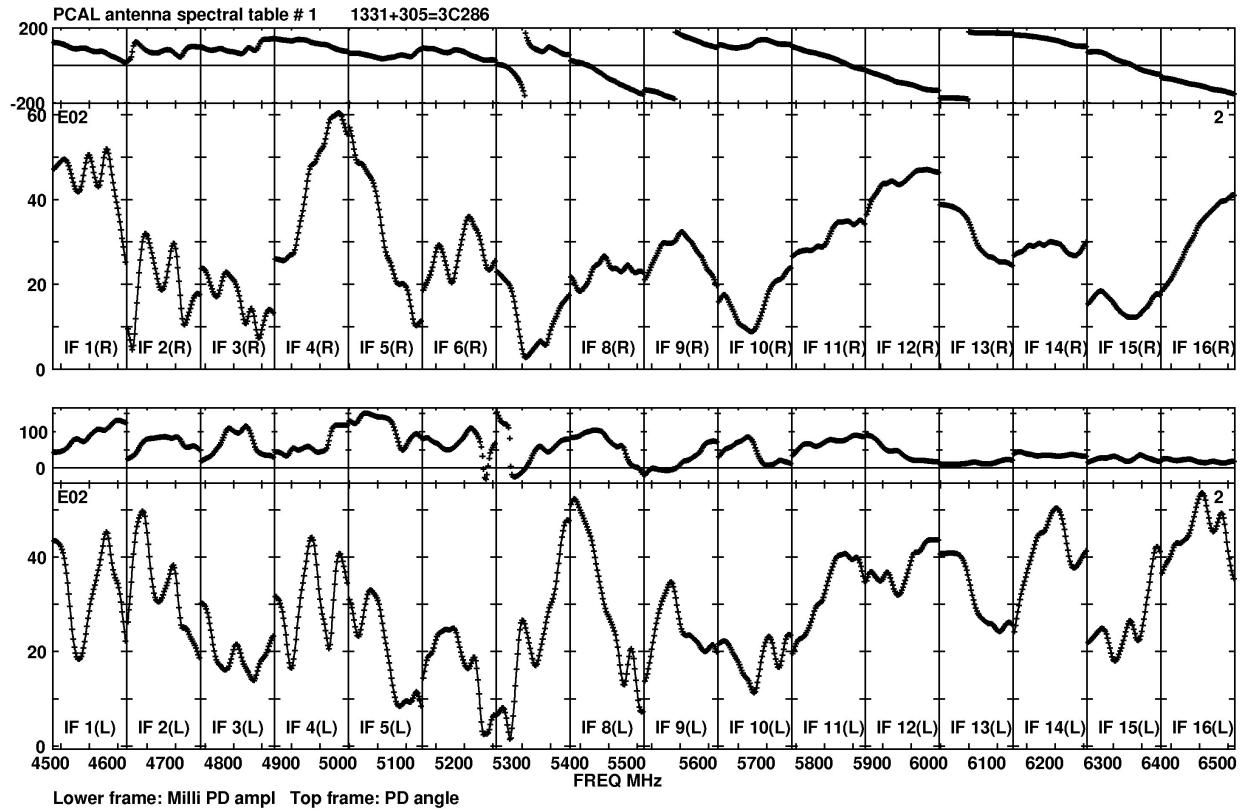


Figure E.1: Example spectrum showing D term solutions for one antenna in right and left polarizations covering about 2 GHz at C band

- > **INTPARM** p_1, p_2, p_3 C_R
- > **BCHAN** c_1 ; **ECHAN** c_2 C_R
- > **SOURC** 'pol_cal1', 'pol_cal2' C_R
- > **POLANGLE** p_1, p_2 C_R
- > **SPECTRAL** 1 C_R
- > **DOAPPLY** 1 C_R
- > **DOPRINT** -1 ; **OUTPRI** 'file_name' C_R
- > **INP** C_R
- > **GO** C_R

- to smooth the data after all calibration has been done.
- to use solutions from channels c_1 through c_2 only, extrapolating solutions to channels outside this range.
- to select the polarization calibrator(s) by whatever form of their names appears in your **LISTR** output. These sources must have known polarization angles.
- to provide the task with the source polarization angle(s) in degrees in source number order. The phase correction will be twice this value minus the observed RL phase. Do not provide values for 3C286, 3C147, 3C48, and 3C138. These are known to **RLDIF** including rotation measures and other spectral dependence.
- to do the correction in spectral mode.
- to apply the solutions to a BP table (making a new modified one) and to the PD and CP tables.
- to write the phase corrections applied to a text file suitable for plotting by **PLOTR**.
- to review the inputs.
- to determine and apply the corrections.

Use **UVPLT**, **LISTR** or **POSSM** to check that the expected RL and LR phases now appear with all calibrations turned on. Following these steps, you apply the polarization calibration in any task offering **DOPOL**. A value of 1 will apply the spectral solution if present or the continuum one is there is no PD table. A value of 6 for **DOPOL** requests the continuum solution despite the presence of a spectral solution. Use **POSSM** to plot the calibration sources in RL, LR, Q, and U polarization to make sure that all has functioned correctly (these are newly revised tasks).

If you do not have good observations of a polarization angle calibrator, it may be possible to correct the data in some circumstances. See the help files for **QUOUT** and **QUFIX** for possible methods.

If you think that the right minus left phase of your reference antenna was not stable over time, but you have produced good model images of Q and U ignoring that, then it may be possible to improve the polarization calibration with the task **RLCAL**. It does a self-calibration of Q and U to find a time-variable right minus left phase to apply to the data.

E.8 Target source data — edit and SPLIT

At this point, your calibration should be finished. You should now do an initial editing on the target sources, much like that done above for the calibration sources. Run **RFLAG** on the target source(s) and perhaps use:

- | | |
|--|---|
| > DEFAULT TVFLG ; INP | to reset all adverbs and choose the task. |
| > INDI <i>n</i>; GETN <i>m</i> C_R | to select the data set on disk <i>n</i> and catalog number <i>m</i> . |
| > DOCAL 1 ; DOBAND 3 C_R | to apply the delay, complex gain, and bandpass calibration. |
| > BCHAN <i>c1</i> ; ECHAN <i>c2</i> C_R | to average across a range of channels — not as flexible as ICHANSEL but probably okay here. |
| > BIF <i>j</i> ; EIF BIF C_R | to edit one IF only, which will suffice for problems that are not IF dependent, such as drop outs, antenna not on source, etc. Choose an IF that is reasonably free of RFI . |
| > NCHAV ECHAN-BCHAN+1 C_R | to average all the channels into one number. |
| > CALCODE '-CAL' C_R | to do just target sources now. |
| > DPARM(6) Δ <i>t</i> C_R | to do no time averaging in the work file set Δ <i>t</i> to the data interval in seconds. |
| > GO C_R | to start the task. |

Again, remember to set it to flag all channels and IFs. You may have to select sub-windows and force the averaging to one times Δ *t* to edit in detail, or perhaps the default time averaging will be beneficial. In general, the **DISPLAY AMP V DIFF** is a powerful way to catch bad amplitudes and phases. It will catch drop outs either as bright lines for strong sources or dark grey ones for weak sources.

Since **EVLA** data sets tend to be large and unwieldy, it is recommended that you separate the data into the separate target sources, applying the current calibration and flagging once and for all. The imaging task **IMAGR** can do this on the fly, but, especially for observations of spectral-line sources, this is excessively expensive.

- | | |
|---|---|
| > DEFAULT SPLIT ; INP | to reset all adverbs and choose the task. |
| > INDI <i>n</i>; GETN <i>m</i> C_R | to select the data set on disk <i>n</i> and catalog number <i>m</i> . |
| > DOCAL 1 ; DOBAND 3 C_R | to apply the delay, complex gain, and bandpass calibration. |
| > CALCODE '-CAL' C_R | to do just target sources now. |
| > GO C_R | to write out separate calibrated data sets for each target source. |

A task new to 31DEC13 will help calibrate data sets which contain a strong, *e.g.*, maser, line in one or a few channels with interesting but weaker signals in other channels and/or the continuum. After applying the best standard calibration described above, you can split out the maser channel(s) with [UVCOP](#). Make images and self-calibrate following the processes described in Chapter 5 making an SN table containing the full calibration needed to apply to the maser. Then use task [SNP2D](#) to convert this into delays and phases to apply to the full, multi-IF data set. This will work so long as the residual phases found in the self-cal are small and is actually required to do the best calibration possible over very wide bandwidths.

Unless [TYAPL](#) has been used, [EVLA](#) data sets have weights which only reflect the integration time in seconds. Calibration routines do not change these weights when changing the data amplitudes. There is a task called [REWAY](#) which computes a robust rms over spectral channels within each IF and polarization. It can simply base the weights on these on a record-by-record, baseline-by-baseline basis. Alternatively, it can use a scrolling buffer in time so that the robust rms includes data for a user-specified number of records surrounding the current one. A third choice is to average the single-time rmses over a time range and then convert them to antenna-based rmses. In all three modes, the task can then smooth the rmses over time applying clipping based on user adverbs and the mean and variance found in the rmses. A flag table (extension file) may be written to the input data file removing those data found to have rmses that are either too high or too low. For these weights to be meaningful, the bandpass and spectral polarization calibration must be applied and it helps to omit any [RFI](#) or other real spectral-line signal channels from the rms computation. For the weights to be correctly calibrated, all amplitude calibration must also be applied. For these reasons, [REWAY](#) might well be used instead of [SPLIT](#) — when [TYAPL](#) was not used — running it one source at a time. Thus,

- | | |
|---|---|
| > DEFAULT REWAY ; INP | to reset all adverbs and choose the task. |
| > INDI n; GETN m C_R | to select the data set on disk <i>n</i> and catalog number <i>m</i> . |
| > DOCAL 1 ; DOBAND 3 C_R | to apply the delay, complex gain, and bandpass calibration. |
| > SOURCE 'target₁', '' C_R | to do one target source. |
| > APARM 11, 30, 12, 0, 10, 4 C_R | to use a rolling buffer of 11 times separated by no more than 30 seconds and then smoothed further with a Gaussian 12 seconds in FWHM. Data are flagged if the rms is more than 4 times the variance away from the mean averaged over all baselines, IFs, and polarizations. Flagging on the variance of the rms from the mean on a baseline basis is essentially turned off by the 10. |
| > GO C_R | to write out a calibrated, weighted data set for the first target source. |

Then, when that finishes

- | | |
|--|------------------------------|
| > SOURCE 'target₂', '' ; GO C_R | to do another target source. |
|--|------------------------------|

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

E.9 Spectral-line imaging hints

Some spectral-line data sets contain only a few spectral windows, each with a great many spectral channels. The continuum subtraction task [UVLSF](#) requires that the continuum signal be a linear function of channel across each spectral window. Converting the data to more spectral windows with task [MORIF](#) or [REIFS](#) may make the data come closer to this requirement. Having more spectral windows with fewer channels may also speed up imaging in the [LINIMAGE](#) procedure mentioned below.

In many spectral-line observations you will want to separate the continuum signal from the channel-dependent signals. This is discussed in some detail in § 8.3. The larger number of channels from the **EVLA** does mean that continuum may be estimated with greater accuracy than when there were rather few channels which were both free of edge effects and spectral-line signal. The wider total bandwidth may, however, invalidate the assumption that the continuum signal at each visibility point can be represented by a polynomial of zero or first order. If there is a single dominant continuum source offset from the phase center, the assumption may be rendered valid by shifting the data with **UVLSF** to center the continuum source temporarily in order to subtract it. To examine this assumption and to determine which channels appear safe to use as “continuum” channels, use **POSSM**.

- > **DEFAULT POSSM ; INP** to set the task name and clear the adverbs.
 - > **INDI Tn ; GETN Tm C_R** to select the calibrated line data set on disk Tn and catalog number Tm .
 - > **DOTV 1 ; NPLOTS 1 C_R** to plot only on the TV, one baseline at a time.
 - > **ANTEN $n1, n2, n3, n4$ C_R** to select the antennas nearest the center of the array and only them.
 - > **BIF j ; EIF BIF C_R** to plot one IF at a time.
 - > **APARM 0 C_R** to display vector averaged spectra. Scalar averaged spectra will turn up at the edges reflecting the decreased signal to noise in the outer channels which will assist in determining channels that should be omitted.
 - > **GO C_R** to run the task. Make notes of the desirable channels IF by IF.
- Note also whether the continuum appears to be a linear function of channel. If so, then use **UVLSF** to fit the continuum signal, writing a continuum only and a spectral-line only data set:
- > **DEFAULT UVLSF ; INP** to set the task name and clear the adverbs.
 - > **INDI Tn ; GETN Tm C_R** to select the calibrated line data set on disk Tn and catalog number Tm .
 - > **ICHANSEL $c11, c12, 1, if1, c21, c22, 1, if2, c31, c32, 1, if3, \dots$ C_R** to select the range(s) of channels which are reliable for fitting the continuum. For a multi-IF data set, you will need to select the channel ranges carefully by IF.
 - > **ORDER 1 C_R** to select fitting the continuum in real and imaginary parts with a first order polynomial in channel number. **UVLSF** offers orders up to four, but they are not for the faint at heart and will give bad results if there are large ranges of channels left out of the fit due to line signals.
 - > **DOOUTPUT 1 C_R** to have the continuum which was fit written as a separate data set. This may be used to image the continuum.
 - > **SHIFT $\Delta x, \Delta y$ C_R** to shift the phase center to the dominant continuum source temporarily for the fitting.
 - > **GO C_R** to run the task.

Imaging the continuum output may, in addition to any scientific value of the continuum image, provide additional flagging and even self-calibration information which may be applied to the line data.

If **UVLSF** cannot be used, flag the channels at the edges and those with spectral signals using **UVFLG**. Construct a continuum image with **IMAGR** on this flagged, spectral-line data set. Note that you might want to reduce the size of the data set with time averaging (**UVAVG**) and/or channel averaging (**SPLIT** or **AVSPC**) before beginning the imaging. Imaging is discussed in detail in § 5.2 through § 5.3.6 and will not be discussed here. You may find that additional editing is needed and that iterative self-calibration is of use. Be sure to copy those flags (but not the edge and spectral-signal flags) and final SN table back to the line data set. Apply them with **SPLIT** and then subtract the final continuum model with **UVSUB**. If you have had

to use the spectral index options of [IMAGR](#), you may do the proper subtraction including these options with [OOSUB](#) rather than [UVSUB](#).

At present, the [EVLA](#) observing setup allows you to select the initial frequency of observation based on a desired LSRK velocity in the central channel. From there, however, the observations are conducted at a fixed frequency. Furthermore, the information about rest frequencies, source velocities, and even more fundamental parameters such as reference frame (LSRK or barycentric) and type of velocity (radio or optical) are lost. [SETJY](#) allows you to correct this. First use [SETJY](#) to set the desired rest frequencies (note that they are allowed to be a function of IF) and the [VELTYP](#) and [VELDEF](#). Then use [OPTYPE='VCAL'](#) over all sources and IFs. This will compute the velocities at which you observed for the first time you observed each source and enter the values in the source table.

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

Spectral-line imaging of [EVLA](#) data will resemble that for the old [VLA](#) except for the increased number of spectral channels and the consequent increase in the data set size. Since [IMAGR](#) must read the full data set to select the data for the next channel to be imaged, it is important that the data set be small enough to fit in computer memory if at all possible. OSRO data sets may not need this operation and skipping it may simplify any continuum imaging that you wish to do. However, separating the IFs into separate files will not interfere with spectral imaging and will help with the data set size problem:

- > [DEFAULT UVCOP ; INP](#) to reset all adverbs and choose the task.
- > [INDI Tn; GETN Tm CR](#) to select the calibrated target data set on disk *Tn* and catalog number *Tm*.
- > [DOWAIT 1 CR](#) to have the task resume AIPS only after it has finished.
- > [OUTSEQ 0 ; OUTDISK INDISK CR](#) to avoid file name issues and select the output disk.
- > [FOR BIF = 1 TO N; EIF = BIF ; GO; END CR](#) to make separate files of each of the *N* IFs.
- > [DOWAIT -1 CR](#) to turn off waiting.

Doing this [UVCOP](#) step on large data sets will be worth any extra trouble it may cause. A [RUN](#) file and procedure called [LINIMAGE](#) has been written to assist in this process. It even runs [FLATN](#) and reassembles the images from the separate IFs into cubes with [MCUBE](#) or [FQUBE](#) if necessary. Note that you could perform the separation into separate IFs before [UVLSF](#) which will speed up [POSSM](#) and [UVLSF](#). However, the continuum output would then have to be assembled using [VBGLU](#), which is why the steps above were shown in the present order.

Spectral-line imaging is discussed in § 8.4 as well as throughout Chapter 5. With large numbers of spectral channels, you may wish to have [IMAGR](#) find appropriate Clean boxes for you. Set [IM2PARM\(1\)](#) through [IM2PARM\(6\)](#) cautiously. [IM2PARM\(7\)](#) controls whether the boxes of channel *n* are passed on to channel *n+1*. The default does not pass the boxes along when auto-boxing which is probably the correct decision. The end result of the imaging will be one image "cube" for each IF since each IF has to be imaged separately even with a multi-IF input data set. (If you set [BIF = 1](#); [EIF = 0](#) and try to image channel 103, you will actually image the average of channel 103 from each of the IFs.) To put the individual cubes together into one large cube, use [MCUBE](#) (§ 8.5.1).

The wide bandwidths of the **EVLA** have revealed an error in the old code. **IMAGR** and **MCUBE** now control the units of data cubes carefully, making sure that each plane is in units of Jy per *header* beam. The actual restoring beamwidths used are now maintained in a CG table. This allows the best resolution to be used in each plane and the approximate match between the units of the residual image and the restored components to be maintained, while still returning correct fluxes when integrating brightness over area. **CONVL** with **OPCODE** = 'GAUS' will now use the CG table to find the exact Gaussian needed to produce a constant resolution in each plane. Use of **BMAJ** = 0 in **IMAGR** followed by **CONVL** is now the best way to insure a constant resolution with correct image units throughout.

E.10 Continuum imaging hints

The first problem that continuum observers will notice with their **EVLA** data is that the spectral and time resolution of the data, by default anyway, will be rather more than their science requires. It will be possible to instruct the software which extracts data from the archive to do some averaging in both frequency and time. However, detailed editing for **RFI** and other issues may require excellent resolution in both these domains. After the data have been edited, you can average data in both domains so long as you are careful not to average so much that you produce radial (bandwidth) and/or transverse (time) smearing within the image area. Note that the increased sensitivity of the **EVLA** will increase the area over which non-negligible astronomical objects may be found while the wide bandwidth will mean that lowest frequency part of your band will be sensitive, because of its larger primary beam, to a much larger area on the sky than the highest frequency part. The spectral averaging can be done with **SPLIT**; use **APARM(1)=1** and set **NCHAV**, **CHINC**, and perhaps **SMOOTH** appropriately. Similarly, **AVSPC** can be used with **AVOPTION='SUBS'**, setting **CHANNEL** and **SMOOTH** suitably. You will almost certainly wish to retain some spectral separation, so do not use the "channel 0" option.

Time averaging should be done with **UVAVG**:

- > **DEFAULT UVAVG ; INP** to reset all adverbs and choose the task.
- > **INDI *Sn*; GETN *Sm* *C_R*** to select the calibrated target data set on disk *Sn* and catalog number *Sm*.
- > **YINC *Δt* *C_R*** to average to *Δt* seconds.
- > **OPCODE 'TIME' *C_R*** to have all samples in a given interval written with the same time; this helps with self-cal. Other options are available.
- > **GO *C_R*** to produce the averaged data set.

UBAVG will do a more aggressive averaging, using baseline-dependent time intervals appropriate for the desired field of view. Do not use **UBAVG** if you are planning to use self-calibration since it destroys the time regularity in the data on which **CALIB** depends. **IMAGR** may now do this extra averaging for you on the fly to reduce the size of the work file it uses. Set **IM2PARM(11)** and (12).

Imaging of the continuum is discussed at great length in Chapter 5 and those details will not be repeated here. Bandwidth-synthesis imaging, which will be the only form of continuum imaging with the **EVLA**, will make certain adverbs more important. Set **BCHAN** and **ECHAN** to avoid the noisier edge channels. Set **NCHAV** = **ECHAN** - **BCHAN** + 1 and **CHINC** = **NCHAV**. This will then image all of your IFs and spectral channels into a single image, positioning each channel correctly in the *uv* plane. With the **EVLA**, you will be imaging a wider field of view than you did with the **VLA**. Use **SETFC** with **IMSIZE 0 ; CELLSIZE 0** to see if you should image with a single facet or with multiple facets. If using multiple facets and trying for significant dynamic range, start imaging with **OVERLAP 2 ; ONEBEAM -1**, but consider **OVRSWTCH** = -0.05 or so to switch into faster methods of Cleaning when the dynamic range in the residual is small enough.

It has been widely noted that the noise in the outer channels of each IF (spectral window) is higher than in the more central channels. You may wish to experiment with down-weighting channels by a function of

the bandpass correction amplitude. In 31DEC13, task [BPWGT](#) offers several options for doing just this, using [WEIGHTIT](#) to control what exponent of the bandpass amplitude is used in the correction. A solution, which should be better, is available in 31DEC14 with task [BPWAY](#). This task operates on either a scan-by-scan or a source-by-source basis to evaluate the rms as a function of time on each channel individually. It determines the rms on short intervals, like [RFLAG](#), normalizes them over each spectral window, and then smooths the normalized rmses over time functions which can be quite long. The results are then applied to the data to modify the weights on a channel-by-channel, baseline-by-baseline basis. This process should allow data from the edges of the spectral windows to be used appropriately, especially in bandwidth synthesis. Note however, that channel 1 in each window includes signal from both that channel and from channel $N_{\text{chan}} + 1$ (due to aliasing) and so is probably best ignored.

[IMAGR](#) allows you to request automatic finding of the Clean boxes ([IM2PARM](#) of 1 through 6). In cases with low sidelobes, this works rather well, but you should probably keep an eye on what it does with [DOTV](#) 1 in any case. [IM2PARM](#)(12) controls the baseline-dependent time averaging while specifying the maximum field of view you expect. This allows you to reduce the size of the work file considerably which will at least reduce the time required for many of the steps in the imaging proportionally. It may be rather better than that if the work file is very large otherwise, requiring actual reading of the disk every time the data are accessed. Note, however, that the uniform weighting of your data will be affected. This averaging reduces the number of samples at short spacings disproportionately and so appears to reduce their weight in the imaging. Some [UVTAPER](#) could be used to compensate for this.

By default, bandwidth synthesis imaging assumes that the primary beam and all continuum sources are the same at every frequency. In fact, the primary beam size varies linearly with frequency (to first order anyway) and sources have spectral index. [IMAGR](#) will allow you to compensate for the average spectral index at almost no cost with [IMAGRPRM](#)(2). A far more accurate and expensive correction for spectral index may be made if you do the following. First image each spectral channel (or group of closely-spaced channels) separately. Combine them into a cube with [FQUBE](#), transpose the cube with [TRANS](#), and solve for spectral index images with [SPIXR](#). To use these images, set [IMAGRPRM](#)(17) to a radius (> 0) in pixels of a smoothing area and put the image name parameters in the 3rd and 4th input image names. Note that this algorithm is expensive, but that it can be sped up with judicious use of the [FQTOL](#) parameter. The change of primary beam with frequency may be corrected by setting [IMAGRPRM](#)(1) = 25 for the diameter of the [EVLA](#) dishes. Note that this algorithm is expensive, but that it can be sped up with judicious use of the [FQTOL](#) parameter. These two corrections work together, so that doing both costs very little more than doing just one of them.

A [RUN](#) file and procedure named [OOCAL](#) has been written to do self-calibration using the spectral-index image(s) and primary-beam correction in the manner used in [IMAGR](#). The procedure runs [OOSUB](#) to produce a divided data set, then [CALIB](#) to solve for complex gains, and finally [TACOP](#) to move the SN table to the real input data set. If your model is a large image rather than Clean components, the procedure [IMSCAL](#) in this [RUN](#) file will convert the model image into faceted Clean components and then run [OOCAL](#). Similar procedures for [FRING](#), called [OOFRING](#) and [IMFRING](#) are also available.

If you are observing a strong source and trying for very high dynamic range, you may have to correct for errors that are baseline- rather than antenna-dependent. One source of these errors is the antenna polarization leakage which affects the parallel-hand visibilities in a non-closing fashion. Task [BLCAL](#) can be used after you have as good an image as you can get without it. This task will divide the data by the model and average over a user-specified time to find baseline-dependent corrections which may then be applied to the data by setting adverb [BLVER](#). We recommend that you average the divided data over all of the times in your data to get a single correction for each baseline (and IF and polarization). If you use shorter intervals, you run the risk of forcing your data to look too much like your model. Since the polarization leakage is probably a function of frequency, an experimental version of [BLCAL](#) called [BLCHN](#) has been released. It determines the same correction but does not average over channels. [BLCHN](#) can even adjust the resulting correction table to correct for spectral index prior to applying and storing the values. The correction is saved in a table which [POSSM](#) and [BPLOT](#) are able to display. However, the calibration routines do not know

how to apply this table, so **BLCHN** writes out the corrected data as well as the table. A new task called **BDAPL** has been written to apply this BD table to another data set.

31dEC16 contains a new TV-menu driven editing task called **UFLAG**.² If your image shows stripes, indicative of residual bad data, this task allows you to view the data as it is gridded for imaging and to flag bad cells in the *uv* plane or even some, but not all, of the individual visibilities contributing to apparently bad cells.

E.11 Additional recipes

E.11.1 Columbian fresh banana cake with sea foam frosting

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

E.11.2 Chewy banana split dessert

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

²Greisen, E. W. 2016, "Editing on a *uv* grid in AIPS" AIPS Memo 121, <http://www.aips.nrao.edu/aipsdoc.html>

E.11.3 Banana July cocktail

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

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

E.11.4 Banana Dream Pizza

1. Preheat oven to 400° F. In a large bowl, combine 2 1/2 cups **all-purpose flour**, 2 tsp **baking powder**, and a pinch of **salt**. Add 4 Tsp softened **sweet cream butter** and blend. Add 3/4 cup warm **milk** and mix well. If the dough is still sticky, add a small amount of flour.
2. Form the dough into a ball. Knead it on a floured surface until it is smooth. Roll out the dough and place it in an oiled, 16-inch pizza pan. Bake for 15-20 minutes, or until the crust is light brown.
3. In a nonmetallic bowl, mash 4 **bananans**. Add 1 teaspoon **lime or lemon juice** and 6 tablespoons **honey**; mix well.
4. Slice 2 **bananas** horizontally and place the slices in water to cover. Add 1 teaspoon **lime or lemon juice** to prevent discoloration.
5. Spread the banana mixture on the crust.
6. Drain the sliced bananas and blot them with paper towels. Place them in a circular pattern on the banana mixture. Baste the banana slices with 3 tablespoons **melted butter**.
7. Bake for 20–30 minutes at 400° F until the crust is golden brown.
8. Remove from the oven and top with 1 quart **vanilla ice cream** and 1/2 cup chopped **macadamia nuts** while still hot. Serve immediately.

E.12 Concluding remarks, early science

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

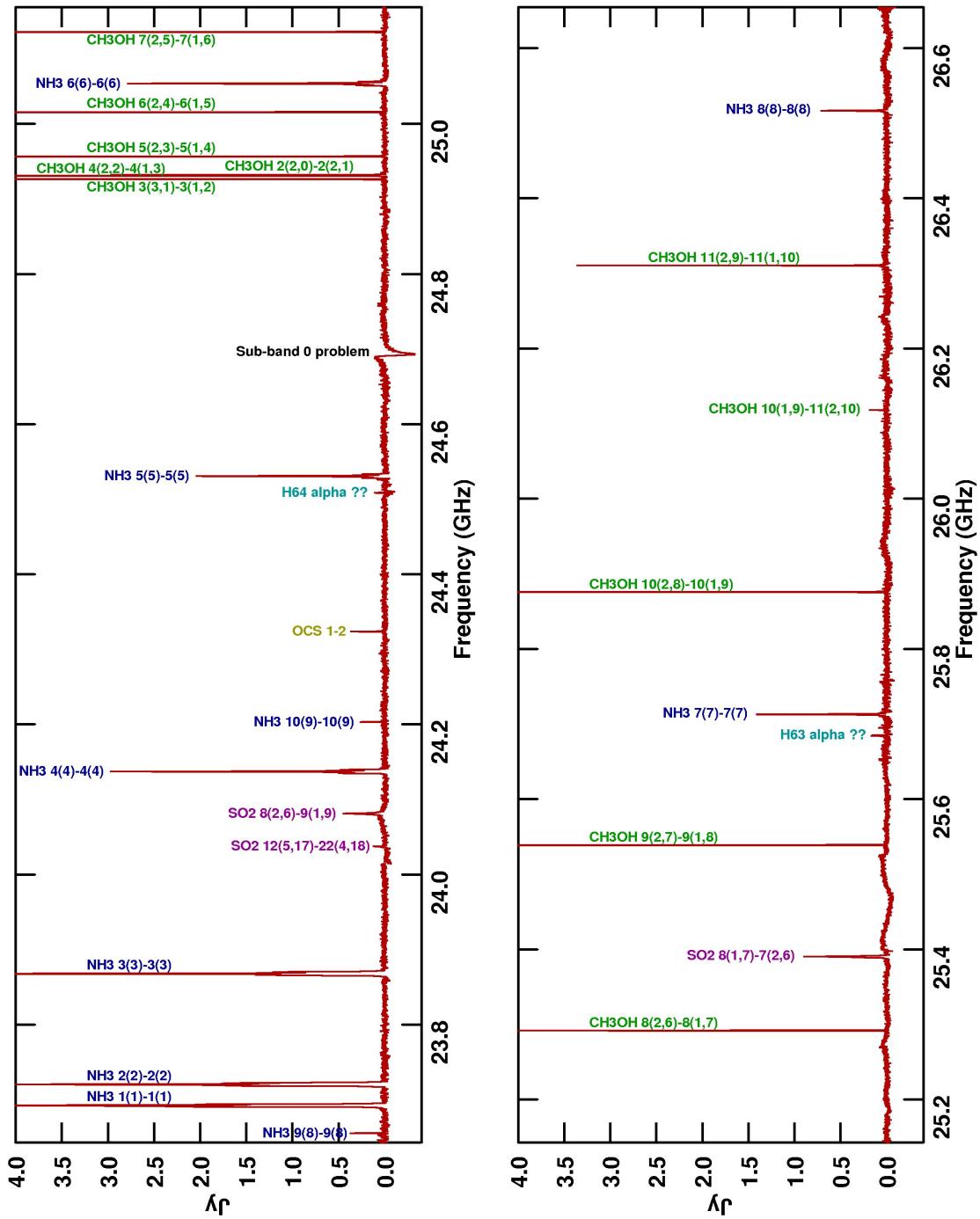


Figure E.2: Early EVLA science: The spectrum of the hot core of Orion A at K band. Three separate observations of 8192 channels each 0.125 MHz wide were made using 12 antennas in the D array. Two hours total telescope time went into each of the two lower thirds of the spectrum and 1 hour was used for the highest third. The plot was made using [ISPEC](#) over a 54 by 60 arc second area. Line identifications provided by Karl Menten.

L Special Considerations for EVLA P-band Data Processing in AIPS

The [EVLA](#) has been equipped with new, wide-band receivers in the range 230 MHz to 470 MHz, known as P-band. Most of the data reduction and imaging at this band are similar to those at other bands, so the following will assume some degree of familiarity with the rest of the *CookBook*, particularly Appendix E. This guide was initially prepared by Minnie Mao from an earlier guide written by Susan Neff. Enough has changed with the P-band system that a thorough revision of the Appendix has been done.

Note that the choice of file names, disk assignments, catalog numbers, and the like depends on your circumstances and will almost certainly not be the same as those shown in the examples below. Note also that experienced (but still reasonable) [EVLA](#) users disagree on the details of data reduction. Some compromise in these details has been attempted below.

L.1 P-band calibration and editing in AIPS

1. Download your data set from the archive found at

<https://archive.nrao.edu/archive/ArchiveQuery>

making sure that you select SDM-BDF data set (all files) as the download data format.

2. Before starting AIPS and while in the terminal window you intend to use for AIPS, enter

> cd *my_data_directory* C_R to change to the data directory containing the top-level directory of your data set.

> export MYAREA='pwd' C_R for bash shells, or

> setenv MYAREA 'pwd' C_R for tcsh and c shells.

Note the back ticks around *pwd* which cause the output of *pwd* to appear in the typed line.

3. Start AIPS with, for example,

> aips tv=local pr=3 C_R to use Unix sockets for the TV and printer 3
and enter your user number when prompted.

4. Look at the SDM-BDF file contents with [BDFLIST](#) — make sure you do *not* have a quote at the end of the file name, or else the lower case characters get changed into upper case and the file won't be found.

> default bdflist C_R initialize all adverbs
> dowait 1; doct1 1 C_R to get output on terminal as well as log file.
> asdmfile(1) = 'MYAREA: C_R two parts are concatenated.
> asdmfile(2) = 'TSUB0001.sb23643640.eb23666175.56454.14034020834 C_R
> bdflist C_R to examine file contents.

Pay particular attention to the configuration numbers. These refer to each unique correlator *and* antenna setup, not just the size of the current [VLA](#). Each configuration that you want will need to be loaded individually.

5. Load data into AIPS with [BDF2AIPS](#) — I'm doing everything on disk *d* because that's where there's more room. Note, do not use [DEFAULT](#) to retain the adverbs from the previous step. Loading data may take a while depending on the size of your data.

```
> outn 'my-name' CR to select a meaningful file name.  
> config c CR to select configuration c.  
> outdi d CR to select the disk.  
> bdf2aips CR to translate the data into AIPS.
```

Do this as often as needed to load all desired configurations.

6. Look at header of the AIPS file written by [BDF2AIPS](#)

```
> indi d; getn 1 CR to select disk and catalog number, assuming you had no files on disk d previously.  
> imh CR to see the details of the uv-data set header.
```

7. look at where the telescopes were during your observation with [PRTAN](#)

```
> default prtan CR  
> indi d; getn 1 CR  
> doct = -1 CR to make a printer listing to keep at your terminal.  
> go prtan CR
```

8. Look at the observation summary with [LISTR](#)

```
> default listr CR  
> indi d; getn 1 CR  
> opty 'scan' CR to list scans and sources.  
> doct = -1 CR to make a printer listing to keep at your terminal.  
> go CR
```

At this stage, Minnie's summary recommends separating P-band from 4-band data. At present, only P-band is recorded so that step is not needed. Then, Minnie recommends using [POSSM](#) to find dead antennas plus those with crossed polarization. Then she recommends [UVFLG](#) to flag the dead ones and [FIXRL](#) to uncross the mis-wired ones. These problems have been essentially eliminated. In addition, P-band data are now delivered by default in a number of relatively narrow spectral windows (called IFs in AIPS) which helps minimize data lost to [RFI](#) and eliminates the [NOIFS](#) and [MORIF](#) steps from Minnie's guide.

9. However, the requantizer gains are now allowed to vary and must be corrected.

```
> default tyapl CR to select the needed task  
> indi d; getn 1 CR to select the input file.  
> inext 'SY' ; optype 'PGN' CR to select requantizer gain correction.  
> inext 'SY' ; in2ver = 1 CR to apply the SysPower table, the version number is not defaulted.  
> outdis d ; go CR to stay on one disk.
```

10. Narrow-band [RFI](#) will normally cause ringing in P-band data. Use [POSSM](#) to examine some baselines from a calibration source to check this if you want to be careful. The default setup delivers data from outer IFs for which the sensitivity of the P-band receiver is poor. Use [to](#) drop the insensitive IFs and to do Hanning smoothing to reduce the ringing.

> default splat C _R	to select the needed task
> indi d; getn 2 C _R	to select the (new) input file.
> flagver 1 C _R	to apply the on-line flags.
> smooth = 1, 0 C _R	to do Hanning smoothing once and for all.
> bif = 3; eif = 15 C _R	to select the more sensitive IFs.
> outdis d ; go C _R	to put the output on the same disk.

11. Edit UV data with [EDITA](#) and the SY table. *The SY table P_{sum} parameter is sensitive to the occasional bursts of RFI which render the data useless. Flag all seriously discrepant points but do not flag really tightly since some noise in this parameter is normal and “correct.”*

```
> default edita CR
> indi d; getn 3 CR
> inext 'SY' CR
> go CR
```

12. At this time, it is best to identify those spectral channels affected by RFI throughout the run and to flag them once and for all. [FTFLG](#) will display the data combining all included baselines in a single display (per polarization), while the very similar [SPFLG](#) allows more discrimination (and more work) by displaying the baselines separately. Be sure to flag all sources if you flag channels over all times.

> default ftflg C _R	
> indi d; getn 3 C _R	
> calcode '*' C _R	to examine calibrators only
> dparm(6) = x C _R	to specify the basic integration time in the data in seconds.
> go C _R	

If you use [SPFLG](#), you should limit the times, sources, and/or baselines examined. The flagging then is best done with [UVFLG](#) from notes made while you study the data with [SPFLG](#).

> default uvflg C _R	
> indi d; getn 3 C _R	
> bif = bi ; eif = ei C _R	to set range of spectral windows.
> bchan = bc ; echan = ec C _R	to set range of spectral channels.
> antennas = a1, a2, a3 C _R	to select antennas to flag.
> baseline = b1, b2, b3, b4 C _R	to limit flags to the baselines from the pairs a1 through a3 with b1 through b4.
> reason 'bad channels' C _R	to set a reason in the flag table.
> go C _R	and repeat as needed with different IFs, channels, antennas, etc.

13. Calibrate for the delay errors with [FRING](#). Choose a time range in the middle of the observation where you know the antennas have settled down. A short time range is all you need — look at [LISTR](#) — because you don’t want time variations to come into play. This is also why you should choose something bright — like 3C286, which is used here as the name for the generic fringe- and baseline-fitting calibration source..

> default fring C _R	
> indi d; getn 3 C _R	
> calso '3C286' C _R	to select the primary calibration source.
> timerang 0 3 40 0 0 3 40 55 C _R	to select < 1 minute of good data from the calibration source.

```
> solint 1 CR to average over all included times.  

> aparm(5) 1 ; aparm(6) 1 CR to fit a single delay over all IFs and to print some extra  
detail in the fitting.  

> dparm(9) 1 CR to avoid any fitting for rates — important.  

> refant 22 CR to set a reference antenna, if one is known to be better  
than others.  

> go CR
```

This print level will display the delays found, which should all be a few nano-seconds although larger (correct) delays sometimes arise.

14. Apply these delay solutions to the calibration (CL) table with [CLCAL](#)

```
> default clcal CR  

> indi d; getn 3 CR  

> calso '3C286'; CR  

> snver 1 CR to specify the SN table version written by FRING.  

> go CR to copy CL table 1 to version 2 adding the delay  
correction, taken to be constant in time.
```

15. You should flag out any bad data on your bandpass calibrator. Some users feel that a quick-and-dirty flagging is enough at this stage. Use [CLIP](#) with a carefully chosen, perhaps IF-dependent, cutoff level. A more detailed editing is done by other users. Here are options to use [RFLAG](#) and [SPFLG](#). Maybe you'd like to [RFLAG](#) first and then eyeball with [SPFLG](#). The options are endless! The first execution of [RFLAG](#) determines flagging level parameters and makes plots on the TV for you to be sure that they are reasonable, The second execution applies these levels and determines new levels in case you want to run [RFLAG](#) even more times.

```
> default rflag CR  

> indi d; getn 3 CR  

> sources(1) '3C286' CR  

> docal 1 CR to correct delays and a-priori gains.  

> avgchan 11 CR to set the width of the median-window filter run across  
the spectral channels in each IF  

> fparm = 3, x, -1, -1 CR to do a small rolling time buffer, set the sample interval,  
and to have NOISE and SCUTOFF set IF-dependent cutoff  
levels.  

> fparm(13) = 1000 CR to clip amplitudes at 1000 Jy. There are many more  
FPARM options you may choose also.  

> doplot 8 ; dotv 1 CR to do the 2 most useful plots on the TV.  

> go CR
```

[RFLAG](#) will return cutoff levels to the appropriate adverbs. Therefore, to apply the flags using the flagging levels found above, do *not* do a tget rflag which will reset the levels to zero. Instead, simply do

```
> doplot = -8 CR to apply the cutoff levels and determine new ones,  
plotting on the TV.
```

```
> go CR
```

[RFLAG](#) will make a new flag table each time it determines flags ([DOPLLOT](#) \leq 0). Check the results with

```
> default spflg CR  

> indi d; getn 3 CR  

> sources(1) '3C286' CR to examine only the bandpass calibrator.
```

```
> dparm(6) x CR                                to set sampling time in seconds
> docal 1 CR                                  to correct delays and a-priori gains.
> go CR
```

It is at this point that experienced users differ significantly in their approach. If you have only one bandpass-calibration scan and the ionosphere was reasonably behaved (phases nearly constant through that scan), then you should run **BPASS** on the calibration source in place. However, when one has multiple bandpass scans and/or a badly behaved ionosphere, it may be best to split the calibration scans out and self-calibrate them on a short time interval. The addition of multiple scans where the source has been phased up, but the remaining **RFI** has not, will reduce the effect of that **RFI** substantially. This scheme is described next.

16. Split off your calibrator for bandpass calibration — this is only necessary for self-calibrating the BP calibrator.

```
> default split CR
> indi d; getn 3 CR
> outdi indi ; source '3C286' CR          to select calibrator and stay on the same disk.
> outcl 'bpspl' CR                         to indicate the use for this file.
> docalib 1 CR                            to apply delay and a-priori calibration.
> go CR                                    To make a single-source file.
```

17. Self-cal on your bandpass cal with **CALIB** to solve for wiggles in phase as a function of time.

```
> default calib CR
> indi d; getn 4 CR
> refant 22 CR                           to stay with the chosen reference antenna.
> solint y CR                          to average n records in each solution interval, set y = nx.
> solty 'L1R' ; solmode 'P' CR          to do phase-only solutions.
> aparm(6) 1 CR                        to get some diagnostic messages.
> go CR                                to write a new SN table.
```

Eyeball this new table with **SNPLT** — should be flat, not all over the place.

```
> default snplt CR
> indi d; getn 4 CR
> inext 'sn' ; opty 'phas' CR          to plot the phases in the new table.
> nplots 4 ; dotv 1 CR                to have 4 plots per TV page.
> opco 'alif' ; do3col 1 CR          to combine all IFs in each colorful plot.
> go CR
```

Or, if these plots are too crowded, turn off **OPCODE** and **DO3COL**.

18. Use **BPASS** to calibrate the bandpass shape — corrects for wiggles in phase and amplitude as a function of frequency

```
> default bpss CR
> indi d; getn 4 CR
> docalib 1 CR                          to apply the new SN table phases.
> refant 22 CR
> solty 'l1r' ; solint = -1 CR          to average the entire time range.
> bpssprm(5) = 1; bpssprm(10) = 3 CR  to normalize the solutions at the end rather than the data record-by-record.
```

> go C_R To generate a new BP table containing one record per antenna.

19. Eyeball the resulting bandpass (BP) table — **APARM(8) = 2** plots the BP table in **POSSM**. If you see odd bandpass shapes or spikes in the bandpass, you may need to edit your data more extensively and repeat steps 15 through 18.

```
> tget possm CR
> indi d; getn 4 CR
> solint = -1 CR
> aparm(8) 2 ; aparm(9) 1 CR to plot all IFs together for each bandpass.
> nplots 2 CR 2 per page is enough here.
> go CR
```

20. Copy the bandpass table back to your main data with **TACOP**.

```
> default tacop CR
> indi d; getn 4 CR
> inext 'bp' CR
> outdi indi ; geton 3 CR to select output name parameters
> go CR
```

21. Since **RFLAG** probably made a large number of flags, let us copy the data file with **UVCOP** and apply the flags.

```
> default uvcop CR
> indi d; getn 3 CR
> outdi indi ; outn inna CR to stay on d.
> outcl 'fixcp' CR with a meaningful class name
> flagver n CR to apply the highest flag table (n); use IMHEAD if unsure.
> go CR Version 0 means no flagging here.
```

22. Use **SETJY** to compute the flux for the primary flux calibrator (3C286), storing it in the SU table.

```
> default setjy CR
> indi d; getn 5 CR
> source '3C286' CR specify all "known" calibrator sources.
> opty 'calc' CR to calculate and display fluxes from the frequencies plus
tables of known sources.
> go CR
```

23. Go through all of your calibration sources, one at a time, to check for **RFI** with **POSSM** and, if present, to address it with **SPFLG** and/or **RFLAG**

```
> tget possm CR
> source 'calsrci' CR to examine the ith calibration source.
> indi d; getn 5 CR
> docal 1 ; dobond 1 CR apply our hard-won calibration.
> solint 0 CR to look at the average over all time.
> aparm(9) 1 CR to plot all IFs together.
> nplots 2 CR 2 per page is enough here.
```

> go C_R

If RFI is present (it almost certainly will be), decide whether SPFLG might be worth the work. If so

> default spflg C_R

> indi d; getn 5 C_R

> source 'calsrc_i' C_R

to examine the *i*th calibration source.

> dparm(6) x C_R

> docal 1 ; dobond 1 C_R

apply the calibration.

> go C_R

Two rounds or more of RFLAG may also be needed. Check with POSSM to be sure.

> default rflag C_R

> indi d; getn 5 C_R

> source 'calsrc_i' C_R

to examine the *i*th calibration source.

> docal 1 ; dobond 1 C_R

apply the calibration.

> avgchan 11 C_R

to set the width of the median-window filter run across the spectral channels in each IF

> fparm = 3, x, -1, -1 C_R

to do a small rolling time buffer, set the sample interval, and to have NOISE and SCUTOFF set IF-dependent cutoff levels.

> fparm(13) = 1000 C_R

to clip amplitudes at 1000 Jy. There are many more FPARM options you may choose also.

> doplot 8 ; dotv 1 C_R

to do the 2 most useful plots on the TV.

> go C_R

RFLAG will return cutoff levels to the appropriate adverbs. Therefore, to apply the flags using the flagging levels found above, do *not* do a tget rflag which will reset the levels to zero. Instead, simply do

> doplot = -8 C_R

to apply the cutoff levels and determine new ones, plotting on the TV.

> go C_R

RFLAG will make a new flag table each time it determines flags (DOPLOT ≤ 0).

24. Since RFLAG probably made a large number of flags, let us copy the data file once again with UVCOP and apply the flags.

> default uvcop C_R

> indi d; getn 5 C_R

> outdi indi ; outn inna C_R

to stay on *d*.

> outcl 'flaged' C_R

with a meaningful class name

> flagver n C_R

to apply the highest flag table (*n*); use IMHEAD if unsure. Version 0 means no flagging here.

> go C_R

25. Since all flag tables $< n$ were already merged into table *n* which we have applied, delete all flag tables in the output file.

> default extdest C_R

> indi d; getn 6 C_R

> inext 'fg' ; invers = -1 C_R

to delete all remaining versions of flag table.

> extdest C_R

26. Find the amplitude and phase calibration as a function of *time* using [CALIB](#)

```
> default calib CR
> indi d; getn 6 CR
> calcode '*' CR
> docalib 1 ; dobands 1 CR
> refant 22 CR
> solint 0 CR

to select all calibration sources
to apply all existing calibration.
to keep our reference antenna.

One solution per calibrator scan is okay with good
short-term phase stability. Otherwise, set to the longest
time in minutes over which phase is stable.

> solty 'L1R' ; solmode 'a&p' CR
to solve for amplitude and phase with an L1 method
iterated robustly.

> aparm(6) 1 CR
to print closure error statistics.

> go CR
```

27. The solution table (SN) produced by the previous step will contain amplitude gains at multiple levels, correct ones for those sources with fluxes in the source table (3C286) and incorrect ones for those which [CALIB](#) was forced to call 1 Jy. To solve for the unknown fluxes of the secondary calibrators

```
> default getjy CR
> indi d; getn 6 CR
> source 'calsrc1', 'calsrc2', 'calsrc3', ... CR
to list the secondary calibrators.
> calsour = '3c286' CR
to list the known calibrator(s).
> snver = n CR
to select only the SN table just written by CCALIB. Use
IMHEADER to find the maximum version number n.
> go CR
to solve for the secondary fluxes as functions of IF.
```

Check the values displayed carefully to make sure that they are sensible. [SOUSP](#) offers tools to examine and adjust the fluxes and SN table *n*.

28. Look at both amplitude and phase solutions with [SNPLT](#).

```
> tvini CR
> default snplt CR
> indi d; getn 6 CR
> inext 'sn' ; opty 'amp' CR
to plot amplitudes.
> nplots 4 ; dotv 1 CR
to plot 4 panels on each TV page.
> opco 'alif' ; do3col 1 CR
to plot all IFs in each panel using color to distinguish
them.

> go CR
```

One can also plot IFs in separate panels or one at a time if these plots are too crowded. Then

```
> opty 'ph'; go CR
To check the phases.
```

29. Apply the solutions to the calibration table with [CLCAL](#)

```
> default clcal CR
> indi d; getn 6 CR
> calcode '*' CR
to select all calibration sources
> sampty 'box' ; bparm 0.3, 0.3 CR
To smooth the SN table if you used SOLINT less than full
scans. Leave this out if you used scan averages.

> refant 22 CR
> go CR
```

30. Use [SNPLT](#) to check that the solutions have been interpolated between calibrators.

```
> tget snplt CR
> inext 'cl' CR
> go CR
```

31. Make an image of 3C286 as a sanity check using [IMAGR](#)

> default imagr C _R	to apply the calibration.
> indi d; getn 6 ; outdi indi C _R	to select only the primary calibrator.
> docal 1 ; dobands 1 C _R	to omit edge channels in each IF, where n is the number of channels in an IF, now usually 128.
> source = '3c286' C _R	to average all spectral channels into the image.
> bchan 5; echan = $n - 4$ C _R	to set the image cell spacing — depends on configuration.
> nchav = echan-bchan+1; chinc nchav C _R	to make a largish image to see sources around 3C286.
> cellsize 1. C _R	
> imsi 1024 C _R	
> outn '3C286_quick' C _R	
> niter 100 ; dotv 1 C _R	to do some Cleaning and guide the progress with the TV
> go C _R	

32. At this point, it may be simpler to [SPLIT](#) the data set and apply calibration to the target sources which we then can work with one at a time.

```
> defult split CR
> indi d; getn 6 CR
> outdi indi CR
> docal 1 ; dobands 1 CR
> sources = 'target1', 'target2', ... CR      to select only target sources.
> go CR
```

To create a number of separate, single-source *uv* data sets each with separate target source.

33. Flag [RFI](#) and bad data out of the target data set with [SPFLG](#), which can be tedious.

```
> default spflg CR
> indi d; getn 9 CR      to select the first of the split target data sets.
> dparm(6) x CR        to set integration time.
> go CR
```

Then do two or more rounds of [RFLAG](#).

```
> default rflag CR
> indi d; getn 9 CR
> avgchan 11 CR          Median window width on spectra.
> fparm = 3, x, -1, -1 CR to do a small rolling time buffer, set the sample interval,
                               and to have NOISE and SCUTOFF set IF-dependent cutoff
                               levels.
> doplot 8 ; dotv 1 CR    To examine the two important plots on the TV.
> go CR
```

and then apply the flags levels and determine new ones

```
> doplot -8 ; go CR
```

and perhaps repeat this line for more flagging.

34. Because we used [TYAPL](#) only to apply the re-quantizer gains, your data weights simply reflect integration time. For better continuum imaging, weights should reflect the actual data uncertainties. So re-weight your data with [REWAY](#)

```
> default reway CR
> indi d; getn 9 CR
> outdi indi CR
> aparm 31,0,500 CR
```

to find rmses from rolling buffers 31 time intervals long on a baseline basis and then smooth them over 500 seconds.

```
> go CR
```

Note that this applies the final [SPFLG-RFLAG](#) flag table. Delete the other flag table versions with

```
> default extdest CR
> indi d; getn 10 CR
> inext 'fg' ; invers = -1 CR
> extdest CR
```

to delete all remaining versions of flag table.

L.2 P-band imaging in AIPS

35. The following steps relate to imaging your calibrated data. As the aim of this appendix is calibration, the imaging will be discussed only briefly here. Make a dirty image of the center with [IMAGR](#)

```
> default imagr CR
> indi d; getn 10; outdi indi CR
> bchan 5; echan = n - 4 CR
> nchav = echan-bchan+1; chinc nchav CR
> cellsize 10 CR
> imsi 1024,1024 CR
> niter 0 CR
> go CR
```

to omit edge channels in each IF, where n is the number of channels in an IF.

to average all spectral channels into the image.

to set a cell size which will depend on the [VLA](#) configuration.

to do no Cleaning

36. Clean image with faceting, auto-boxing, and TV display; we use [DEFAULT IMAGR](#) here to initialize all of [IMAGR](#)'s many parameters

```
> default imagr CR
> indi d; getn 10; outdi indi CR
> task 'setfc' CR
> cellsize 0 ; imsize 0 CR
> bparm 2.2, 5, 2, 8, 0.3 CR
> flux 1 CR
> boxfile 'MYAREA:Target1.box' CR
> go CR
```

to have [SETFC](#) determine the correct cell and image size for these data.

to image fully out to a radius of 2 degrees and to include 0.3 Jy NVSS sources out to a radius of 8 degrees.

to include exterior sources above 1 Jy only.

to list the facets found.

The following does not use default since we need adverbs set for, and by, [SETFC](#), such as inname, imsize, and cellsize

```

> task 'IMAGR' CR
> flux 0 CR
> bchan 5; echan = n - 4 CR          to omit edge channels in each IF, where n is the number
                                           of channels in an IF.

> nchav = echan-bchan+1; chinc nchav CR    to average all spectral channels into the image.

> niter 5000 CR                         to Clean 5000 iterations.

> do3dim true ; overlap 2 CR              to use true 3-D geometry, Cleaning one facet at a time in
                                           each cycle (recommended).

> dotv true CR                          to control the imaging with the TV.

> oboxfile boxfile CR                   to save all changes to Clean boxes.

> im2parm=2,0 CR                        to allow up to 2 Clean boxes to be created automatically
                                           at each cycle.

> go CR
```

Note, the field may have enough spectral index variation to require imaging (and self-calibration) of each spectral window (IF) individually.

37. There is normally enough flux in a P-band field to do phase self-calibration and we recommend that you now do that. Use **CALIB** with the output of step 36 as the model including all **NFIELD** facets as the **NMAPS** images.

```

> default calib CR
> indi d; getn 10; outdi indi CR
> nmaps = nfield CR                     to include all facets
> in2d = indisk; get2n M CR           to set the 2nd file name to the output image of class
                                           ICL001 (catalog number M).
> soltyp 'l1r' ; solmode 'p' CR        to start with phase-only solutions.
> solint = nx CR                      to set a short integration time as n times the sampling
                                           interval.

> go CR
```

38. Examine the output **SN** table with **EDITA** to see if there are times with bad solutions. If not, repeat steps 36 and 37 using

```

> tgt imagr; docal = 1; go CR            to make new images.
> tgt calib; in2seq = 2 CR             to point at the second set of images.
> solmode 'a&p' CR                  to do amplitude as well if there is enough flux in the field.
> go CR
```

Re-examine the new **SN** table with **EDITA**. In particular, watch for periods of time when the amplitude gain of an antenna gets rather larger than normal. It has been found that the ionosphere can defocus sources on scales as small as the antenna diameter. It is best to delete such data.

39. Make image(s) and final calibrated data sets into FITS files with **FITTP**. Exit AIPS and

```

> setenv FIT /lustre/mmao/Pband CR      for tcsh and c shells
> export FIT=/lustre/mmao/Pband CR       for bash shell
```

Then start AIPS again

```

> default fittp CR                      to select an image or data set
> indi d; getn M CR                to make an appropriate output name
> dataout 'FIT:VirA.FITS' CR
> go CR
```

And repeat with new *M* and **DATAOUT** values.

L.3 Additional Recipes

L.3.1 Banana mandarin cheese pie

1. In large mixer bowl, beat 8 ounces softened **cream cheese** until fluffy.
2. Gradually beat in 8 ounces **sweetened condensed milk** until smooth.
3. Stir in 1 teaspoon **lemon juice** and 1 teaspoon **vanilla extract**.
4. Slice 2 medium **bananas**, dip in lemon juice, and drain.
5. Line 8(?)-inch **graham cracker pie crust** with bananas and about 2/3 of an 11-ounce can (drained) **mandarin oranges**.
6. Pour filling over fruit and chill for 3 hours or until set.
7. Garnish top with remaining orange segments and 1 medium **banana** sliced and dipped in lemon juice.

L.3.2 Banana poundcake

1. Mix in large bowl until blended:

1 $\frac{1}{3}$ cups mashed **bananas** (4 medium)
 1 pkg. (18 $\frac{1}{2}$ oz.) **yellow cake mix**
 1 pkg. (3 $\frac{3}{4}$ oz.) instant **vanilla pudding mix**
 $\frac{1}{3}$ cup **salad oil**
 $\frac{1}{2}$ cup **water**
 $\frac{1}{2}$ teaspoon **cinnamon**
 $\frac{1}{2}$ teaspoon **nutmeg**
 4 **eggs** at room temperature

2. Beat at medium speed for 4 minutes.
3. Turn batter into greased and lightly floured 10-inch tube pan.
4. Bake in 350° F oven for 1 hour or until cake tester inserted in cake comes out clean.
5. Cool in pan 10 minutes, then turn out onto rack and cool completely.
6. If desired, dust with confectioners sugar before serving.

Thanks to the United Fresh Fruit and Vegetable Association.

L.3.3 Frozen Push-Ups

1. Peel 2 **bananas** and slice into blender or food processor.
2. Add 1 6-ounce can frozen **orange juice** (thawed), 1/2 cup instant non-fat **dry milk**, 1/2 cup **water**, and 1 cup plain low-fat **yogurt**.
3. Cover and blend until foamy. Pour into small paper cups and freeze.
4. To eat, squeeze bottom of cup.

Thanks to Ruthe Eshleman *The American Heart Association Cookbook*.

F File Sizes

Your data reduction strategy will be more effective if you have an idea of how big the data and map file sizes will be on disk. Also, it will help you estimate just how many files you can backup to tape. In this appendix, we will discuss file sizes in bytes, which are 8 bits in size, rather than “blocks,” which can vary in size between different computers. Inside *AIPS*, the definition of “byte” is perverted, but all systems now use a 1024-byte (8192 bit) “block.”

F.1 Visibility (*uv*) data sets

uv data files contain information about the coherence function of a wavefront at random locations and random times. Consequently, the way this information is stored on disk is different from that for images where the pixels are on a regular grid. AIPS *uv* data are stored on disk in a manner similar to the way that it would be organized on a FITS “random group” tape. The data are stored as logical records; each record (a “visibility”) contains all the data taken on one baseline at a given time. Consequently, a record may contain information for several IFs, several frequencies at each of those IFs and more than one polarization combination for each frequency/IF. The first part of each logical record contains what are known as “random parameters” e.g., spatial frequency coordinates and time. After the random parameters, there is a small, regular array of data.

For a multi-source data set such as might be created by **FILLM**, the random parameter group will include the following. **UU-L-SIN**, **VV-L-SIN**, and **WW-L-SIN** give the spatial frequency coordinates, computed with a sine projection in units of wavelengths at the reference frequency. **TIME1** is the label for the time in days. **BASELINE** is the baseline number ($256ant_1 + ant_2 + subarray/100.$) and **SOURCE** is the source number corresponding to an entry in the source table. If you have frequency table identifiers (which is usually the case these days), then there will be an additional random parameter, **FREQSEL**. For a compressed database, two additional random parameters will be required — **WEIGHT** to give a single data weight for all samples in the record and **SCALE** to give the gain used to compress that record.

The regular data array is similar to an image array in that the order of axes is arbitrary. However, the convention is for the first axis to be of type **COMPLEX**, having a dimension of 3 for uncompressed data (real, imaginary, weight) and a dimension of 1 for compressed data. The other axes of the regular array are **IF**, **RA**, **DEC**, **FREQ** and **STOKES**.

F.1.1 *uv* database sizes

The number of words in each “visibility” is given by

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

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

$$(\text{word per record}) \times (\# \text{ vis}) \times 4 \text{ bytes}$$

The number of visibilities is given (approximately) by

$$\frac{\text{length of observation}}{\text{integration time}} \times \text{number of baselines}$$

where the number of baselines is the usual $\frac{1}{2}n(n - 1)$. This is equal to 351 for the **VLA** for the usual 27-antenna array.

For example, a 12-hour observation with 30 second integrations, one frequency and 2 IFs with RR, RL, LR and LL written in compressed format (`DOUVCOMP = TRUE` in `FILLM`) will occupy about 34 Mbytes on disk. In practice, the *uv* file will usually be a little larger due to the way the system allocates space on the disks. You must also remember to allow room for the extension tables — see § F.3. If this database had been written in uncompressed format, the *uv* data would have occupied around 62 Mbyte, but would have carried information about different data weights for different IFs.

Consider another example illustrated by the `IMHEAD` listing below:

```
Image=MULTI      (UV)          Filename=20/07/90    .L BAND.   1
Telescope=VLA           Receiver=VLA
Observer=AFTST          User #= 1364
Observ. date=20-JUL-1990 Map date=23-JUL-1990
# visibilities 105198   Sort order TB
Rand axes: UU-L-SIN VV-L-SIN WW-L-SIN BASELINE TIME1
              SOURCE FREQSEL WEIGHT SCALE
-----
Type    Pixels   Coord value at Pixel   Coord incr   Rotat
COMPLEX     1   1.0000000E+00   1.00 1.0000000E+00   0.00
STOKES      4   -1.0000000E+00  1.00-1.0000000E+00   0.00
IF          2   1.0000000E+00   1.00 1.0000000E+00   0.00
FREQ        1   1.4524000E+09   1.00 2.5000000E+07   0.00
RA          1   00 00 00.000    1.00      3600.000   0.00
DEC         1   00 00 00.000    1.00      3600.000   0.00
-----
Maximum version number of extension files of type HI is   1
Maximum version number of extension files of type AN is   1
Maximum version number of extension files of type NX is   1
Maximum version number of extension files of type SU is   1
Maximum version number of extension files of type FQ is   1
Maximum version number of extension files of type CL is   1
Maximum version number of extension files of type SN is   2
```

This compressed (COMPLEX Pixels = 1) *uv* database contains 9 random parameters, 4 polarizations, 1 frequency, and 2 IFs. for *each* of 105198 visibilities. The size of the database file itself is, therefore,

$$\left[\{9 + [1 \times 4 \times 1 \times 2]\} \times 105198 \times 4 \right] \text{bytes} = 7.153 \text{ Mbytes.}$$

Note that this data set is 17/33 the size of an uncompressed data set.

F.1.2 Compressed format for *uv* data

The use of “compressed” data can make substantial savings in the amount of disk space that you require, particularly for spectral-line databases. All tasks should now be able to handle either the compressed or the uncompressed formats. Compressed data files can be identified by the dimension of 1 for the COMPLEX axis in the database header. (Uncompressed data will have a dimension of 3.) The savings can be close to a factor of three for spectral line observations.

This is achieved by converting all data weights into a single `WEIGHT` random parameter, by finding a single `SCALE` random parameter with which to scale all real and imaginary parts of the visibilities into 16-bit integers, and by packing the real and imaginary terms into one 32-bit location using magic-value blanking for flagged data. This is to be compared with the uncompressed format in which each of the real, imaginary and weight terms are each stored in a 32-bit floating-point location. The use of a single weight value masks

real differences in system temperatures between polarizations and IFs, which one should retain for the lowest possible noise in imaging.

In general, data compression is a good thing and should be used, but with a little caution. With a single frequency, single IF, and single polarization, you will not save any disk space. In all other cases, there are respectable savings to be made. However, the use of a packed data word for the real and imaginary parts of the visibility function along with magic value blanking imposes a restriction on the “spectral dynamic range” of the data set of around 32000:1. Consequently, there are some situations where compressed data should *not* be used. For example, if the spectral dynamic range in the *uv* database is likely to be greater than, say, 1000:1, you must use *uncompressed* data format to avoid loss of accuracy. This situation can arise in maser spectra, for example, in which there are maser lines of 1 Jy and > 32000 Jy; in this case, you should never use compressed data. Bandpass calibration can cause large correction factors to be applied to the edge channels of a database. In the presence of noise or interference, bad channels can become very much greater in amplitude than good channels. In such cases you must either use uncompressed format or be very careful to flag bad channels or to drop them with the **BCHAN** and **ECHAN** adverbs as you apply the bandpass calibration. In general, continuum data sets can be loaded with data compression since these dynamic range considerations will not normally apply. The loss of weight information may not be worth the savings in disk space, however. Compressed data takes less time to read/write due to the small number of bytes, but casts in cpu time some due to the extra computation.

If there has been on-line or later flagging that depends on polarization, IF, or spectral channel (*i.e.*, **RFI** excision) or differences in the intrinsic weights between polarizations, IFs, or spectral channels (*i.e.*, different system temperatures for different IFs), then data compression causes a serious loss of information related to the data weight.

F.2 Image files

Since images are regular arrays, the sizes of image files are easier to calculate. The images are stored as floating-point numbers, *i.e.*, 32 bits per pixel, so the image file size in bytes is given by

$$4 \times \prod_i \text{length}(i)$$

where $\text{length}(i)$ is the number of pixels on the i^{th} axis. For example, a 128-channel cube with each plane a 256×256 image will require around 34 Mbytes of disk storage space. This may be increased a little due to the way in which the system allocates space for files.

F.3 Extension files

Subsidiary data about *uv* database and image files are written in “extension files.” These include, for example, history records (HI files), plot instructions (PL files), calibration solutions (SN files), ad nauseum. Some extension files can become large. A history file uses 1024 bytes for every 14 history records, a small amount under most circumstances. A plot file is normally small, but the output from **GREYS** can be as large as one plane of the image and the output of **KNTR** is larger by a factor of the number of panes in the plot.

The CL (calibration) table contains the total model of the interferometer at each interval. Many of these logical records are blank in the case of most interferometers but, for the VLBA, these records will contain essential information. The CL table contains a logical record for each antenna in the array at each CL time

stamp. The CL time stamps are set by the user when loading the data. The default for **VLA** data is every 5 minutes. For VLBI data, it is every 1 minute. Each CL logical record requires

$$\{15 + [14 \times (\# IFs) \times (\# pols)]\} \times 4 \text{ bytes}$$

For a 12-hour observation with the **VLA** with 2 IFs and 4 polarization pairs, with entries every 5 minutes, the CL table will occupy about

$$\left[\{15 + [14 \times 2 \times 2]\} \times 4 \times \frac{(12 \times 60)}{5} \times 27 \right] \text{ bytes} = 1.05 \text{ Mbytes}$$

Since most other files are considerably smaller, their sizes can be ignored. That they exist and may require some disk, should not be forgotten. To look at the full *AIPS* disk usage on your computer in summary form:

- | | |
|--|--|
| > TASK 'DISKU' ; INP C _R | to review the inputs. |
| > INDISK 0 ; DETIME 0 C _R | to look at all disks and all data sets |
| > USERID 32000 C _R | to look at all user numbers. |
| > DOALL FALSE ; GO C _R | to run the task in a summary mode. |

Then to look at file sizes in detail:

- | | |
|---|---|
| > INDISK <i>n</i> ; USERID 0 C _R | to restrict the display to your data sets and one disk. |
| > DOALL TRUE ; GO C _R | to run the task to list all files on disk <i>n</i> . |

F.4 Storing data on tape

Images and *uv* databases are written to magnetic tape by **FITTP** for archival purposes and for transfer to other computers and sites. Three FITS-standard formats are available, controlled through the adverb **FORMAT**. The preferred format is 32-bit floating point (IEEE standard) format. There are no dynamic range limitations in this format and, on many modern computers, no bit manipulation is required since they use IEEE floating internally.

Of the two integer formats, there is little reason to use the 32-bit integer since it poses dynamic range, rescaling, and other problems with no saving in space. The 16-bit integer format uses 16-bit signed 2's complement integers to represent the data. Such numbers are limited to the range -32768 to 32767. **FITTP** has to find the maximum and minimum in the image and then scale the data to fit in this numeric range. For images of limited dynamic range, this format is perfectly adequate. In fact, **FITAB** offers the option to reduce the dynamic range even further with the **QUANTIZE** adverb. For images written to FITS disk files, this allows for better compression before the files are transmitted over the Internet. For high-dynamic range images, the 16-bit format may not be adequate. (The integer formats are no longer allowed for *uv* data. More than one user has reduced all his "good" spectral channels to pure 0 by scaling all the *uv* data to include one really horrendously bad sample.) A less important benefit of the floating point format is that the numbers representing your data are recorded exactly on tape as they are stored on disk; there are no "quantization errors". This may be important for software development.

The preceding paragraphs do not tell the full story, however. The portion of the FITS standard used by **FITTP** does not allow for *uv* data on tape in a compressed format. Instead, **FITTP** expands the data into the uncompressed form and then writes the data on tape. In the conversion, the real and imaginary values that were stored in one packed number are expanded into three real values — one each for real, imaginary and weight terms — and the weight and scale random parameters are removed since they are no longer required. Consequently, the compressed data are expanded to

$$\frac{\{(\# \text{random parameters} - 2) + [(\# pol) \times (\# IFs) \times (\# frequencies) \times 3]\}}{\{(\# \text{random parameters}) + [(\# pol) \times (\# IFs) \times (\# frequencies)]\}}$$

the original size (where $\# \text{ random parameters}$ is the original number in the compressed database).

As an example, let us consider a multi-source spectral-line database stored on disk in compressed format. The data set has seven channels each at 2 IFs with 2 polarizations. There are nine random parameters and 834031 visibilities. From § F.1, we can calculate the size of the *uv* file to be 123 Mbytes. (Remember, this doesn't include any of the extension files, some of which might be several Mbytes in size.) Before the file is written to tape in 32-bit floating format, it is first expanded by a factor of

$$\frac{\{(9 - 2) + [2 \times 2 \times 7 \times 3]\}}{\{9 + [2 \times 2 \times 7]\}} = 2.333.$$

Consequently, the data will occupy

$$123 \times 2.333 \text{ Mbytes} = 287 \text{ Mbytes}$$

on tape. In other words, this database and all the associated extension files will not fit on a standard, 6250 bpi tape even using **BLOCKING** = 10, but modern tape technology solves this problem.

Note that **FITTP** writes history file data into the FITS header and writes table extension files as extensions after the main image or data set within the same tape file. Plot (PL) and slice (SL) files are not saved to tape.

Task **FITAB** uses "binary tables" to represent visibility data rather than the old, mildly deprecated "random groups" form of the FITS format. This has several advantages for *uv* data. It allows compressed data to be recorded in that form exactly (except for byte-order questions which should not concern the reader). It also allows the data and attached tables to be divided into pieces which will reduce the size of files to be copied over the Internet, making copying and tape storage somewhat more reliable. The principal disadvantage of **FITAB** *uv* data output is that only two packages can read it so far. The one "outside" package that reads the format is **obit** available from Bill Cotton at NRAO Charlottesville.

F.4.1 DAT and Exabyte tapes

The arrival of modern tape technologies has hastened the demise of 9-track tapes. First Exabyte (8mm) and then DAT (4mm) have provided much higher storage capacities than the 9-track tapes and have also provided faster seeks between file marks and greater data reliability. The new technologies are very much cheaper as well, in part because they have been adopted by the PC market. They are both technically quite complex internally. The DAT tape has a "system log" area at the beginning which allows for the fast seeks. It is a bit fragile, however, since it is updated when the tape is unloaded and hence can be incorrect if there is an unfortunate power failure. Both technologies are still evolving and both now offer various data encoding/compression options. Unfortunately, the data compression techniques vary considerably with tape model and manufacturer and hence should not be used to archive or transport data. The data are blocked on the tapes by means known only to the manufacturers and are not significantly under user control. It is still probably good to use a large **BLOCKING**, but only for I/O transfer reasons. The EOF marks can be expensive on these tape devices.

Exabytes at low density have a capacity of about 2.2 Gbytes on a 112m tape and use about 1 Mbyte (or maybe even 4 Mbytes) for each EOF mark. The large size of the EOF limits the number of files you can write rather significantly. The EOFs are also slow to process mechanically. Exabytes at high density have a capacity of 4.5 Gbytes on a 112m tape and use 48 Kbytes per EOF mark. DATs have a capacity of 2.0 Gbytes on a 90m tape, but also come in 60m and 120m sizes. The EOF mark size is not readily available, but is probably no more than 48 Kbytes. The early warning of the end-of-medium is 40 Mbytes before the actual end of tape.

F.5 Very large data sets

FITTP and **FITAB** cannot write multi-volume tapes. Some spectral-line and VLBA databases (and perhaps some continuum databases) may be so large that the file cannot fit on one modern tape even with **BLOCKING** = 10. What can you do to backup your data? The simplest solution is to use **FITAB**. This task can write *uv* data in compressed form and can break up a data set into “pieces.” You write as many pieces as will fit on the first tape, noting what the piece number is when the end-of-tape is reached. You can then tell **FITAB** to begin with that piece number on the next tape (**BDROP** = $n-1$ where n is the number of the piece that encountered the end-of-tape). Unfortunately, the FITS format used by **FITAB**, while perfectly legitimate, is only understood by *AIPS* versions beginning with 15APR99. Furthermore, **FITAB** output is only understood by *AIPS* and **OBIT**. If the data set is so large that it will not fit on the required tape device using **FITTP** or the data must be taken to a system that does not understand **FITAB**’s format, there are several approaches that you can adopt.

First, you could **SPLIT** out the database into *single-source* databases and back each of these up individually. Alternatively, you could subdivide the large database in several smaller databases with **UVCOP** by specifying a different time range for each of the smaller databases and then back these up individually. Another way to solve the problem is to realize that the calibration and flagging information that you have carefully generated during the calibration is contained in the extension tables — the raw data that you loaded is not modified until you finally **SPLIT** out the individual sources. Consequently, you can write create a dummy *uv* database to which all the extension tables are attached with the task **TASAV**, then save this “database” on tape with **FITTP**. The raw visibilities can be saved in the form of copies of the archive tapes.

F.6 Additional recipes

F.6.1 Banana stuffing

1. Pare and rub 4 **bananas** through a sieve into bowl.
2. Add 1/2 grated **onion**, 1 **green pepper** chopped fine, 3 tablespoons finely chopped **parsley**, 4 slices cooked **bacon** chopped fine, 1 1/4 cups **bread crumbs**, pinch of **thyme**, 1 teaspoon **salt**, and 1 **egg**.
3. Mix thoroughly, fill 1 **chicken**, and roast in the usual manner.

F.6.2 Banana nut bread

1. Cream 1 cup **sugar** and 1/2 cup **margarine** together.
2. Add 2 **eggs**, 2 cups **flour**, 1/2 teaspoon **salt**, and 1 teaspoon **baking soda** and mix thoroughly.
3. Add 1 cup chopped **nuts** (walnuts or pecans), 3/4 cup mashed **bananas**, and, lastly, 4 teaspoons **sour milk** and mix well.
4. Put in greased loaf pan.
5. Bake in 350° F oven for 1 hour.

Z System-Dependent *AIPS* Tips

This appendix has been updated. The information contained herein should be indicative of current conditions, but not correct in detail. This is particularly true of the Charlottesville and Green Bank information.

Although *AIPS* attempts to be system independent, some aspects of its use depend inevitably on the specific site. These vary from procedural matters (*e.g.*, assignment of workstations and location of sign-up sheets, tape drives, and workstations or other terminals) to the hardware (*e.g.*, names and numbers of workstations and tape and disk drives) to the peculiar (*e.g.*, the response of the computer to specific keys on the terminal, the presence of useful job control procedures). This appendix contains information specific to the NRAO's individual *AIPS* installations. It is intended that non-NRAO installations replace this appendix with one describing their own procedures, perhaps using this version as a template. The general description of using *AIPS* on workstations was given in Chapter 2 and will not be repeated here.

Within the NRAO, *AIPS* is installed on two main architectures — Linux PCs and Mac workstations. All our old SUN, IBM RS/6000, CONVEX C-1 and DEC VAX 11/750 and 11/780 systems have been decommissioned. Currently the fastest *AIPS* machines in the Observatory are the whichever PC was bought last.

Z.1 NRAO workstations — general information

All NRAO workstations run some version of the Unix operating system, Linux on PCs and Mac OS/X on Apple products. Unix systems are intrinsically sensitive to the difference between upper and lower case. Be sure to use the case indicated in the comments and advice given in the following notes. *AIPS* itself is case-insensitive, however; conversion of lower-case characters to upper-case occurs automatically. (Unix systems have a variety of characters for the prompt at monitor (job-control) level, and allow users to set their own as well. We will use \$ as the prompt in the text below.)

Z.1.1 The “midnight” jobs

The versions of *AIPS* on all NRAO PC systems are kept up to date continually with the master versions on the Socorro Linux PC called dave. This is achieved by automated jobs that start running at very antisocial hours of the early morning. Any changes formally made to the TST version of *AIPS* are copied to the relevant computers and recompiled/relinked. Midnight jobs run in Charlottesville, Socorro, Green Bank, and many other sites around the world.

Z.1.2 Generating color hard copy

Z.1.2.1 Color printers

Color printers are, these days, simply printers that understand the color extensions to the PostScript language used to describe plots. The NRAO owns several Tektronix color printers, two public ones in Charlottesville (ps1color in the *AIPS* Cage and ps3color in the Library) and two at the AOC in Socorro (ps213c on the ground floor and ps324c upstairs in the former library). You may display your PostScript file on the printer in Socorro simply by typing

\$ lpr -Pps324c *filename* C_R where *filename* is the name of your file.

The paper size is 8.5×11 inches, which is the default for *AIPS* tasks [TVCPS](#) and [LWPLA](#). To have the file printed on transparency paper use queue `ps1tektran` rather than `aoc324c/trans`. Full control over this complex printer is available with the `multiprint` command; type `multiprint --help` C_R for information. If you do not wish to save the plot as a disk file, you may also print it directly from within *AIPS*. The color printer is one of the printer choices when you start up *AIPS*, but you probably want to select a regular PostScript printer as your default printer. You can change your printer selection with the verb [PRINTER](#); use `PRINTER 999` C_R to see what your choices are and then `PRINTER n` C_R to choose the printer numbered n . *AIPS* print routines will re-direct PostScript files that actually contain color commands to the first PS-CMYK printer in the list, but will not re-direct ordinary print jobs to some printer other than a color printer.

Z.1.2.2 Software to copy your screen

To obtain a color hard-copy of what is on your screen, there are several software options you can choose. These include [TVCPS](#), `xv`, and `import`. Having created a PostScript (or other format) file, you can print it on color printers at the NRAO or copy the file via e-mail, `scp`, or `ftp` to some other site.

The [TVCPS](#) task in *AIPS* will create a color Encapsulated PostScript file from whatever is displayed on the *AIPS* TV server ([XAS](#)). If you use the `OUTFILE` adverb, this file is saved with whatever name you specify (see § 3.10.1). If you specify a black-and-white output to [TVCPS](#), then the output can be sent to any PostScript printer. Color PostScript must be sent to a color printer. You can, of course, edit the saved file (if you are a PostScript wizard or use [HELP POSTSCRIPT](#)) and can insert the file (since it is encapsulated) in another document.

The `xv` program is a Unix utility program available on most systems at the NRAO. It is mainly intended for image display of GIF, JPG, TIFF, and other format files. When you start `xv`, click the right button mouse anywhere in the `xv` window to bring up the control window. One of its features is a screen grab which is controlled by the “Grab” button in the lower right corner of the control window. *Before* you press this, arrange your windows and icons so that you can see exactly what it is you want to grab (e.g., the [XAS](#) server). Now press the “grab” button. A window with instructions will appear. Move the cursor to the top left of the area you want to grab. Then press and hold down the center mouse button, and drag the mouse cursor until it is at the bottom right of the area you want to grab. As you do this, you will see a box pattern on the screen outlining the area selected. Once you are done selecting the area, release the mouse cursor. When `xv` has finished grabbing the screen, whatever you grabbed will appear in the main `xv` window. You can now use the “save” button of the control window to save this as any format you want. One nice feature of this is the “save as Postscript” option. It allows you to scale, rotate, and position the image in relation to the page. Its user interface is better than most image utilities.

Finally, the `import` program provides similar functionality to the “grab” feature of `xv`, with many options about output formats and much more. Enter `import -help` C_R for a summary of the options. For example, enter `import -quality 100 outfile.jpg` C_R . The cursor will change to a plus sign. Position it at the top left corner of the area you wish to grab, hold down a mouse button, and drag the cursor to the bottom right of the area you wish to capture. When you let up on the button, the file `outfile.jpg` will be written in jpg format. The file extension determines the format, so `.eps` will produce encapsulated PostScript.

Z.1.3 Using the tape drives

For a general discussion of magnetic tapes, including the *required* software mount, see § 3.9. The following describes how to deal with the individual tape drives at the AOC.

Z.1.3.1 Mounting tapes on Exabyte and DAT drives

There are still tape devices in each of the *AIPS* Caiges: rooms 209, 211, and 257. Be aware that they are not much used and are not maintained on a regular basis. You may have to try more than one to find a device that works for your tape. There are “toaster” disk devices on every public workstation. These are more reliable and have much more storage capacity, so tape should never be a medium of choice now. The devices are kept to allow old data tapes to be read.

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

On most drives, there will be a single button on the front panel of the device somewhere. When the device becomes available, press this button to open the door. If there was already a tape in the drive, it will be ejected after some whirring and clanking and a few seconds. If a tape is ejected, remove it. Now put your tape in the drive, label facing upwards. On Exabytes, push the door closed gently. For DAT drives, lightly push the tape into the drive until the device “grabs” the tape and pulls it in the rest of the way. Exabyte and DAT tapes have a small slide in the edge of the tape which faces out which takes the place of the write ring of 9-track tapes. For 8mm (Exabyte) tapes push the slide to the right (color black shows) for writing and to the left (red or white shows) for reading. With 4mm DAT tapes, the slide also goes to the right for writing (but white or red shows) and to the left for reading (black shows).

It is necessary to wait until the mechanism in the drive has “settled down”, *i.e.*, when the noises and flashing lights have stopped, before you can access the drive. The first access is, of course, the software [MOUNT](#) command from inside AIPS.

Z.1.4 Gripe, gripe, gripe, . . .

The so-called “designated AIP” is now Eric Greisen at all times (except when he is on vacation). He assists local and remote users with their *AIPS* problems, providing quick advice or simple fixes to bugs. More complex problems may require some time and may even require receipt of the user’s data in order to debug the problem. Contact the designated AIP (and all members of the group) at the e-mail address daip@nrao.edu. The “my.nrao” web portal lets you into the “helpdesk” which has an *AIPS* department. Management prefers if you use this approach to request assistance.

Suggestions and complaints entered on all computers with the [GRIPE](#) verb (see § 11.1) are sent immediately by e-mail to daip and thereby to all members of the group. All traffic via daip from 2000 to the present is archived at listmgr/cv.nrao.edu/pipermail/daip and is well-known to Google. You might try Googling your *AIPS* error message to see if you get something useful. We stand willing, and are now able, to respond to user problems and requests on a timely basis.

The Grips database described in *AIPS* Memo No. 88 used to be maintained, but appears to have disappeared.

Z.1.5 Solving problems at the NRAO

Below are details specific to NRAO systems for handling some of the problems which may arise in AIPS.

Z.1.5.1 Booting the workstations

Modern workstations, especially the powerful PCs and Macs, are complex Unix systems which may have remote users within the NRAO and guests from elsewhere on the Internet. Users should *never* attempt to boot the system on their own. If the machine appears to be dead, find or call one of the people listed on the bulletin boards in the AIPS Cage for this purpose.

Z.1.5.2 Printout fails to appear

Check the AIPS output messages that appeared shortly after you submitted your print job, whether it be from PRTMSG or LWPLA, or some other task. You should see the output of the Unix command to show the printer queue status. If anything went wrong with the print submission, an error message should be obvious. If not, check the output of the lpq command, see what print queue was involved, and check it again from the Unix command level (not from inside AIPS).

AIPS will delete spooled files about 5 minutes after they are submitted. If the print queue is stalled (due, say, to a jammed printer) or backed up with a lot of jobs, it is possible that the file was deleted before it was gobbled up by the print spooler. This time delay has been made a locally-controlled parameter, so it is possible to set it to values higher than 5 minutes.

Finally, check to see if the printout was (a) diverted to the “big” printer (psnet in room 213 at the AOC or ps3dup in the Charlottesville library) because it was too long for the smaller printers, (b) you forgot which printer you had selected on aips startup, or, at the AOC, (c) someone has taken the output and filed it in the “today” file bin (at the AOC this is on the left side of the post directly behind the psnet printer).

Z.1.5.3 Stopping excess printout

To find out what jobs are in the spooling queue for the relevant printer, type, at the monitor level:

\$ lpq C _R	to list default print queue
\$ lpstat C _R	to list default print queue under Solaris

or to display a specific queue

\$ lpq -P _{PPP} C _R	to show printer <i>PPP</i>
\$ lpstat <i>PPP</i> C _R	to show printer <i>PPP</i> under Solaris

where *PPP* might be psnet at the AOC or ps3dup in Charlottesville. If the file is still in the queue as job number *nn*, you can type simply

\$ lprm -P _{PPP} <i>nn</i> C _R	to remove the job
\$ cancel <i>nn</i> C _R	to remove the job under Solaris

lprm and *cancel* will announce the names of any files that they remove and are silent if there are no jobs in the queue which match the request.

Unfortunately, it is now very difficult to stop long print jobs. The large memories of modern printers mean that more than one print job can already be resident in the printer while your long unwanted job is being

printed. Therefore, turning off the printer is not an option. Try to be more careful and not generate excess printout in the first place (save a tree).

A nice option available for most *AIPS* print tasks or verbs is adverb **OUTPRINT** which allows you to divert the output to a text file. Then you can use an editor like *emacs* to examine the file in detail before printing. The Unix command `wc -l file` will count the number of lines in a text file called `file` for you; note that `-l` is the letter ell, not the number one. *AIPS* provides a “filter” program to convert plain (or Fortran) text files to PostScript for printing on PostScript printers. The command

```
$ F2PS -nn <file | lpr -Pppp
```

will print text file `file` on PostScript printer `ppp`. The parameter `nn` is the number of lines per page used inside *AIPS*; it is likely to be 97 if direct printing comes out in “portrait” form or 61 if the direct print outs come out in “landscape” form.

Z.1.5.4 CTRL Z problems

The last process placed in the background via CTRL Z can be brought back to the foreground by typing `fg C_R` in response to the monitor level % or \$ (or whatever) prompt Alternatively, the user can type `jobs C_R`, which displays all background processes associated with the current login and can bring a specific process to the foreground by typing `fg % m C_R`, where `m` is the job number as displayed by the `jobs` command as [m]. For example, if a user initiated his *AIPSn* by typing `aips new pr=4 C_R` and:

<code>^Z</code>	CTRL Z typed by accident (or intentionally).
<code>Stopped</code>	<code>aips new</code> is put in the background as “stopped” and user is returned to the Unix level.
<code>\$ jobs C_R</code>	to display status of background jobs.
<code>[1] + Stopped aips new</code>	info from Unix, where [1] means job 1, “Stopped” is job 1’s state and “aips new” is the command used to start up job 1.
<code>\$ fg m C_R</code>	to return job <code>m</code> to the foreground.
<code>aips new</code>	appears on the screen just to tell the user to which job he is talking (<i>i.e.</i> , it does <i>not</i> re-execute <code>aips new</code>). You should now be talking to your <i>AIPSn</i> again.
<code>C_R</code>	to get <i>AIPSn</i> > prompt.

Z.1.5.5 “File system is full” message

The message `write failed, file system is full` will appear when the search for scratch space encounters a disk or disks without enough space. This is only a problem when none of the disks available for scratch files has enough space, at which point the task will shut down. Use the **BADDISK** adverb to avoid disks with little available space.

Z.1.5.6 Tapes won’t mount

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

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

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

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

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

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

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

```
ZVTP02 connect (INET): Connection refused
AIPS 1: ZMOUNR: UNABLE TO OPEN SOCKET TO REMOTE MACHINE, ERROR 1
AIPS 1: ZMOUNT: ERROR 1 RETURNED BY ZMOUN2/ZMOUNR
```

then the tape daemons are not running on the remote machine. Log into the remote machine and type:

```
/AIPS/START_TPSERVERS
```

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

Z.1.5.7 I can't use my data disk!

If at some point during your work you find you are prevented from reading or writing files on a data disk, it could be that your *AIPS* number does not have access to that area. If you encounter the message:

```
AIPS 2: CATOPN: ACCESS DENIED TO DISK 8 FOR USER 1783
```

it means that user 1783 has not been given access to write (or read) on disk 8. This can be seen, in the *AIPS* session, by typing *FREESPAC* to list the mounted disks. If you see a data disk listed with an access of Not you, it means your *AIPS* number has not been enabled for that disk. If you feel that you should have access to that particular disk, see the data analysts (at the AOC) or an *AIPS* Manager about enabling your user number.

Z.2 *AIPS* at the NRAO AOC in Socorro

A small visitor facility for *AIPS* users is available in both Charlottesville and Green Bank. However, most non-NRAO people wishing to reduce data at an NRAO facility with *AIPS* should consider coming to Socorro. Not only is the green chili good, but the user facilities are quite substantial.

Public workstations at the AOC are Linux computers, many being two processor, 8 core machines with 24 Gbyte of RAM and a lot (4 Terabytes) of disk. A complete list of the public-use workstations can be found on the web at <http://www.aoc.nrao.edu/computing/workstations.shtml>. All are equipped with a toaster disk and each *AIPS* Caige room has one set of tape (DAT, Exabyte) devices.

Of the several printers available at the AOC, most *AIPS* users will want to use the high-volume PostScript printer *aoc213* and the Tektronix color printer *aoc213c*, both of which are located in Room 213.

Z.2.1 Reserving public-use workstations at the AOC

If you are visiting the AOC for observing or to reduce data then you should fill out a visitor reservation form at least two weeks before you expect to arrive. These forms can be found on the web at <http://www.aoc.nrao.edu/>, select “Facilities”, then “[VLA](#)”, then “Other Info for Observers”, the “Visiting the DSOC/[VLA](#)”, and follow instructions from there. If you check “OBSERVING AND/OR [DATA REDUCTION](#)” as your purpose of visit at the end of Section A of the reservation form, then you will automatically be assigned an observer login account and a workstation. If you have any special data reduction needs, you should make a note of them in the comments section of the form.

If you are unable to fill out this form using the web or if you have any questions about reservations, you should contact AOC reservations at 575-835-7357 or [nmreserv nrao.edu](mailto:nmreserv@nrao.edu). Note that reservations for VLBA projects will not be accepted unless the observer has been notified that correlation is complete and that the data have been scrutinized. If you have any special data reduction needs then you should make a note of them in the Section C of the form.

If you are an AOC staff member and wish to sign up for a public-use workstation then you should contact helpdesk@nrao.edu or visit real people in Room 262. You are expected to use your own login account on the public workstations. In general, visitor’s reservations take priority over reservations for AOC staff.

Z.2.2 Using Linux workstations at the AOC — introduction

In order to start *AIPS* you need to log in to your assigned workstation and open a terminal window. At the AOC, we generally use the windowing environments supplied by the operating system vendors (RedHat Linux) rather than imposing a lowest-common-denominator “standard” environment. This means that the procedures for logging on and manipulating windows depend on the current version of Linux, which will progress with time.

On sitting down at a workstation you will be presented with a log-in window that has a space for you to type your user name. Enter your assigned username and press return and then, when prompted, enter your password and press return again.

You will be presented with a rather vanilla X-Windows screen. Figure out how to start a terminal window of some sort, `xterms` are preferred.

Z.2.3 Starting *AIPS*

Once you have logged in to your workstation and have a terminal emulator window open, you may start *AIPS*. If you wish to run *AIPS* on the machine at which you are sitting we recommend that you start *AIPS* with the following command.

```
aips tst pr=3
```

This will bypass the printer menu and select the high-volume PostScript printer `aoc213/dup` in Room 213 as the default *AIPS* printer.

Note that *AIPS* is configured to start a separate message window at the AOC. If you do not want the separate message window to appear, issue one of the following commands before starting *AIPS*.

```
export AIPS_MSG_EMULATOR=none
```

If you are using the KornShell or bash or

```
setenv AIPS_MSG_EMULATOR none
```

if you are using csh or tcsh. You can move the cursor into the message window and hit CTRL-Z to cause it to disappear instead.

You may replace `xterm` with the name of another terminal emulator (e.g. `dtterm`) if you wish and you can also turn off the Tektronix emulator using one of the following commands.

```
export AIPS_TEK_EMULATOR=none
```

If you are using the KornShell or bash,

```
setenv AIPS_TEK_EMULATOR none
```

if you are using csh or tcsh. This emulator is easily ignored.

If the *AIPS* TV is already running, you may use the following to avoid the messages that are produced as *AIPS* figures out whether the TV is already running.

```
aips tst pr=4 tvok
```

Z.2.3.1 Starting *AIPS* on another machine

If you are going to run *AIPS* on a remote machine and use your local workstation as a display, we recommend that you start *AIPS* on your local workstation first and allow it to start the TV. You can then iconize the terminal that you are using to run *AIPS* and open a new terminal window. You should then use the `slogin` command to log in to the remote machine. You should then start *AIPS* using the following command.

```
aips tst pr=4 tv=mydisplay tvok
```

You should replace `mydisplay` with the name of the workstation at which you are sitting.

Z.2.3.2 On-line `FILLM`

On-line `FILLM` applied to the old [VLA](#) and does not apply now.

Z.3 *AIPS at the NRAO in Charlottesville and Green Bank*

AIPS is installed in Charlottesville based on the Linux computer named `valen`. Similarly, it is installed in Green Bank on a computer named `bratac`. The installations are current, with Midnight Jobs updating the development version each night. However, there are no current members of the *AIPS* group at either site, so the actual state of *AIPS* usage, particularly for visitors, is hard to determine from Socorro. Both sites have visitor procedures and welcome outside visitors. However, Charlottesville is concerned with ALMA users wishing to use CASA and Green Bank is concerned with GBT users needing to use GBT-IDL. Visitor information for both sites can be found from the main NRAO web site, including forms to register a visit and much information. Visitors to Charlottesville and Green Bank should inquire locally for information concerning public workstations supporting *AIPS*.

G Glossary

adverb — See *POPS symbols*.

AIPS monitor — a computer terminal (perhaps lacking a keyboard) whose CRT screen is used in AIPS solely for the display of information related to the progress of the execution of the AIPS tasks. (Except, at those AIPS sites without a terminal dedicated to this use, the AIPS user's interactive terminal is used for dual purposes—i.e., to serve as the AIPS monitor as well.) Many of the messages which the AIPS tasks write to the monitor also are recorded in the *message file* (*q.v.*).

aliased response — in a radio interferometer map, a spurious feature due to a source—or to a sidelobe—that lies outside of the field of view. Consider the sampling of a visibility function V at the lattice points of a rectangular grid as multiplication of V by the comb-like distribution $R(u, v) = \sum_k \sum_l \delta(u - k\Delta u, v - l\Delta v)$. The Fourier transform \widehat{RV} of RV is given by the convolution $\widehat{R} * \widehat{V}$. Since \widehat{R} is again a comb-like distribution, with peaks, or teeth, separated by $\frac{1}{\Delta u}$ in one direction and by $\frac{1}{\Delta v}$ in the perpendicular direction, \widehat{RV} is periodic, and, about the position of each tooth in the comb, it looks like an infinite summation of rectangular pieces of \widehat{V} , each of size $\frac{1}{\Delta u} \times \frac{1}{\Delta v}$, taken from all over the plane. Aliased responses can be suppressed very effectively, by judicious choice of the *gridding convolution function* (*q.v.*).

For a more complete discussion, see Dick Sramek and Fred Schwab's Lecture No. 6 in the *Third NRAO Synthesis Imaging Summer School*. Also see [VLA Scientific Memoranda Nos. 129 and 131](#).

aliasing — in spectral analysis, error which is due to undersampling: one may wish to sample a signal that is known to be bandlimited, but whose bandwidth may not be known a priori. The Fourier transform of *Shannon's series* is periodic; aliasing error is of the form of an overlapping, or superposition, of these "replicated" spectra. See *Nyquist sampling rate* and *aliased response*.

ALU — (Arithmetic Logic Unit) an (optional) microcomputer CPU unit within the I²S TV display device which allows simple arithmetic operations, such as sums, products, and convolutions, to be performed on the data recorded in the I²S image planes. At present, AIPS makes little use of the ALU, since many of its features are unique to the I²S display unit. See I²S.

antenna file — in AIPS, an *extension file*, associated with a *u-v data file*, in which a list of the interferometer antenna positions is stored.

antenna/i.f. gain — Many of the systematic errors affecting radio interferometer measurements are multiplicative in the visibility amplitude and additive in the visibility phase, and are ascribable to individual antenna elements and their associated i.f./l.o. chains. For each antenna/i.f. these sources of error may be lumped together into a complex-valued function of time, $g(t)$, called the *antenna/i.f. gain*. Then, the visibility measurement obtained on the $i-j$ baseline at time t is given by $\tilde{V}(u_{ij}(t), v_{ij}(t)) = g_i(t)\bar{g}_j(t)V(u_{ij}(t), v_{ij}(t)) + \epsilon_{ij}(t)$, where V is the true source visibility and where the spatial frequency coordinates (u, v) have been parametrized by time. $g_i\bar{g}_j$ is the systematic "calibration error", and ϵ_{ij} , an additive error component, is assumed to be random and well-behaved.

(Another type of systematic error, the *instrumental polarization* (*q.v.*), is not included in the g_k , and always must be corrected, by proper calibration, in order to interpret polarization data.)

Some of the most serious sources of error—including atmospheric attenuation, error arising from variations in the atmospheric path length, clock error, and error in the baseline determination—conform fairly well to this multiplicative model. This model relation is exploited heavily by the *self-calibration algorithm* (*q.v.*). Compare *antenna/i.f. phase*, and see *isoplanarity assumption* and *correlator offset*.

antenna/i.f. phase — The *antenna/i.f. phase* for antenna k of an interferometer array is given by the argument (or phase) of the *antenna/i.f. gain* g_k : $\psi_k(t) = \arg g_k(t)$. Often in *self-calibration* one assumes that no amplitude errors are present and solves only for the ψ_k .

antenna residual delay — See *residual delay* and *global fringe fitting algorithm*.

antenna residual fringe rate — See *residual fringe rate* and *global fringe fitting algorithm*.

AP — See *array processor*.

AP-120B array processor — an *array processor* manufactured by Floating Point Systems, Inc., and used at a number of AIPS sites. Its floating-point word length is 38-bits. Typically it is equipped with a main data memory of 32–64 kilowords and a program source memory of 2048 words. With both a pipeline multiplier and a pipeline adder, and a memory cycle time of 167 ns., when programmed at top efficiency it can perform at an arithmetic rate of 12 million floating-point operations per second.

The AP-120B is no longer in production; this product has been superseded by the 5000 series product line. Though the AP-5000's are used at some AIPS sites, their advanced features are not used by AIPS—only those features which are shared by the older model are fully exploited by AIPS tasks.

array processor — a computer peripheral attachment which is capable of performing certain floating-point computations, especially vector and matrix operations, at high speed, and independently of the *host computer* central processing unit. Usually the high-speed performance is achieved by a technique known as pipelining. The basic arithmetic operations of addition and multiplication are performed in stages, by a so-called pipeline adder and a pipeline multiplier. These units operate just like an assembly line in a manufacturing plant. Some array processors (AP's) are constructed with multiple pipelines. Address computations are performed concurrently with the arithmetic operations, by a unit which is separate from the pipelines. The algorithms best-suited to an array processor implementation are those which can be structured so as to keep the pipelines filled a fair fraction of the time. Most AP's have their own high-speed data memory, but some are parasitic on the memory of the host computer. Portions of many AIPS tasks have been programmed for the Floating Systems, Inc. model *AP-120B array processor*, (*q.v.*). Also see *array processor microcode*, *Q-routine*, and *pseudo-array processor*.

array processor microcode — program source code written in the assembly language of an *array processor*, (*q.v.*). Array processor (AP) manufacturers usually provide an extensive library of utility subroutines that may be called from a

G. Glossary

high-level programming language, such as Fortran; however, some computationally-intensive algorithms cannot be easily or efficiently implemented using only these libraries. Portions of these algorithms must be written in microcode—a painstaking process. The assembly languages of different models of AP's differ considerably (as do the subroutine libraries, too, in fact) because of differences in the hardware architectures. Thus the AIPS programming group tries to avoid writing microcode. But portions of the AIPS tasks for mapmaking, deconvolution, and self-calibration are written in AP microcode. Also see *Q-routine*.

associated file — In AIPS, any two or more files among a collection consisting of a *primary data file* and all of its *extension files* are termed *associated*.

auto re-boot — a boot initiated by the computer itself, of its own volition. See *boot*.

back-up — The act of copying the contents of a computer file to some permanent storage medium such as magnetic tape or punched cards, for the purpose of protecting against accidental loss or in order to liberate storage space (e.g., disk space), is termed *backing-up*. The new copy of the file is termed a *back-up* copy, or simply a *back-up*. See *scratch*.

bandwidth smearing — in a radio interferometer map, space-variance of the *point spread function* which is attributable to non-monochromaticity, or finite bandwidth. The point spread function—at a particular point in a map—taking into account bandwidth smearing, but ignoring other instrumental effects, is termed a *delay beam*. Bandwidth smearing is a radial effect: the delay beams become more elongated, in the radial direction from the interferometer *phase tracking center*, as their distance from the phase tracking center increases. The delay beams are easily calculable when all of the receivers in an array have identical, and known, i.f. passbands. E.g., with rectangular passbands of width $\Delta\nu$, and observations centered at a frequency ν_0 , the measured visibility amplitude of a point source is proportional to $\frac{\sin\gamma}{\gamma}$ —where $\gamma \equiv \pi(ux + vy + wz)\frac{\Delta\nu}{\nu_0}$, (u, v, w) denotes the spatial frequency coordinates, measured in wavelengths at ν_0 , and (x, y, z) denotes the direction cosines of the location of the point source, with respect to the phase tracking center. For more details, see Alan Bridle and Fred Schwab's Lecture No. 13 and Bill Cotton's Lecture No. 12 in the *Third NRAO Synthesis Imaging Summer School* and see [VLA Scientific Memo](#). No. 137.

Bandwidth smearing can, in principle, be eliminated (assuming that the bandpasses are known) by applying an image reconstruction algorithm which has a knowledge of the smearing mechanism; that is, by an algorithm which is more general than the usual deconvolution algorithms—see *image reconstruction*. The most common method for reducing bandwidth smearing is the technique of *bandwidth synthesis*, (q.v.).

bandwidth synthesis — a technique of radio interferometry which is intended to diminish the effect of *bandwidth smearing*. Bandwidth synthesis observing is very similar to spectral-line mode observing: the i.f. bandpasses are split up into a number of pieces, or channels, and the data in each channel are treated separately up until the mapping/deconvolution stage of processing. At that stage, the problem can be formulated as a system of simultaneous convolution equations: one has the system $g_1 = b_1 * f + \epsilon_1, \dots, g_n = b_n * f + \epsilon_n$, where n is the number of frequency channels, g_k is the *dirty map* for channel k , b_k the *dirty beam* for that channel, f the unknown radio source brightness distribution (here assuming that f is not a function of frequency), and ϵ_k is noise (were it not for the noise, and for the fact that each deconvolution problem is ill-posed—in its own right—, there would be no reason to treat the equations simultaneously, or even to consider more than a single one of them). (For a description of a refinement

to the bandwidth synthesis technique, for sources with spatially-varying spectral indices, see *broadband mapping technique*.) Note that all the b_k are identical, apart from a dilation factor; i.e., as the *u-v coverage* “shrinks”, toward the low end of the observing band, the b_k dilate by the reciprocal of the *u-v* shrinkage factor.

The present state of software development does not allow solving the problem in quite the way it is formulated above. Rather, some mapping/deconvolution algorithm is applied separately to each of the channels, and the resulting maps are averaged.

baseline-time order — An ordered set of visibility measurements $\{V_{ij}(t_k) \mid 1 \leq i < j \leq n, k = 1, \dots, l\}$ recorded with an n element interferometer at times t_k is said to be in *baseline-time order* if the ordering is such that all of the data for the 1-2 baseline, sorted by time, occur first, followed by the data for the 1-3 baseline, again sorted by time, etc., etc. (This canonical ordering by baseline is the order $V_{12}, V_{13}, \dots, V_{1n}, V_{23}, \dots, \dots, V_{n-1,n}$.) Compare *time-baseline order*.

Baseline-time ordering of a *u-v data file* is convenient for purposes of data display.

batch editor — a *text editor* within the AIPS program which allows the user to prepare *batch jobs* (q.v.), to be run non-interactively.

batch job — AIPS may be run either interactively—allowing the user to make ‘split-second’ decisions—or in batch mode. In batch mode, the user first decides on a set strategy for reducing the data, and then, using the special AIPS *batch editor*, the user prepares a *text file*, containing those AIPS commands which are appropriate to the anticipated data reduction needs. The batch job is placed in a *batch queue*, and the job steps are executed by the *batch processor*, in a non-interactive mode.

batch processor — the server, or scheduler, for *batch jobs* (q.v.). The AIPS batch processor follows certain rules in scheduling: batch jobs requiring the use of an array processor (AP) often are scheduled to run only during nighttime hours; the processor serving one of the *batch queues* might refuse service, altogether, to a job requiring an AP; and batch jobs may be given lower priority than those AIPS tasks which are run interactively.

batch queue — a waiting line for *batch jobs*. The AIPS batch queue is a single-server queue—i.e., the server (the *batch processor*) initiates the execution of the jobs one after the other, rather than in parallel. However, AIPS can be configured with more than one batch queue, each with its own batch processor; this number varies according to site.

“battery-powered” Clean algorithm — a modified version of the Clark Clean algorithm, devised by Fred Schwab and Bill Cotton. At each major cycle of the algorithm, or perhaps less frequently, the residual map is computed not by convolving the current iterate with the dirty beam map, but rather by computing the visibility residuals, and then re-gridding and re-mapping. By this means, the edge effects are compensated, and hence one can search the full dirty map field of view for Clean components. Simultaneously, instrumental effects (finite bandwidth and finite integration time) and sky curvature (the wz term) can be compensated for (i.e., the algorithm solves a more general equation than a convolution equation). See *Clark Clean algorithm*.

A “mosaicing” version of this algorithm is implemented in the AIPS task [MX](#). The deconvolved image is defined over some number $1 \leq n \leq 16$ of rectangular patches. Within each patch, the data are corrected for sky curvature, by the correction appropriate to the center of the patch. Instrumental corrections are not included, at present.

beam — 1. in radio interferometry, the inverse Fourier transform (FT^{-1}) of the *u-v sampling distribution*, or FT^{-1} of a weighted *u-v* sampling distribution, possibly convolved with a gridding convolution function—the idealized response to a point, or unresolved, radio source. 2. a numerical approximation to 1. 3. a digitized version of 2, sampled on a regular grid (usually regarded as a map or image). 4. \approx *point spread function*, q.v. 5. (occasionally) as above, but taking into account instrumental effects, so that the beam depends on position in the sky. See *dirty map*.

Occasionally, any one of the above, other than 5, is termed the *synthesized beam*.

beam patch — in the Clark Clean algorithm, that portion of the central part of the beam which is used in the inner iterations, or the minor cycles. In the AIPS implementation, the beam patch size typically is set at 101 pixels \times 101 pixels. See *Clark Clean algorithm*.

beam squint — In radio interferometry, direction dependent, or space-variant *instrumental polarization*, which is difficult to calibrate, can arise from *beam squint*. The beam squint effect, for the usual case of a pair of (nominally) orthogonally polarized feeds on each array element, is due to differences in their power patterns—in particular, to differences in the directions of their peak response.

blanked pixel — in a digital image, a pixel whose value is undefined. In computer storage of quantized digital images, some special numeric value is assigned to the blanked pixels, so that they may be recognized as undefined and given whatever special treatment is required. See *pixel*.

BLC — *bottom left corner* (of an image). See $m \times n$ map.

blink — See *TV blink*.

boot — A computer is restarted by means of a *bootstrapping* procedure, whereby the operating system and the data management facilities are re-initialized in a succession of steps. This ritual, through which the computer gathers its wits, is termed the *boot*. A boot (\approx *re-boot*) is required after any system crash (e.g., after a power failure). Usually the sequence of steps required to accomplish the boot is posted in a notice located close to the system operating console, or on the CPU panel. On modern computers, such as the Vax, the boot procedure is highly automated. In fact, there may be an abbreviated boot procedure, termed a *quick boot*, to follow after a “soft” system crash. (On such systems, a quick boot should be attempted before resorting to a full boot.) Indeed, some systems (the Vax included) re-boot on their own initiative following a soft system crash—this is termed an *auto re-boot*.

BOT marker — (Beginning-Of-Tape marker) a short strip of metal foil attached near the front, or beginning, end of a computer magnetic tape. The tape drive uses the BOT marker in order to position the tape at its starting position.

bpi — (bits per inch) the basic unit of measurement used to specify the density at which information is recorded on a computer magnetic tape: the effective number of bits per inch per track. The standard recording densities are 800, 1600, and 6250 bpi. Modern computer tapes are nine-track tapes: eight recording tracks are used for the data, and the ninth track is used to record “parity bits” for error-checking. See *tape blocking efficiency*.

broadband mapping technique — a refinement of the radio interferometric method of *bandwidth synthesis* (q.v.), in which one solves simultaneously for the radio brightness distribution $f_{\nu_r}(x, y)$ at some reference frequency ν_r , and for the (spatially varying) spectral index $\alpha(x, y)$ across the observing band. Assuming that the observing band is split into frequency

channels centered at ν_1, \dots, ν_n , one solves the simultaneous system of convolution equations $g_1 = b_1 * f_1, \dots, g_n = b_n * f_n$, where g_k is the *dirty map* from channel k , b_k the *dirty beam* from that channel, and where f_k is given by

$$f_k(x, y) = \left(\frac{\nu_k}{\nu_r} \right)^{\alpha(x, y)} f_{\nu_r}(x, y).$$

All of the b_k are identical, apart from a dilation factor. Assuming that the frequency channels are narrow enough, one can expand the *u-v coverage* considerably, with immunity to the *bandwidth smearing* effect. Fractional bandwidths as large as 20–30% can be used, depending on the linearity of the spectral index variations.

This mapping technique is described by Tim Cornwell [Broadband mapping of sources with spatially varying spectral index, VLB Array Memo. No. 324, Feb. 1984]. Extensive modification of one of the standard deconvolution algorithms is required. The requisite modification of the *Högbom Clean algorithm* is in progress.

b-t order — See *baseline-time order*.

bug — an actual or a perceived programming error or program deficiency. The bug may be in the eye of the beholder since the program user may fancy an application similar to, but differing from, the one for which the program is intended. In AIPS there is a formal mechanism for reporting program bugs; see *gripe file* for a description.

byte — a unit of eight bits of computer storage.

carriage-return key — One of the most used keys on any computer terminal keyboard is the carriage-return key (C_R). This is the button which ordinarily must be depressed when one has finished typing a command to the computer, in order for the computer to accept or acknowledge the command.

catalog entry — an entry within an AIPS *catalog file* (“CA” file) pertaining to a particular *primary data file*.

catalog file — In AIPS, each user has, for each disk on which he has data stored, his own *catalog file*, or “CA” file—a directory of all of his primary data files which reside on that disk. The AIPS verb **CATALOG** (as do its variants **MCAT** and **UCAT**) allows the user to see a summary listing of the contents of his catalog files. See *header record*.

catalog slot — in AIPS, a numbered space reserved in a *catalog file* for the insertion of a *catalog entry*.

cell-averaging — in radio interferometer mapping, gridding convolution which is achieved simply by averaging the visibility data which lie in each *u-v* grid cell. This is equivalent to use of a *gridding convolution function* equal to the *characteristic function* of the rectangle $\{|u| < \Delta u/2, |v| < \Delta v/2\}$, where Δu and Δv denote the grid spacing—i.e., it is equivalent to the use of a so-called *pillbox* function. The Fourier transform of the pillbox gridding convolution function is proportional to a separable product of two $\frac{\sin x}{x}$ functions; this function does not decay rapidly enough to yield very effective *aliasing* suppression. The zero-order *spheroidal functions* offer much better aliasing suppression, at somewhat increased computational expense (equivalent to averaging the data over a region 36 times larger, in the case of the default gridding convolution function used by the AIPS mapping tasks).

cellsize — in radio interferometer mapping, the size $\Delta u \times \Delta v$ of the *u-v* grid cells. Ordinarily, the visibility data are smoothed by an appropriate *gridding convolution function* and this convolution then is sampled at the coordinate locations of the centers of the grid cells. After appropriate weighting, the *discrete Fourier transform* yields the *dirty map*. Δu and Δv are chosen according to *Shannon's sampling theorem*: if the size of the dirty map is x radians by y radians, then $\Delta u = \frac{1}{x}$ wavelengths and $\Delta v = \frac{1}{y}$ wavelengths.

G. Glossary

cereal bowl map defect — same as *negative bowl artifact*.
See *zero-spacing flux*.

characteristic function — The characteristic function χ_A of a set $A \subset X$ is defined for all $x \in X$ by the formula

$$\chi_A(x) = \begin{cases} 1, & \text{if } x \in A, \\ 0, & \text{if } x \notin A \end{cases}$$

(χ_A is also called the *indicator function* of A , and the notations c_A and 1_A commonly are used in lieu of χ_A .) Note that this usage of the term, which is standard in mathematical analysis, differs from its usage in probability and statistics, where it refers to the Fourier transform of a probability measure (i.e., to the FT of the distribution function of a random variable).

chromaticity — in visual perception, essentially the dominant wavelength and the purity of the spectral distribution of light, as perceived. *Hue* and *saturation* determine the chromaticity, which is independent of *intensity*. See *C.I.E. chromaticity diagram*.

C.I.E. chromaticity diagram — a two-dimensional diagram devised in 1931 by the Commission Internationale de l'Eclairage (International Commission on Illumination) to show the range of perceivable colors as a function of normalized chromaticity coordinates (x, y), under standardized viewing conditions. The color, for an additive mixture of monochromatic red, green, and blue (R, G, B denoting the intensities at 650, 520, and 380 nm. wavelengths) as perceived by a ‘standard observer’, is displayed in this diagram as a function of the normalized *chromaticity coordinates* $x = R/(R + G + B)$ and $y = G/(R + G + B)$.

Other chromaticity diagrams can be drawn for different choices of primary hues, for mixtures of nonmonochromatic light, or for ‘nonstandard observers’. In digital imagery, such a diagram may be tailored to a particular color image display unit. See [G. S. Shostak, *Color basics—a tutorial*. In R. Albrecht and M. Capaccioli, I.A.U. Astronomical Image Processing Circular No. 9, Space Telescope Science Institute, Jan. 1983] and [G. Wyszecki and W. S. Stiles, *Color Science*, Wiley, New York, 1967], a comprehensive textbook on colorimetry.

Clark Clean algorithm — a modified version of the Högbom Clean algorithm, devised by Barry Clark in order to accomplish an efficient *array processor* implementation of Clean (see [B. G. Clark, An efficient implementation of the algorithm Clean, *Astron. Astrophys.*, **89** (1980) 377–378]). To operate on, say, an $n \times n$ map, the original Clean algorithm requires on the order of n^2 arithmetic operations at each iteration, and typically there may be hundreds or thousands of iterations. The Clark algorithm proceeds by operating not on the full residual map, but rather by picking out only the largest residual points, iterating on these for a while (during its *minor cycles* or inner iterations) and only occasionally (at the *major cycles*) computing the full $n \times n$ residual map, by means of the [FFT](#) algorithm. After each major cycle, it again picks out the largest residuals and goes into more minor cycles. And, for further economy, during these inner iterations the dirty beam is assumed to be identically zero outside of a relatively small box (termed the *beam patch*) which is centered about the origin. See *Högbom Clean algorithm*.

Clean — See *Högbom Clean algorithm*.

Clean beam — in the Högbom Clean algorithm, an elliptical Gaussian function h with which the final iterate is convolved, in order to diminish any spurious high spatial frequency features—also termed *restoring beam*. h is specified by its major axis (usually the FWHM), its minor axis, and the position angle on the plane of the sky of its major axis. Usually these parameters are set by fitting to the central lobe of the dirty beam. See *Högbom Clean algorithm* and *super-resolution*.

Clean box — a rectangular subregion of a *Clean window* (q.v.).

Clean component — in the Högbom Clean algorithm, a δ -function component which is added to the $(n - 1)$ st iterate in order to obtain the n th iterate. Its location is the location of the peak residual after the $(n - 1)$ st iteration, and its amplitude is a fraction μ (the *loop gain*) of the largest residual. See *Högbom Clean algorithm*.

The AIPS task implementing the (Clark) Clean algorithm stores a list of the Clean components in an extension file which is termed a *components file*.

Clean map — an approximate deconvolution of the *dirty beam* from the *dirty map*, derived by an application of the Högbom Clean algorithm or one of its derivatives. See *Högbom Clean algorithm*.

Clean speed-up factor — in the *Clark Clean algorithm*, a number α in the range $[-1, 1]$ used in determining when to end a major cycle. Smaller α causes a larger number of major cycles to occur (at greater computational expense) but yields a result closer to that of the classical *Högbom Clean algorithm*.

Clean window — in the Högbom Clean algorithm, the region A of the residual map which is searched in order to locate the *Clean components* comprising the successive approximants to the radio source brightness distribution. In the AIPS implementation, A is a union of rectangles, called *Clean boxes*, which may be specified by the user. When A is not explicitly specified, the algorithm searches over the central rectangular one-quarter area of the residual map. See *window Clean* and *Högbom Clean algorithm*.

clipping — the discarding (i.e., the *flagging*) of visibility data whose amplitudes exceed some threshold value, or the discarding of visibility data whose differences from some tentative source model are too large in amplitude. The AIPS task [CLIP](#) is used for clipping. See *u-v data flag*.

closure amplitude — Assume that the visibility observation on the $i-j$ baseline ($i < j$) is given by $\tilde{V}_{ij} = g_i \bar{g}_j V_{ij}$, where V_{ij} is the true visibility and where g_i and g_j are the *antenna/i.f. gains* (ignore any additive error). Then, for certain combinations of (at least four) baselines, one may form ratios of observed visibilities (and their conjugates)—including each visibility only once—in such a manner that the g 's cancel one another. For example, if $i < j < k < l$, then

$$\frac{\tilde{V}_{ij}\tilde{V}_{kl}}{\tilde{V}_{il}\tilde{V}_{jk}} = \frac{V_{ij}V_{kl}}{V_{il}V_{jk}}.$$

The modulus of such a ratio is termed a *closure amplitude* (and its argument, a *closure phase*).

Closure amplitude is called a “good observable”, since, under the above assumptions, it is not sensitive to measurement error. The closure amplitude and closure phase relations are exploited in the *hybrid mapping algorithm* (q.v.). Also see *self-calibration algorithm*.

closure phase — Assume that the visibility observation on the $i-j$ baseline ($i < j$) is given by $\tilde{V}_{ij} = g_i \bar{g}_j V_{ij}$, where V_{ij} is the true visibility and where g_i and g_j are the *antenna/i.f. gains* (ignore any additive error). Then, for a combination of any three or more baselines forming a closed loop, one may sum the visibility phases in such a manner that the *antenna/i.f. phases* ψ_k drop out. For example, if $i < j < k$, then $\arg \tilde{V}_{ij} + \arg \tilde{V}_{jk} - \arg \tilde{V}_{ik} = \arg V_{ij} + \psi_i - \psi_j + \arg V_{jk} + \psi_j - \psi_k - \arg V_{ik} - \psi_i + \psi_k$. Such a linear combination of observed visibility phases is termed a *closure phase*.

Closure phase is called a “good observable”, since, under the above assumptions, it is not sensitive to measurement error. The closure phase relations are exploited in the *hybrid mapping algorithm* (*q.v.*). Also see *closure amplitude* and *self-calibration algorithm*.

color contour display — a color digital image display of a real-valued function f of two real variables (x, y) , in which the color assignment (the *hue*) is a coarsely quantized function of $f(x, y)$. The visual effect of this type of *pseudo-color display*, in the case when f is continuous, is similar to the traditional sort of contour display. One sees curves along which f is constant, separated by swathes of constant hue—each hue corresponding to a distinct quantization level.

color triangle — Any three non-collinear points plotted on a chromaticity diagram determine a color triangle. Since the points are non-collinear, they correspond to basic, or *primary* hues. All of those colors on the chromaticity diagram which fall within the triangle determined by the three points may be produced by addition of the three hues. See *C.I.E. chromaticity diagram*.

compact support — See *support*.

components file — in AIPS, an extension file, associated with an image file containing a *Clean map*, whose content is a list of the positions and amplitudes of the *Clean components* included in that Clean map, as determined by the Clean algorithm. The source model specified by this list of components often is used in *self-calibration*.

conjugate symmetry — that property which characterizes a *Hermitian function* (*q.v.*). Generally an assumption of conjugate symmetry is implicit whenever one speaks of the *u-v coverage* corresponding to some radio interferometric observation.

Conrac monitor — the CRT unit of the I²S TV display device, in use at a number of AIPS installations. See I²S.

convolution theorem — This theorem is well-known, but seldom is quoted in its distributional form: for two distributions, f and g , the Fourier transform of the convolution of f and g is given by $\widehat{f * g} = \hat{f}\hat{g}$, whenever one distribution is of *compact support* and the other is a “tempered” distribution. (Loosely speaking, a tempered distribution is one which does not increase too rapidly at infinity.) See [Y. Choquet-Bruhat, C. Dewitt-Morette, and M. Dillard-Bleick, *Analysis, Manifolds, and Physics*, North-Holland, New York, 1977, ch. VI].

One ought to be aware of this form of the theorem, since often one must deal with convolution of functions that are not of compact support—*dirty beams*, *principal solutions*, *invisible distributions*, etc.—whose Fourier transforms do not exist as ordinary functions, but only as distributions or generalized functions.

Convolution of distributions, itself, is defined, in general, whenever the support of either distribution is compact, or (in one dimension) when the supports of both distributions are limited on the same side. For distributions which are absolutely integrable ordinary functions, and whose Fourier transforms possess the same property, the compact support assumption is not required here, or above. Related fact: convolution is not always associative (*i.e.*, $f * (g * h) \neq (f * g) * h$, in general), but it is associative provided that all the distributions, with the possible exception of one, are of compact support. See the above-cited reference.

convolving function — See *gridding convolution function*.

coordinate reference pixel — in an AIPS *image file*, a “pixel” whose coordinates are recorded in the *image header* together with the coordinate increments (*i.e.*, the pixel coordinate separations) that allow the physical coordinates of all other pixels in the image to be computed. This “coordinate reference pixel”

may not actually be present in the image: all that matters are its physical coordinates and its pixel coordinates (which too are recorded in the header—and which may, in fact, be fractional).

Often, in a radio map (and by default, when the standard AIPS mapmaking tasks are executed), the position of the coordinate reference pixel coincides with the map center and with the *visibility phase tracking center*. See *m × n map* and *pixel coordinates*.

correlator offset — One of the basic assumptions of much of the [VLA](#) calibration software (*e.g.*, the *self-calibration algorithm*) is that the systematic errors in the visibility measurements are multiplicative errors that are ascribable to individual array elements and their associated i.f./l.o. chains, and that—at a given instant—each such antenna-based error has an identical effect on each visibility observation involving that antenna/i.f. combination. Systematic measurement errors which do not conform to this model are called *correlator offsets* or *non-closing errors*. See *antenna/i.f. gain*.

Correlator offsets can be the limiting factor in obtaining high dynamic range [VLA](#) maps. Some observers have reported fairly large multiplicative correlator offsets which vary slowly with time and which do not appear to vary with the *phase tracking center* or with source structure. From observations of an external calibrator, one may estimate, and compensate for, such offsets. This mechanism is provided in the AIPS tasks BCAL1 and BCAL2. See [R. C. Walker, Non-closing offsets on the [VLA](#), [VLA](#) Scientific Memo. No. 152].

crash — the abrupt failure of a computer system or program. More specifically, a *system crash* is the abrupt failure of a computer—or of a computer’s operating system—causing the computer to halt the execution of programs; and a *program crash* is the abrupt failure of a computer program resulting either from a flaw in the logic of the program itself, or from some peculiar interaction with the operating system, the storage management facility, another program, or the user—or from an act of God. A *hardware crash* (*e.g.*, a *disk crash*) is a crash which results from the failure of the computer electronics or electro-mechanics, and a *software crash* is one which results from a flaw or an inadequacy in program logic, or in operating system program logic. A *soft crash* is a crash from which it is easy to recover—*i.e.*, easy to restart the computer and resume work, and a *hard crash* is the opposite.

crosshair — 1. a marker on the *TEK screen*, or *green screen*, which may be moved about through the use of thumbwheel knobs which are located on the terminal keyboard panel. The position of the crosshair may be sensed by the computer program, and thus the user may point out to the program features that are of interest in the graphical display on the CRT screen. 2. a marker with the same function as just described, but on a TV display device, and more likely controlled by a *trackball* than by thumbwheels. Same as *TV cursor*; and see *trackball*.

cube — See *data cube*.

cursor — 1. a marker on an interactive computer terminal indicating the position on the CRT screen where the next character is to be typed. 2. *TV cursor*—on a TV display device, a marker whose manually controlled position may be sensed by the computer. See *crosshair*.

data cube — 1. in [VLA](#) spectral line data analysis, a three-dimensional map or “image” representing a function of three real variables—two spatial variables representative of position in the sky, and one variable related to frequency or velocity. 2. any n -dimensional *image*, $n \geq 3$.

Computer access of a multi-dimensional data array, residing in any standard type of storage medium such as disk or magnetic tape, is sequential, as if the data were one-dimensional. Spectral

G. Glossary

line data cubes are stored plane-by-plane, row-by-row, column-by-column. Permutation of the correspondence between plane, row, and column, and the coordinate axis numbering, is referred to as *transposition* of the data cube.

database — a computer filing system, or file structure system. For example, the AIPS database consists not only of the data themselves, but also of the directories and the cross-reference lists of all the AIPS data files (including extension files), the data format definitions, etc., as well as the rules and principles governing the use thereof.

data file — on a computer storage medium, such as disk or magnetic tape, the concrete, or physically present representation of a logically distinct grouping of data in a manner permitting repeated access by computer programs.

data flag — See *u-v data flag*.

deconvolution — the numerical inversion of a convolution equation, either continuous or discrete, in one or several variables; i.e., the numerical solution (for f) of an equation of the form $f * g = h + \text{noise}$, given g and given the right-hand side of the equation. Except in trivial cases, deconvolution is an ill-posed problem: In the absence of constraints or extra side-conditions, and in the case of noiseless data—assuming that some solution exists—there usually will exist many solutions. In the case of noisy data, there usually will exist no exact solution, but a multitude of approximate solutions. In the latter case, if one is not careful in the choice of a numerical method, the computed approximate solution is likely not to have a continuous dependence on the given data. The so-called *regularization method* (*q.v.*) (of which the *maximum entropy method* is a special case) is an effective tool for the deconvolution problem.

Discrete two-dimensional deconvolution is an everyday problem in radio interferometry, owing to the fact that—under certain simplifying assumptions—the so-called *dirty map* is the convolution of the *dirty beam* with the true celestial radio image. In addition to the maximum entropy method, the *Högbom Clean algorithm* is commonly applied to this problem. See Tim Cornwell and Robert Braun's Lecture No. 8 in the *Third NRAO Synthesis Imaging Summer School*.

delay — See *residual delay*.

delay beam — in radio interferometry, the *point spread function* or *beam*, taking into account *bandwidth smearing*, but ignoring other instrumental effects. See *bandwidth smearing*.

DFT — an abbreviation for *discrete Fourier transform* and *direct Fourier transform* (*q.v.*). When used in disciplines other than radio astronomy, it usually signifies the former.

Dicomed Image Recorder (Model D47) — a computer-controlled image display device intended for photographic reproduction of digital images. The film is exposed by a cathode ray tube. The device is capable of 4096 pixel \times 4096 pixel resolution and of both black-and-white and color reproduction. The digital exposure control and eight-bit pixel input allow 256 discrete exposure levels. The CRT has a single electron gun and a screen with a white phosphor; color reproduction is accomplished by means of multiple exposures, with the insertion of red, green, and blue filters. There is a Dicomed recorder at the NRAO in Charlottesville, and another at the [VLA](#).

direct Fourier transform — a term used imprecisely in radio astronomy to mean either: 1) a finite trigonometric sum, of the form

$$\sum_{j=0}^{n-1} a_j e^{2\pi i u_j x},$$

with a_j complex, where the (real) u_j are irregularly-spaced; 2) the brute-force evaluation of such a sum; or 3), the naïve, or brute-force evaluation (using $O(n^2)$ arithmetic operations) of the (n -point) *discrete Fourier transform*.

The direct Fourier transform, in senses 1) and 2) of the definition, arises in synthesis mapping applications because of the irregular distribution of the visibility measurements. Common practice is to use a *gridding convolution function* to interpolate the data onto a regularly-spaced lattice, so that, for computational economy, the *fast Fourier transform algorithm* may be used.

dirty beam — in radio interferometry, simply a *beam*, but computed with precisely the same operations as those used to compute some companion *dirty map* (i.e., with the same *u-v* coverage, the same manner of gridding convolution, the same *u-v* weight function and taper, etc.). In Cleaning a dirty map, only the companion dirty beam should be used.

dirty map — 1. ignoring instrumental effects, the inverse Fourier transform (FT^{-1}) of the product of the visibility function V of the radio source and the (possibly *weighted* and/or *tapered*) *u-v sampling distribution* S ; i.e., FT^{-1} of the *u-v measurement distribution*. 2. a discrete approximation to 1; in this case, the product SV is convolved with some function C , of *compact support*, and an inverse discrete Fourier transform of samples of $C * (SV)$ taken over a regular grid yields the *dirty map*. 3. as in 2, but corrected for the taper (\tilde{C} , the FT^{-1} of C) induced by the convolution. 4. any of the above, but now taking into account various instrumental effects (receiver noise, non-monochromaticity or finite bandwidth, finite integration time, sky curvature, etc.).

If it is assumed that $V \equiv 1$, then the map, or point source response, so obtained is termed the *beam* (*q.v.*). Also see *gridding convolution function*, *u-v taper function*, *u-v weight function*, *dirty beam*, and *principal solution*.

discrete Fourier transform — The (one-dimensional) discrete Fourier transform (DFT) y_0, \dots, y_{n-1} of a sequence of complex numbers x_0, \dots, x_{n-1} is given by the summation

$$y_k = \sum_{j=0}^{n-1} x_j e^{2\pi i j k / n}.$$

(The multi-dimensional generalization is straightforward). The x_j are given by the *inverse DFT* of the y_k :

$$x_j = \frac{1}{n} \sum_{k=0}^{n-1} y_k e^{-2\pi i j k / n}.$$

(Frequently the forward and inverse transforms are defined in the manner opposite to that given here, and the $\frac{1}{n}$ normalization factor sometimes is moved about.) The DFT arises most naturally in numerically approximating the Fourier coefficients $c_m = \frac{1}{2\pi} \int_0^{2\pi} f(x) e^{-imx} dx$ of a 2π -periodic function f which is representable by the trigonometric series $\sum_{m=-\infty}^{\infty} c_m e^{imx}$. The *fast Fourier transform algorithm* (*q.v.*) can be used for efficient numerical evaluation of the DFT.

disk hog — a derogatory term, used to connote a computer user whose disk data files are excessively voluminous or numerous, therefore putting other computer users at a relative disadvantage. Unneeded data files should be *scratched*, or destroyed, in order to free up disk space. Large disk files which will not be needed for a time should be *backed-up* on magnetic tape and then deleted from disk.

dynamic range — a summary measure of image quality indicative of the ability to discern dim features when relatively

stronger features are present—i.e., a measure of the ability to distinguish the dim features from artifacts of the *image reconstruction* procedure (in a radio map, from remnants of the sidelobes of stronger features) and from noise. The dynamic range achievable in a radio interferometer map is determined primarily by the uniformity of the *u-v coverage*, the density and extent of the coverage, the sensitivity of the array, and the quality of the calibration.

If the true radio source brightness distribution f is known, one can define the dynamic range of a reconstruction \tilde{f} as, say, the ratio of the maximum value of $|f|$ to the r.m.s. difference between f and \tilde{f} . When f is unknown, as is usually the case, an empirical measure of the dynamic range is used—perhaps the ratio of the maximum value of $|\tilde{f}|$ to the r.m.s. level in an apparently empty region of the map, or the ratio of the strongest feature to the weakest “believable” feature—but there is no widely-accepted definition.

What one might wish to call the “true” dynamic range of a radio map is a spatially-variant quantity. The ability to discern a dim feature depends on its proximity to brighter features, because there are relatively stronger sidelobe remnants near the bright features. The quality of a map (and perhaps the dynamic range—depending on how it is defined) deteriorates away from the *phase tracking center*, because of the inability of the image reconstruction algorithms to compensate for various instrumental effects (e.g., bad pointing, *bandwidth smearing*, etc.).

EDT — a sophisticated text editor (a *screen editor*) used on the Vaxes. It makes use of the “keypad” feature of the fancier terminals. EDT can be run only on certain model terminals: on the DEC (Digital Equipment Corp.) Models VT-52 and VT-100, and on terminals such as the Visual-50’s and the Visual-100’s which are capable of emulating the DEC terminals. See *text editor*.

EMACS — a sophisticated text editor used on the Vaxes, as well as on many computers which run under the UNIX operating system. (There is also a version for the IBM-PC.) EMACS is a *screen editor*, and the one which is favored by most among those in the AIPS programming group. On terminals with the “keypad” feature, the keypad keys can be programmed by the user to perform many useful editing tasks; however, EMACS can be run from other models of terminals, as well. EMACS provides two powerful and convenient features which most other text editors do not offer: the ability to temporarily exit from the editor and “return to monitor level,” and the ability to initiate an interactive “job control session,” or initiate *sub-tasks*, in an EMACS buffer. See *text editor*.

explain file — in AIPS, a *text file* containing a detailed explanation of a particular AIPS *task* or *verb*, often including hints, suggested applications, algorithmic details, and bibliographical references. Issuing the AIPS verb **EXPLAIN** causes the contents of an explain file to be printed on the terminal screen or on a line printer. Compare *help file*.

EXPORT format — a visibility data magnetic tape format for transport of **VLA** data from the DEC-10 computer or the online computer at the **VLA**.

EXPORT tape — a magnetic tape containing data recorded in the *EXPORT format*.

exp × sinc function — a useful gridding convolution function: same as the *Gaussian-tapered sinc function* (*q.v.*), except that the exponent of the argument to the exponential function may be other than two.

extension file — in AIPS, a data file containing data supplemental to those contained in a *primary data file* (either a *u-v data file* or an *image file*). Whenever a primary data file is deleted by the standard mechanism within AIPS for file destruction, all

extension files associated with that primary data file also are destroyed. Extension files, however, may be deleted without deleting the the associated primary data file.

Extension files are grouped into categories of named types. Examples: *plot files*, *history files*, *slice files*, *gain files*, etc.

When an AIPS task creates a new primary data file from an old one, generally it attaches, to the new file, clones of any extension files associated with the old file that remain relevant to the new one.

false color display — In digital imagery, a *false color display* is one which is generated by using a number $n > 1$ of real-valued functions $f_1(x, y), \dots, f_n(x, y)$ to control the proportions, at each *pixel* coordinate (x, y) , of an additive mixture of three primary hues. In practical terms, the user of a digital display system supplies f_1, \dots, f_n , and twists knobs that control the mapping $\mathbf{R}^n \rightarrow \mathbf{R}^3$ that sends the n pixel values at each (x, y) into the proper image *chromaticity* and *intensity*. Compare *pseudo-color display*.

A so-called *true color display* is obtained with $n = 3$ and with *transfer functions* chosen such that the color assignment corresponds in an approximate way to the actual coloration of a scene (as in a color photograph).

fast Fourier transform algorithm — a fast algorithm for the computation of the *discrete Fourier transform* (DFT) y_0, \dots, y_{n-1} of a sequence of n complex numbers x_0, \dots, x_{n-1} ,

$$y_k = \sum_{j=0}^{n-1} x_j e^{2\pi i j k / n},$$

typically requiring only $O(n \log n)$ arithmetic operations—or a multi-dimensional generalization thereof. By contrast, straightforward, or naïve evaluation of the DFT requires $O(n^2)$ operations. The fast Fourier transform algorithms (**FFT**s) which currently are the most popular are the Cooley-Tukey (1965) algorithms, for the case of n highly composite. For n a power of two, the (radix-2) Cooley-Tukey **FFT** requires about $2n \log_2 n$ real multiplications and $3n \log_2 n$ real additions. More generally, the Cooley-Tukey algorithms require a few times $n\sigma(n)$ complex arithmetic operations, where $\sigma(n)$ is the sum of the prime factors of n , counting their multiplicities. S. Winograd has produced **FFT** algorithms which are more efficient than those of Cooley and Tukey, typically requiring about the same number of additions, but only about 20% the number of multiplications. (Computation of the required complex exponentials—or sines and cosines—is not counted, since these generally are either pre-computed and stored in compact tables, or generated recursively.)

A further advantage of the **FFT** algorithms is their avoidance of round-off error, which can build up severely when the DFT is evaluated by brute-force. There are related, fast algorithms for the convolution of sequences of real numbers, for the discrete cosine transform, etc. Algorithmic details may be found in [H. J. Nussbaumer, *Fast Fourier Transform and Convolution Algorithms*, Springer-Verlag, Berlin, 1982]. The computational complexity of the DFT is discussed by L. Auslander and R. Tolimieri [Is computing with the finite Fourier transform pure or applied mathematics, *Bull. (New Series) Amer. Math. Soc.*, **1** (1979) 847–897].

AIPS programs which use the **FFT** make use of the Cooley-Tukey algorithm. When an *array processor* is used to compute the large two-dimensional DFT’s of data which reside on disk, as typically is required in synthesis mapping, the input/output time greatly exceeds the actual computation time.

FFT — See *fast Fourier transform algorithm*.

FITS format — (Flexible Image Transport System) a magnetic tape data format well-tailored for the transport of image

G. Glossary

data among observatories. The FITS format is recommended for bringing data into and out of AIPS. See [D. C. Wells, E. W. Greisen, and R. H. Harten, FITS: A flexible image transport system, *Astron. Astrophys. Suppl. Ser.*, **44** (1981) 363–370]. Also see *u-v FITS format* and *FITS tape*.

FITS tape — a magnetic tape containing data recorded in the *FITS format*. FITS format data blocks are 2880 bytes in length. The resultant *tape blocking efficiency* is 83%, 75%, and 61% at recording densities of 800, 1600, and 6250 bpi, respectively.

flagging — in AIPS, the act of discarding one or more visibility data points by setting a *u-v data flag* (*q.v.*). Compare *clipping*.

fringe rotator — in a correlating-type radio interferometer, a mechanism to introduce a time-varying phase shift into the local oscillator signal of a receiver, in order to reduce the frequency of the oscillations of the correlator output. Fringe rotation allows the correlator output (whose amplitude is proportional to visibility amplitude) to be sampled at a lower rate. The natural fringe frequency can be as high as 200 Hz on the [VLA](#). The fringe rotation is chosen so that the fringe frequency for a point source located at the so-called *fringe stopping center* would be reduced to zero, or at least close to zero. Usually the fringe stopping center and the *delay tracking center* coincide; both then are called the *visibility phase tracking center*. For further details, see A. R. Thompson's Lecture No. 2 and L. R. D'Addario's Lecture No. 4 in the *Third NRAO Synthesis Imaging Summer School*, and see R. M. Hjellming and J. Basart's Ch. 2 of the *Green Book*.

full-synthesis map — in earth-rotation aperture synthesis, with stationary interferometer elements, a *map* derived from an observation which is of such lengthy duration that the fullest possible *u-v coverage* is obtained (i.e., from an observation extending from “horizon to horizon”). Compare *snapshot*.

gain file — in AIPS, an *extension file*, associated with a *u-v data file*, in which a table of approximate *antenna/i.f. gains* (typically obtained by *self-calibration*) is stored.

Gaussian-tapered sinc function — A useful *gridding convolution function* (*q.v.*), of *support width* equal to the width $m\Delta u$ of m *u-v* grid cells, is given by the separable product of two Gaussian-tapered sinc functions, each of the form

$$C(u) = \begin{cases} \left(\frac{\pi u}{b\Delta u}\right)^{-1} e^{-\left(\frac{\pi u}{b\Delta u}\right)^2} \sin \frac{\pi u}{b\Delta u}, & |u| < \frac{m\Delta u}{2}, \\ 0, & \text{otherwise.} \end{cases}$$

The choice $m = 6$, $a \approx 2.52$, and $b \approx 1.55$, yields what is, in a certain natural sense, an optimal gridding convolution function of this particular parametric form (see [F. R. Schwab, Optimal gridding, [VLA](#) Scientific Memo. No. 132]). Also see *spheroidal function*.

Gerchberg-Saxton algorithm — a simple iterative algorithm which, in the field of signal processing, is used for the extrapolation of band-limited signals—and, in image processing, for deconvolution. Assume that the Fourier transform \hat{f} of an image f has been measured over a region B , and that f is known to be confined to a region A . Let χ_A denote the *characteristic function* of A and χ_B that of B . Denote the measured data by \hat{g}_{approx} —i.e., $\hat{g}_{\text{approx}} = \chi_B \hat{f} + \text{error}$. From the initial approximant f_0 ($f_0 \equiv 0$ may be used) a sequence f_n of successive approximants to f is obtained, via the formula

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

Here, \sim denotes inverse Fourier transform, and μ is a fixed scalar, analogous to the *loop gain* parameter of the Högbom Clean algorithm.

To apply the algorithm in radio interferometry, one may identify χ_B with the *u-v sampling distribution* and think of A to be analogous to a *Clean window*. Denoting the *dirty map* by g and the *dirty beam* by b , the iteration can be written as

$$f_{n+1} = f_n + \mu \chi_A \cdot (\hat{g} - \hat{b} \hat{f}_n)^\sim \quad (= f_n + \mu \chi_A \cdot (g - b * f_n)).$$

The Gerchberg-Saxton algorithm has been implemented by Tim Cornwell in an AIPS program named [APGS](#). [APGS](#) includes an *ad hoc* nonnegativity constraint—at each iteration, any pixel value which would be driven negative is modified to become nonnegative. Convergence usually is sluggish.

Some algorithms which are very similar to the Gerchberg-Saxton algorithm are the Lent-Tuy algorithm, which is used in medical imaging, the Papoulis, or Papoulis-Youla algorithm, used in signal processing, and the so-called method of alternating orthogonal projections, used in image reconstruction. See [J. L. C. Sanz and T. S. Huang, Unified Hilbert space approach to iterative least-squares linear signal restoration, *J. Opt. Soc. Am.*, **73** (1983) 1455–1465] and references cited therein.

Gibbs' phenomenon — in the neighborhood of a discontinuity of a periodic function f , the overshoot and oscillation (or ringing) of the partial sums S_n of the Fourier series for f . In the vicinity of a simple jump discontinuity, S_n always overshoots the mark by about 9%, regardless how large n . See [H. S. Carslaw, *Introduction to the Theory of Fourier's Series and Integrals*, Dover, New York, 1930, ch. IX].

In harmonic analysis, often the Fourier coefficients are multiplied by a weight function tending smoothly to zero at the boundaries of its *support*, in order to smooth out the discontinuities and thereby reduce the ringing in the synthesized spectrum. (This degrades the spectral resolution, however.) See *Hanning smoothing*. For a discussion of Gibbs' phenomenon in the context of [VLA](#) cross correlation analysis, see Larry D'Addario's Lecture No. 4 in the *Third NRAO Synthesis Imaging Summer School*.

GIPSY — (Groningen Image Processing System) a data reduction system, similar in scope to AIPS, used in the Netherlands for analysis of Westerbork Synthesis Radio Telescope (WSRT) data.

global fringe fitting algorithm — an antenna-based algorithm (in the spirit of the *self-calibration algorithm*) for VLBI fringe search. For an n element array, the classical VLBI fringe fitting technique, a correlator-based method, requires the estimation of $n^2 - n$ parameters. The global fringe fitting method reduces this number to $3n - 3$. Expressing the *antenna/i.f. gain* for antenna k of the array as $g_k(t, \nu) = a_k e^{i\psi_k(t, \nu)}$ (here we include a frequency dependence) one has that the observed visibility on the $i-j$ baseline, to first-order, is given by

$$V_{ij}(t, \nu) = a_i a_j V_{ij}(t_0, \nu_0) \times e^{\sqrt{-1}((\psi_i - \psi_j)(t_0, \nu_0) + (r_i - r_j)(t - t_0) + (\tau_i - \tau_j)(\nu - \nu_0))},$$

where V_{ij} is the true visibility, and where the r_k are the *antenna residual fringe rates* and the τ_k the *antenna residual delays*.

Given a source model, one may solve for the $\psi_k(t_0, \nu_0)$, the r_k , and the τ_k , using either a least-squares method or a Fourier transform method. Because of the overdeterminacy provided by a simultaneous solution for the parameters, this method allows proper delay and fringe rate compensation of data on baselines of too low signal-to-noise for the correlator-based method to work effectively. A full description of the method is given in [F. R. Schwab and W. D. Cotton, Global fringe search techniques for VLBI, *Astron. J.*, **88** (1983) 688–694]. This algorithm is implemented in the AIPS program [CALIB](#).

graphics overlay plane — same as *graphics plane*.

graphics plane — a storage area within a TV display device, such as the I^2S , in which a full screen load of one-bit graphics

information (labeling, plotting, axis lines, etc.) is stored. A typical I²S unit is equipped with four graphics planes, each 512 pixels × 512 pixels in area. Compare *image plane*.

gray-scale display — a black-and-white display of a digitized *image*—typically either a photographic or a video display.

gray-scale memory plane — same as *image plane*.

Green Book — *An Introduction to the NRAO Very Large Array*, edited by R. M. Hjellming, NRAO, Socorro, NM—a useful reference on many of the technical aspects of the [VLA](#).

green screen — same as *TEK screen*.

gridding convolution function — in radio interferometer mapmaking, a function C —usually supported on a square the width of, say, six u - v grid cells—with which the u - v measurement distribution is convolved. The purpose is twofold: 1) to interpolate and smooth the data, so that samples may be taken over the lattice points of a rectangular grid (in order that the fast Fourier transform algorithm may be applied) and 2) to reduce aliasing (the convolution in the u - v plane induces a taper in the map plane). See *aliased response*, *gridding correction function*, *cell-averaging*, *dirty map*, and *uniform weighting*.

With judicious choice of C , a high degree of aliasing suppression is possible. A high degree of suppression is desirable, even when there are no “confusing” radio sources very near the field of interest, because the effect is not only to reduce the spurious responses due to sources lying outside of the field of view, but also to reduce the response to sidelobes of the source of interest, which too are aliased into the map from outside the field of view. See *spheroidal function*.

gridding correction function — in radio interferometry, the reciprocal $1/\hat{C}$ of the Fourier transform (FT) of the *gridding convolution function* C . Since the map plane taper induced by the gridding convolution usually is very severe, the dirty map normally is corrected by pointwise division by the FT of the convolution function. Obviously C should be chosen such that \hat{C} has no zeros within the region that is mapped. See *dirty map*.

gripe — in AIPS, an entry in the *gripe file* (*q.v.*).

gripe file — in AIPS, a disk file repository for formal reports of program *bugs*, and for formal complaints and suggestions of a more general nature. A mechanism by which the user may enter gripes into the gripe file is activated by the issuance of the AIPS verb [GRYPE](#). The AIPS group provides prompt, written responses to all *gripes*.

Hanning smoothing function — in the analysis of power spectra, a weight function w by which the measured correlation function is multiplied, in order to reduce that oscillation (*Gibbs' phenomenon*) in the computed spectrum which is due to having sampled at only a finite number of lags. w , as a function of lag, is given by

$$w(\tau) = \begin{cases} \frac{1}{2} \left(1 + \cos \frac{\pi\tau}{\tau_{\max}}\right), & |\tau| < \tau_{\max}, \\ 0, & \text{otherwise.} \end{cases}$$

This is equivalent to convolving the discrete spectrum with the sequence $\{\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\}$.

Hanning smoothing sometimes is applied to the cross correlation measurements obtained in [VLA](#) spectral line observing, in order to reduce the effect of sharp bandpass filter cutoffs. It also is used frequently in radio astronomical autocorrelation spectroscopy. See *Gibbs' phenomenon*, and for more on smoothing see [R. B. Blackman and J. W. Tukey, *The Measurement of Power Spectra*, Dover, New York, 1958].

hard copy — computer output printed on paper (rather than, say, written on magnetic tape); e.g., a printed contour plot or gray scale display, or a listing of a catalog file.

hardware mount — the combined acts of installing a computer external storage module, such as a disk pack or a reel of magnetic tape, in some electro-mechanical unit (e.g., a disk drive or a tape drive) that provides computer access to this data storage medium, and placing that unit in readiness to be operated under computer control (e.g., positioning a magnetic tape at the *BOT marker*). Compare *software mount*.

header record — a distinguished record within a *data file*—generally the first record—which serves to define the contents of the other records in the file by supplying relevant parameters, units of measurement, etc.; also termed simply *header*.

In AIPS, however, the header record of each *primary data file* is stored apart from that file, in a file which is termed a “CB” file. And a directory, termed a *catalog file* (*q.v.*), or “CA” file, of all of each user’s primary data files on a given disk is stored on that disk. AIPS *extension file* headers are stored within the extension files themselves.

help file — in AIPS, a *text file*, whose contents may be displayed on the terminal screen of the interactive user, giving a brief explanation of a particular AIPS verb, adverb, pseudoverb, task, or miscellaneous general feature. Compare *explain file*.

Hermitian function — a complex-valued function, of one or more real variables, whose real part is an even function and whose imaginary part is odd. The Fourier transform (FT) of a real-valued function is Hermitian, and the inverse FT of a Hermitian function is real.

Since each of the radio brightness distributions $I(x, y)$, $Q(x, y)$, $U(x, y)$, and $V(x, y)$ representing *Stokes’ parameters* is real-valued, *Stokes’ visibility functions* have the property of *conjugate symmetry*: $V_I(-u, -v) = \bar{V}_I(u, v)$, $V_Q(-u, -v) = \bar{V}_Q(u, v)$, $V_U(-u, -v) = \bar{V}_U(u, v)$, and $V_V(-u, -v) = \bar{V}_V(u, v)$. (Here, $V_I = \hat{I}$, $V_Q = \hat{Q}$, etc., where $\hat{\cdot}$ denotes FT.)

history file — in AIPS, an *extension file* containing a summary of all, or most of the processing, by AIPS tasks, of the data recorded in all associated files.

Hogbom Clean algorithm — a deconvolution algorithm devised by Jan Högbom for use in radio interferometry [J. A. Högbom, Aperture synthesis with a non-regular distribution of interferometer baselines, *Astron. Astrophys. Suppl. Ser.*, **15** (1974) 417–426]. Denote (the discrete representations of) the dirty map by g and the dirty beam by b . The algorithm iteratively constructs discrete approximants f_n to a solution f of the equation $b * f = g$, starting with an initial approximant $f_0 \equiv 0$. At the n th iteration, one searches for the peak in the residual map $g - b * f_{n-1}$. A δ -function component, centered at the location of the largest residual, and of amplitude μ (*loop gain*) times the largest residual, is added to f_{n-1} to yield f_n . The search over the residual map is restricted to a region A termed the *Clean window*. The iteration terminates with an approximate solution f_N either when N equals some iteration limit N_{\max} , or when the peak residual (in absolute value) or the r.m.s. residual decreases to some given level.

To diminish any spurious high spatial frequency features in the solution, f_N is convolved with a narrow elliptical Gaussian function h , termed the *Clean beam*. Generally h is chosen by fitting to the central lobe of the dirty beam. Also, one generally adds the final residual map $g - b * f_N$ to the approximate solution $f_N * h$, in order to produce a final result, termed the *Clean map*, with a realistic-looking level of noise. See *super-resolution*.

host computer — In the parasitic relationship of a computer program or program package, such as AIPS, to the computer on which it runs, the latter is termed the *host computer*. Also, in the

G. Glossary

master–slave relationship of a computer to one of its peripheral devices, such as an array processor, the master may be termed the *host*.

hue — one of the three basic parameters (*hue*, *intensity*, and *saturation*) which may be used to describe the physical perception of the light that reaches one's eye. Hue, which is also termed *tint*, or simply *color*, refers to the dominant wavelength of the coloration, at a given location in an image or scene. The term also may be used to describe a multimodal color spectrum—e.g., one speaks of a purple hue. Different spectral distributions of light, of identical intensity and saturation, are capable of producing identical retinal responses; these unique responses comprise the set of perceptible hues.

Color matching tests have established that there are three basic types of human retinal receptors, whose peak responses are to red, green, and blue light. These are the three *primary hues* used in additive color mixing—e.g., in digital image display. They may be used to produce all, or virtually all, of the perceptible hues.

See C.I.E. *chromaticity diagram*.

hybrid mapping algorithm — an algorithm for calibration of radio interferometer data which is essentially equivalent to the *self-calibration algorithm* (q.v.) (used in VLA data reduction), except in that it makes explicit use of the *closure phase* and *closure amplitude* relations, rather than explicit use of the relation $\tilde{V}_{ij} = g_i \bar{g}_j V_{ij}$ relating observed visibility to the product of the true visibility and a pair of *antenna/i.f. gains*. Hybrid mapping, which is used extensively in VLBI data reduction, is described in [A. C. S. Readhead *et al.*, Mapping radio sources with uncalibrated visibility data, *Nature*, **285** (1980) 137–140].

Either algorithm (assuming that one cares to make some distinction) can be applied to data obtained with connected (e.g., the VLA) and non-connected-element interferometers (e.g., VLBI arrays). Any differences in the results produced by the two algorithms would be attributable primarily to differences in the effective weighting of the data (in particular, early implementations of both algorithms discarded data which could have been used to obtain overdetermined solutions for the calibration parameters).

IIS — See I^2S .

image — in the context of AIPS, any finite-volume, linear, rectangular, or hyper-rectangular array of pixels; e.g., a digitized photograph, or a radio map. The term also is used (less technically) to refer to the *display* of data—e.g., a television picture of a radio map.

image catalog — in AIPS, a disk file containing data records describing the data stored on the TV display device *image planes*. These records are essentially identical in structure to the *header records* stored in the *catalog file*. The data in the image catalog furnish the information that is required for proper axis labeling, pixel value retrieval, etc.

image file — in AIPS, a *primary data file* whose content is an *image*.

image plane — a storage area within a TV display device, such as the I^2S , in which a full screen load of single word pixels is stored. A typical I^2S unit is equipped with four image planes, each 512 pixels \times 512 pixels in area (each pixel is represented by eight bits). Often several image planes are used at one time—either for black-and-white or *pseudo-color* display of a large image, sections of which may occupy different image planes—or for *false color* or *true color* display of a smaller image, now using, say, three image planes—one to control each of the three electron guns (for red, green, or blue phosphor) in the TV display. Compare *graphics plane*.

image reconstruction — the attempted recovery of an *image* after it has undergone the distorting effects, the blurring, etc., produced by some physical measurement and recording device, such as a camera, a radio interferometer, or a tomography machine. The operation of many measurement devices can be adequately modeled by a linear Fredholm integral equation of the first kind. In the two-dimensional case, e.g., one assumes that the measurement $g(x, y)$ is related to the undistorted image $f(x, y)$ by the equation

$$g(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} K(x, y, x', y') f(x', y') dx' dy' + \epsilon(x, y).$$

(Often it is convenient to use the more compact, operator notation, $g = \mathbf{K}f + \epsilon$.) The kernel K of the equation is called the *point spread function*, (q.v.). Measurement error and the error arising from any simplifying assumptions are lumped together into the $\epsilon(x, y)$ term. Some particularly well-behaved measurement systems can be adequately modeled by a simple convolution equation, in which case K is given by $K(x, y, x', y') = h(x - x', y - y')$. This is the case, e.g., when the VLA is used to observe a small ‘unconfused’ radio source; then g may be identified with the *dirty map* and h with the *dirty beam*. Or when K , considered as a function of (x, y) , is given at each (x', y') by the *delay beam* for that position, the equation models the *bandwidth smearing effect* (q.v.); as the bandwidth $\rightarrow 0$, the convolution model again becomes valid.

Except in trivial cases, solution of the Fredholm equation always is an ill-posed problem. Mild conditions on K and f (the classical ‘Picard conditions’—see F. Smithies [*Integral Equations*, Cambridge Univ. Pr., London, 1958]) ensure the existence of (non-unique) solutions when $\epsilon \equiv 0$. But, because of the effect of measurement noise, one usually does not seek an exact solution, but rather an approximate solution—one which fits the data to within the measurement errors. Uniqueness and regularity of the computed approximate solution are obtained by imposing such constraints as known *support*, nonnegativity, and smoothness conditions. See *regularization method*. Also see H. C. Andrews and B. R. Hunt [*Digital Image Restoration*, Prentice-Hall, Englewood Cliffs, NJ, 1977] and *phaseless reconstruction*.

inputs file — in AIPS, a *text file*, whose contents may be displayed on the terminal screen of the interactive user, giving a summary of the *adverbs* relevant to a given *verb* or a given AIPS *task*.

instrumental polarization — any contamination of a polarization measurement by an instrument’s response to an undesired polarization state. In radio interferometry, the instrumental polarization arises mainly from feed imperfections and from plumbing leaks between the feeds and the receiver front-ends. One tries to remove the instrumental polarization by applying corrections derived from observations of calibration sources whose polarization properties are known. Within AIPS, there is, at present, no facility for polarization calibration. The polarization calibration of VLA data normally takes place on the DEC-10 computer at the VLA. For more details, see Carl Bignell’s Lecture No. 4 in the 1985 *Summer School Proceedings*. See *beam squint*.

intensity — one of the three basic parameters (*hue*, *intensity*, and *saturation*) which may be used to describe the physical perception of color. Intensity is a measure of the energy of the spectral distribution, at a given point in an image or scene, weighted by the spectral response of the visual system. *Luminance* is the energy of the physical spectrum, but not weighted by the visual response. *Brightness* sometimes is used synonymously with either term.

See C.I.E. *chromaticity diagram*.

invisible distribution — in the context of radio interferometry, a function f (or a generalized function—or distribution) whose Fourier transform \hat{f} vanishes everywhere that the interferometer pairs have sampled. This term was introduced by R. N. Bracewell and J. A. Roberts [Aerial smoothing in radio astronomy, *Austr. J. Phys.*, **7** (1954) 615–640]. Also see *principal solution*.

For an actual interferometer, there exist fewer physically plausible invisible distributions than for an idealized interferometer. This is because each visibility sample is not a point sample of \hat{f} , but rather some kind of local average. By the *Paley-Wiener theorem*, if \hat{f} is nontrivial and vanishes in some open neighborhood, then f cannot be of *compact support*, and hence it may be considered implausible.

IPL — (Initial Program Load) same as *boot*.

isoplanaticity assumption — in the context of radio interferometry (the term is used too in optics), the assumption that over each element of an array all wavefronts arriving from different parts of the sky to which the interferometer pairs are sensitive are subject to identical atmospheric phase perturbations. A patch of sky over which the assumption is valid is referred to as an *isoplanatic patch*.

Approximate validity of the isoplanaticity assumption is a necessary condition for the success of calibration (*self-calibration*, in particular) of radio interferometer data (from an earth-based array) if one is to rely on a model incorporating time-varying *antenna/i.f. gains*, one per antenna, whose arguments (or phases) are to include the atmospheric phase corruption. However, see F. R. Schwab [Relaxing the isoplanatism assumption in self-calibration; applications to low-frequency radio interferometry, *Astron. J.*, **89** (1984) 1076–1081].

I²S — (International Imaging Systems Models 70 and 75) a TV display device, capable of both black-and-white and color display, manufactured by the Stanford Technology Corporation. At an AIPS site typically it is equipped with four 512 pixel \times 512 pixel eight-bit *image planes*, four one-bit *graphics planes*, a *trackball*, and sometimes an *ALU*. The eight-bit pixel representation (in the image planes) allows the intensity of each of the three electron gun beams to be set at any of 256 discrete levels. (Actually, 1024 levels can be used, because of an extra two bits of capability provided in the *transfer function* tables and the internal arithmetic unit.) An I²S is attached to three of the NRAO's computers on which the AIPS system runs (the **VLA** and Charlottesville Vaxes).

line editor — a *text editor* (q.v.) which allows the modification of single lines or records within a text file, but one which does not allow the simultaneous modification of more than one line. *SOS* and *SEDT* are both line editors. *Screen editors* (q.v.) are more versatile than line editors.

lobe rotator — same as *fringe rotator*, (q.v.).

loop gain — in the Högbom Clean algorithm, the fraction μ of the largest residual which is used in determining the amplitude, or flux, of a *Clean component*. Convergence can be achieved for μ in the range (0, 2), but generally a small value, say $\mu = \frac{1}{10}$, is recommended, especially in dealing with extended sources. See *Högbom Clean algorithm*.

luminance — See *intensity*.

l_1 solution algorithm — See *self-calibration gain solution algorithm*.

l_2 solution algorithm — See *self-calibration gain solution algorithm*.

major cycle — In the *Clark Clean algorithm* (q.v.), a number of minor cycles, or inner iterations, followed by the computation by the **FFT** algorithm of the full residual map, comprise a major cycle.

map — an *image*, one or more of whose coordinate axes represents some spatial coordinate.

maximum entropy method — a regularization method (q.v.) for the numerical solution of ill-posed problems, given noisy data, in which the regularizing (or smoothing) term—which measures the roughness of the computed approximate solution \hat{f} —is given by the negative of the Shannon entropy of \hat{f} , $-H(\hat{f})$: in the continuous case, letting A denote the domain of definition of \hat{f} , $H(\hat{f}) \equiv -\int_A \hat{f}(x) \log \hat{f}(x) dx$, where \hat{f} has been normalized so that $\int_A \hat{f}(x) dx = 1$ (and $0 \log 0 \equiv 0$); and in the discrete case, $H(\hat{f}) \equiv -\sum \hat{f}(x_i) \log \hat{f}(x_i)$, where \hat{f} has been normalized so that $\sum \hat{f}(x_i) = 1$. The underlying philosophy of the method, espoused early on by Jaynes ("Jaynes' method of prior estimation", [E. T. Jaynes, Prior probabilities, *IEEE Trans. Syst. Sci. Cyb.*, **SSC-4** (1968) 227–241]) and by J. P. Burg at a 1967 meeting of the Society of Exploration Geophysicists, is that one is being "maximally noncommittal" in regard to the insufficiency of the data if one maximizes the entropy, and thus minimizes the "information content", of \hat{f} , subject to the constraint that \hat{f} should agree with the given data.

For one-dimensional discrete convolution equations, with noiseless, regularly-spaced data, there exists a closed-form solution—for other cases, iterative methods are used, as with other forms of the regularization method.

Use of the method in radio astronomy was encouraged by J. G. Ables in 1972 in public lectures, and it now is in common use in radio interferometry (cf. [S. F. Gull and G. J. Daniell, Image reconstruction from incomplete and noisy data, *Nature*, **272** (1978) 686–690]). Nonnegativity of the computed solution is a natural by-product of the method. For reconstruction of polarized brightness distributions in interferometry (Stokes' Q , U , and V), which, unlike the total intensity, may assume negative values, Ponsonby has derived an appropriate generalization of the method [J. E. B. Ponsonby, An entropy measure for partially polarized radiation..., *Mon. Not. R. Astr. Soc.*, **163** (1973) 369–380]. See *Variational Method*.

memory page — See *virtual memory page*.

memory paging — same as *virtual memory page swapping*.

memory thrashing — an excessive amount of *virtual memory page swapping* (q.v.) on a computer (such as the Vax) with a virtual memory operating system. A condition of memory thrashing is likely to occur whenever too many programs with large memory requirements are active (a single program with excessive memory requirements also can cause memory thrashing).

message file — in AIPS, a *text file* containing progress report messages generated during the execution of AIPS *tasks* and also containing a chronicle of the user's interaction (via *verb* commands) with AIPS. Each AIPS user is assigned a message file, the contents of which may be printed out, typed upon a terminal display screen, or emptied—at will—by invoking the appropriate verb command. See *AIPS monitor*.

message terminal — same as *AIPS monitor*.

minor cycle — in the *Clark Clean algorithm* (q.v.), an inner iteration, in which the peak residual over a subregion (the *Clean window*) of the full residual map is found and is used to obtain the next successive iterate. Compare *major cycle*.

microcode — See *array processor microcode*.

monitor — See *AIPS monitor* or *Conrac monitor*.

$m \times n$ map — The convention adopted for AIPS is opposite the standard matrix algebra terminology: whereas an $m \times n$

G. Glossary

matrix is comprised of m rows and n columns, an $m \times n$ map or *image* in AIPS has, in the usual display format, m pixels along the horizontal axis (usually termed the *x*-axis) and n pixels along the vertical axis (usually termed the *y*-axis). Moreover, pixels of a two-dimensional map in the usual display format are numbered from the bottom left-hand corner: the pixel location specified by the ordered pair (i, j) is in column number i and row number j , counting from the bottom left. In other than two-dimensional "images", the $(1, \dots, 1)$ pixel is also said to be at the "bottom left corner" (BLC), just as in the two-dimensional case. See *data cube*, *pixel coordinates*, and *coordinate reference pixel*.

MX — See "battery-powered" *Clean algorithm*.

natural weighting — See *uniform weighting*.

negative bowl artifact — See *zero-spacing flux*.

non-closing offset — See *correlator offset*.

Nyquist sampling rate — the slowest rate of sampling which, according to the *Shannon sampling theorem* (q.v.), would allow a band-limited function $f(t)$ to be recovered via the *Shannon series*. If the smallest symmetric interval which contains the *support* of the Fourier transform of f is the interval $[-a, a]$, then the Nyquist sampling rate for f is $2a$; i.e., the interval between samples (the *sampling period*) must be less than the *reciprocal bandwidth* $1/2a$. The terms *oversampling* and *undersampling* refer to sampling at rates faster or slower than the Nyquist rate. The difference between f and the Shannon series formed from too coarsely spaced samples is called *aliasing*.

operating system —

page — See *virtual memory page* and *terminal page*.

page swapping — See *virtual memory page swapping*.

Paley–Wiener theorem — The classical Paley–Wiener theorem says that a square-integrable complex-valued function f , defined over the real line, can be extended off the real line as an entire function of exponential type $\leq 2\pi a$ if and only if $f(x) \equiv 0$ for $|x| > a$ —i.e., iff f is band-limited to $[-a, a]$ (here \hat{f} denotes Fourier transform). (An everywhere-analytic function $g(z)$ is said to be of exponential type $\leq A$ if $\exists c$ such that, for all z , $|g(z)| \leq ce^{A|z|}$.) For a derivation, see H. Dym and H. P. McKean [Fourier Series and Integrals, Academic Press, 1972]. The *Shannon series* is a means of extending \hat{f} to \mathbb{C} . The extension of the Paley–Wiener theorem to the case of generalized functions (to tempered distributions) is called the Paley–Wiener–Schwartz theorem.

The Fourier transform $\hat{f} : \mathbf{R}^n \rightarrow \mathbf{C}$ of a function f with support in a given n -dimensional convex compact set K can be analytically extended to all of \mathbf{C}^n . Growth properties on \hat{f} which are sufficient in order for the converse to hold are given by K. T. Smith, D. C. Solomon, and S. L. Wagner [Practical and mathematical aspects of the problem of reconstructing objects from radiographs, *Bull. Amer. Math. Soc.*, **83** (1977) 1227–1270] (in addition to the classical version of the multi-dimensional Paley–Wiener theorem, for rectangular K , they give versions with tighter growth bounds, and for arbitrary convex K). Smith *et al.* use the Paley–Wiener theorems to establish indeterminacy theorems for tomographic reconstruction. Their results are also relevant to Fourier synthesis, because of the connection between the two-dimensional Fourier transform and the one-dimensional Radon transform. The Paley–Wiener theorems have also been used in establishing results on the problem of *phaseless reconstruction* (q.v.) and in proving the convergence of constrained Gerchberg–Saxton-type algorithms (see A. Lent and H. Tuy [An iterative method for the extrapolation of band-limited functions, *J. Math. Anal. Appl.*, **83** (1981) 554–565]).

phaseless reconstruction — the reconstruction of an image f (see *image reconstruction*) from knowledge of (only) the magnitude $|\hat{f}|$ of the Fourier transform of f (and usually from only partial knowledge of $|\hat{f}|$). Phaseless reconstruction has been considered for the NRAO's proposed millimeter wave interferometer array [T. J. Cornwell, Imaging of weak sources with compact arrays, NRAO Millimeter Array Memo. No. 12]. Recent results on phaseless reconstruction appear in the JOSA Feature Issue on Signal Recovery [*J. Opt. Soc. Am.*, **73** No. 11 (Nov. 1983)]. Also see the papers by J. R. Fienup and by R. H. T. Bates *et al.* in the 1983 *Sydney Conference Proceedings*.

phase tracking center — same as *visibility phase tracking center*, (q.v.).

physical memory — core or semiconductor memory within a computer (as opposed to slower memory—virtual memory, disk storage, magnetic tape footage, etc.). A typical Vax is equipped with a physical memory 3–4 megabytes in size.

pillbox — See *cell-averaging*.

pixel — (picture element) an element of a digitized image (or of a map). A pixel is characterized by its position in the image and by its numerical value. See *$m \times n$ map*, *coordinate reference pixel*, and *pixel coordinates*.

pixel coordinates — in an AIPS *image file*, the pixels are numbered consecutively, beginning with $(1, \dots, 1)$ at the bottom left corner (BLC) of the image. See *coordinate reference pixel* and *$m \times n$ map*.

plot file — an AIPS extension file containing plotting information, in the form of the commands which are necessary in order for a line drawing peripheral device, such as a Calcomp or other pen plotter, a green screen, or an electrostatic printer/plotter, to generate a plot.

point source response — same as *point spread function*.

points per beam — in a digitized radio map, the characteristic width, somehow defined, of the major lobe of the *beam* pattern, or *point spread function*, divided by the *pixel separation*. Ordinarily the number of points per beam is calculated by measuring the narrowest diameter of the 50% contour level of the major lobe of the beam. To avoid excessively severe discretization error, deconvolution algorithms such as the Högbom *Clean algorithm* and the *maximum entropy method* require, as a rule-of-thumb, at least three (and preferably 4–5) points per beam.

point spread function — (PSF) 1. the response of a system or an instrument to an impulsive, or point source, input. 2. in radio interferometry, the response of the instrument to a point, or unresolved, radio source—a fancy term for *beam*. Ignoring instrumental effects, such as finite bandwidth and finite integration time, the response does not depend upon the displacement of the source away from the *visibility phase tracking center*—hence the term *space-invariant PSF* (SIVPSF), and the contrary term *space-variant PSF* (SVPSF).

A so-called linear space invariant measurement system (i.e., a linear system with an SIVPSF) is equivalently described as a system which can be modeled by a convolution equation; a linear space-variant measurement system is modeled by a more general linear Fredholm integral equation of the first kind. See *image reconstruction*.

POPS — (People-Oriented Parsing System) the parser, or command interpreter, embedded within the AIPS program; that part of the AIPS program which attempts to interpret the user's commands (*POPS symbols*) and then initiate the appropriate reaction. POPS is used in other astronomical data reduction programs at the NRAO: in Condare, TPOWER/SPOWER, and the Tucson 12 m single-dish packages.

POPS procedure — See *POPS symbols*.

POPS symbols — The AIPS user's primary means of communicating his wishes to AIPS is by typing commands, termed *POPS symbols*, at the keyboard of a computer terminal. There are four classes of POPS symbols: *adverb*, *verb*, *pseudoverb*, and *procedure*. An *adverb* is a symbol representing the storage area for a datum or for data that are used to control the action of verbs, tasks, and procedures; that is to say that the adverb symbols are used to set *control parameters*. A *verb* is a symbol which causes POPS (or AIPS) to initiate some action after POPS has finished interpreting, or compiling, the command line typed at the computer terminal. A *pseudoverb* is a symbol which suspends, temporarily, the normal parsing of an input line and which causes some action to take place while the line is being compiled, and, possibly, after compilation. A *procedure* is a symbol representing a pre-compiled sequence of POPS symbols. Also see *task*.

primary beam correction — in radio interferometry, the multiplicative correction of a radio *map* by the reciprocal of an average of the power patterns of the array elements. Measurements of the primary beam parameters of the 25 m [VLA](#) elements are given by Peter Napier and Arnold Rots in the memorandum [[VLA](#) primary beam parameters, [VLA](#) Test Memo. No. 134, Feb. 1982]. There an average power pattern and its reciprocal are approximated by radial functions, polynomials in the distance from the pointing position. The AIPS task [PBCOR](#) is used to apply this correction to [VLA](#) maps. The appropriate correction at large distances from the pointing position is not well-determined, thus [PBCOR](#) "blanks" the map pixel values beyond a certain radius (see *blanked pixel*).

primary data file — in AIPS, either a *u-v datafile*, containing measurements of the visibility function of a radio source, or an *image file*, containing a digitized image or a radio map. Compare *extension file*.

principal solution — in the context of radio interferometry, the inverse Fourier transform of the *u-v measurement distribution*; i.e., the *dirty map* (*q.v.*) in sense 1 of the definition. This term was introduced by R. N. Bracewell and J. A. Roberts [Aerial smoothing in radio astronomy, *Austr. J. Phys.*, 7 (1954) 615–640]. Except in the trivial case, the principal solution to the mapping problem in interferometry is a physically implausible solution, because the principal solution has not the property of *compact support*.

An *invisible distribution* (*q.v.*) added to the principal solution yields another solution—i.e., another brightness distribution which is consistent with the observations.

procedure — See *POPS symbols*.

prolate spheroidal wave function — an eigenfunction of the finite, or truncated, Fourier transform—more precisely, for given c , one of the countably many solutions of the integral equation

$$\nu f(\eta) = \int_{-1}^1 e^{icnt} f(t) dt;$$

equivalently, a solution of the differential equation $(1 - \eta^2)f'' - 2\eta f' + (b - c^2\eta^2)f = 0$; or, equivalently, a solution of the wave equation in a system of prolate spheroidal coordinates. The eigenfunction of the above equation associated with the largest eigenvalue ν is termed the 0-order solution.

If we want a gridding convolution function C , of *support width* equal to the width of m grid cells, that is optimal in the sense that its Fourier transform \hat{C} has the property that the concentration ratio

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

is maximized, then C is the separable product of two 0-order prolate spheroidal wave functions, with $c = \pi m/2$. See *gridding convolution function* and *spheroidal function*.

prompt character — a character (often the dollar sign "\$" or the greater-than sign ">") which the computer program or the operating system prints on the terminal screen of the interactive user in order to prompt, or invite, a typed response from the user. The AIPS program's standard prompt character is the greater-than sign, and on the Vaxes at the NRAO the operating system's prompt character is the dollar sign. On most UNIX systems, the prompt character is the percent sign. Thus, most commands (or *POPS symbols*) peculiar to AIPS must be typed on a line beginning with the >-character, and any command to the operating system, such as the command to mount a tape, must be typed on a line beginning with the \$- or %-character.

When operating in some lesser-used, special modes, AIPS employs other prompt characters: ":" for procedure building, ";" for procedure editing, "!" for entry of gripes, "<" for batch file preparation, and "#" for parameter reading.

Prussian helmet Clean algorithm — a modified version of the Högbom Clean algorithm, devised by Tim Cornwell. The idea is to drive the Clean algorithm toward an approximate solution f of minimal Euclidean norm—i.e., to find an f consistent with the data, confined to the Clean window, comprised of a small number of point components, and such that $\iint_{\text{window}} [f(x, y)]^2 dx dy$ is minimized. This is accomplished by adding a δ -function of amplitude ω , centered at the origin, to the dirty beam, and then just proceeding as normal with the Clean algorithm. Proper choice of ω depends on the distribution of measurement errors. See [T. J. Cornwell, A method of stabilizing the Clean algorithm, *Astron. Astrophys.*, 121 (1983) 281–285]. A provision for this modification is incorporated in the AIPS tasks [APCLN](#) and [MX](#). See *regularization method*.

pseudo-AP — See *pseudo-array processor*.

pseudo-array processor — in AIPS, the term which is applied to a collection of Fortran subroutines which may be used to emulate the operation of an FPS Model AP-120B array processor. At those AIPS sites which do not have an array processor, the AIPS tasks which normally would make use of an array processor use the pseudo-array processor subroutines instead. See *array processor*.

pseudo-color display — In digital imagery, a *pseudo-color* display is one which is derived from a single real-valued function $f(x, y)$ and a mapping $\mathbf{R}^1 \rightarrow \mathbf{R}^3$ that controls the *hue*, *intensity*, and *saturation*—or, equivalently, the proportions in an additive mixture of three primary hues—of the coloration at each *pixel* coordinate (x, y) of the display, according to the value of $f(x, y)$. A pseudo-color display might be used, for example, to represent measurements of the intensity of the radio continuum flux density of a source.

Compare *false color display* and see *color contour display*.

pseudo-continuum u-v data file — in [VLA](#) spectral line data reduction, a *u-v data file* containing the visibility measurements from a small number of spectral line channels, recorded in the same format as continuum visibility data. The purpose is to enable the use, for spectral line data analysis, of programs originally intended only to handle continuum data reduction.

pseudoverb — See *POPS symbols*.

PSF — See *point spread function*.

Q-routine — in AIPS, a primitive level subroutine designed to function on a particular manufacturer's production model of an *array processor*. A goal of the AIPS project is to construct

G. Glossary

libraries of Q-routines—one library appropriate to each model of array processor which might be used in conjunction with AIPS—with identical names, argument lists, and functionality. Existing Q-routines emulate the standard library of Floating Point Systems, Inc.'s, model *AP-120B array processor*.

quick boot — an abbreviated boot procedure. See *boot*.

RANCID — (Real (or Radio) Astronomical Numerical Computation and Imaging Device) the name by which the AIPS data reduction system formerly was known.

re-boot — Having booted once already, one *re-boots*. See *boot*.

regularization method — in the numerical solution of ill-posed problems, given noisy data, a method in which the original problem is converted into a well-posed problem by requiring of the solution to the modified problem (which now is an approximate solution to the original problem) that it satisfy some smoothness constraint. The prototypical ill-posed problem has the form $Kf = g + \epsilon$, where K is a known linear integral operator (e.g., a convolution operator), where $g + \epsilon$, which is given, represents some noisy measurement, and where f is unknown. In the context of radio interferometry, one may take $g + \epsilon$ to be the *dirty map* and K to be the operator which convolves the “true” radio source brightness distribution f with the *dirty beam*. Now, denoting our approximate solution to the ill-posed problem by \tilde{f} , \tilde{f} is found by minimizing the expression

$$(1 - \lambda)\|g - K\tilde{f}\|^2 + \lambda S(\tilde{f}),$$

for some given choice of the *regularization parameter* λ , $0 < \lambda < 1$. $\|g - K\tilde{f}\|^2$ is the mean squared residual (occasionally some other measurement of the error is used), and $S(\tilde{f})$ is a measure of the roughness of the computed solution—say, some power of a norm or seminorm of \tilde{f} , or a similar quantity, such as the negative of the (Shannon) entropy of \tilde{f} .

Proper choice of λ must be based on statistical considerations which depend on the distribution of measurement errors; often, one chooses λ in order achieve an *a priori* reasonable value of the mean squared residual. The *maximum entropy method*, *Tikhonov regularization*, and the *Prussian helmet Clean algorithm* are special cases of the regularization method. Appropriate choice of S is discussed by J. Cullum [The effective choice of the smoothing norm in regularization, *Math. Comp.*, **33** (1979) 149–170], and the choice of S and λ , by a statistical method known as “cross validation”, is described by G. Wahba [Practical approximate solutions to linear operator equations when the data are noisy, *SIAM J. Numer. Anal.*, **14** (1977) 651–677]. Often, some Sobolev norm is chosen for S .

Usually, in addition to the smoothness constraint, f is assumed to be of known, *compact support*. Other constraints, such as nonnegativity, may be included as well. In the case in which the data are exact—i.e., when $\epsilon = 0$, so that $g = Kf$ —one may obtain the regularized solution corresponding to $\lambda = 0$ as the limit of regularized solutions \tilde{f}_λ as $\lambda \rightarrow 0$. See *Variational Method*. Also see D. M. Titterington [General structure of regularization procedures in image reconstruction, *Astron. Astrophys.*, **144** (1985) 381–387].

regularization parameter — in the *regularization method* (*q.v.*) for the solution of ill-posed problems, a smoothing parameter λ , $0 < \lambda < 1$, which controls the trade-off between an error term, measuring agreement of the computed solution \tilde{f} with the given data, and a term $S(\tilde{f})$, which measures the roughness of \tilde{f} . I.e., λ controls the amount of “regularization”. See *super-resolution*.

re-IPL — same as *re-boot*.

residual delay — Expressing the *antenna/i.f. phase*, ψ_k , for antenna k of a VLBI array as a function of frequency as well as of time, the residual delay on the $i-j$ baseline at (t_0, ν_0) is given by $\tau_{ij} \equiv \left. \frac{\partial(\psi_i - \psi_j + \phi_{ij})}{\partial \nu} \right|_{(t_0, \nu_0)}$, where ϕ_{ij} denotes the visibility phase on the $i-j$ baseline. (The partial w.r.t. t is called the *residual fringe rate*.) Usually the major contributor to residual delay is the difference in the station clock errors. The residual delay is a group delay, rather than a phase delay. It is termed residual because it is assumed that geometric effects have already been compensated for.

The “antenna components” of τ_{ij} , namely $\tau_k \equiv \left. \frac{\partial \psi_k}{\partial \nu} \right|_{(t_0, \nu_0)}$, are called the *antenna residual delays*. They are among the solution parameters of the global fringe fitting algorithm for VLBI. See *residual fringe rate* and *global fringe fitting algorithm*.

residual fringe rate — Expressing the *antenna/i.f. phase*, ψ_k , for antenna k of a VLBI array as a function of frequency as well as of time, the residual fringe rate on the $i-j$ baseline at (t_0, ν_0) is given by $r_{ij} \equiv \left. \frac{\partial(\psi_i - \psi_j + \phi_{ij})}{\partial t} \right|_{(t_0, \nu_0)}$, where ϕ_{ij} denotes the visibility phase on the $i-j$ baseline. (The partial w.r.t. ν is called the *residual delay*.) Usually the major contributor to residual fringe rate is the drift of the station clocks.

The “antenna components” of r_{ij} , namely $r_k \equiv \left. \frac{\partial \psi_k}{\partial t} \right|_{(t_0, \nu_0)}$, are called the *antenna residual fringe rates*. They are among the solution parameters of the global fringe fitting algorithm for VLBI. See *residual delay* and *global fringe fitting algorithm*.

resolution — See *spatial resolution*.

restoring beam — same as *Clean beam*.

roam — See *TV roam*.

run file — in AIPS, a *text file* written by an AIPS user and containing a sequence of AIPS commands (*POPS symbols*). Run files are useful for the storage of strings of commands which one might wish to execute repeatedly (in particular, for the storage of lengthy *procedures*). The run files for all users at a particular AIPS installation are stored in a common area. These files ordinarily are created through use of one of the standard *text editors* of AIPS' host computer.

sampling theorem — See *Shannon sampling theorem*.

saturation — one of the three basic parameters (*hue*, *intensity*, and *saturation*) which may be used to describe the physical perception of color. Saturation is a measure of the (perceived) narrowness of the color spectrum, or the difference of the hue from a gray of the same intensity. Neutral gray—or a “white” spectrum—is termed 0% saturated, and a monochromatic spectrum is termed 100% saturated.

See *C.I.E. chromaticity diagram*.

scratch — 1. The act of deleting a data file—i.e., surrendering the storage medium space which that file occupies—is termed *scratching* the data file. Use of the term *delete* may be preferable, but *scratch* is more common among AIPS users. One who is about to delete a data file may wish first to create a back-up copy. See *back-up*. 2. an adjective meaning *temporary*, as in *scratch file*.

In AIPS a primary data file and all of its associated extension files can be deleted by means of the verb *ZAP*.

scratch file — a data file intended for temporary storage (esp., of data which represent intermediate results—i.e., *scratchwork*). Many of the AIPS tasks use scratch files; the necessary scratch files are created and destroyed automatically by the tasks. However, when an AIPS task *crashes*, sometimes a scratch file remains.

screen editor — a *text editor* (*q.v.*) which, unlike a *line editor*, allows the simultaneous modification of more than one line or record within a text file. For example, a mechanism to facilitate alignment of margins often is incorporated by a screen editor. *EDT*, *EVE*, *vi* and *EMACS* are screen editors.

scroll — See *terminal scroll* and *TV scroll*.

self-calibration algorithm — Many of the systematic errors affecting interferometer visibility measurements may be assumed to be multiplicative and ascribable to individual array elements. That is, in an n element array, the observations on the $n(n-1)/2$ baselines are afflicted by n sources of systematic error, the so-called *antenna/i.f. gains* $g_k(t)$. Given a rough estimate of the true source visibility, a model obtained, say, by mapping and Cleaning roughly calibrated data, one may solve for the unknown gains—and it is not unreasonable to do so, because there are $(n-1)/2$ times more observations than antenna gains. The number of degrees of freedom can be held further in check by assuming that the $g_k(t)$ are slowly-varying or that they are of unit modulus (*i.e.*, that no amplitude errors are present), or by designing an array with redundant spacings.

Having once solved for the unknown g_k , one may correct the data, make another map, and repeat the process. This iterative scheme, which yields successive approximations to the true radio source brightness distribution, is known as *self-calibration*. Self-calibration is essentially identical to the technique of *hybrid mapping*, which is widely used in VLBI. See *self-calibration gain solution algorithm*; also see Tim Cornwell and Ed Fomalont's Lecture No. 9 in the *Third NRAO Synthesis Imaging Summer School* and the review paper by T. J. Pearson and A. C. S. Readhead [Image formation by self-calibration in radio astronomy, *Ann. Rev. Astron. Astrophys.*, **22** (1984) 97–130].

self-calibration gain solution algorithm — In self-calibration, the unknown *antenna/i.f. gains* $g_k(t)$ may be approximated by minimizing a functional $S(g_1, \dots, g_n)$ given by a weighted discrete l^p norm of the residuals:

$$S(\mathbf{g}) = \left(\sum_{1 \leq i < j \leq n} w_{ij} |\tilde{V}_{ij} - g_i \bar{g}_j V_{ij}|^p \right)^{1/p}.$$

Here \tilde{V}_{ij} is the visibility measurement obtained on the $i-j$ baseline (at a given instant), V_{ij} is the corresponding *model* visibility, and w_{ij} is a suitably chosen weight. Usually the g_k may be assumed not to vary too rapidly with time, so that one may minimize, instead, the functional

$$S(\mathbf{g}) = \left(\sum_{1 \leq i < j \leq n} w_{ij} |\langle \tilde{V}_{ij}/V_{ij} \rangle - g_i \bar{g}_j |^p \right)^{1/p},$$

where $\langle \tilde{V}_{ij}/V_{ij} \rangle$ is the time-average of the ratio of observed visibility to model visibility, over a time period during which the g_k may be assumed constant.

The AIPS implementation allows the choices $p = 1$ and $p = 2$. Choosing $p = 2$ yields the least-squares solution for \mathbf{g} . When one chooses $p = 1$, so that a weighted sum of the moduli of the residuals is minimized, the computed gain solutions are less influenced by wild data points, but there is some loss of statistical efficiency—*i.e.*, the least-squares solutions are superior when the distribution of measurement errors is well-behaved. (Probably the choice $p \simeq 1.2$ would offer a better compromise between efficiency and robustness). See [F. R. Schwab, Robust solution for antenna gains, *VLA* Scientific Memo. No. 136] for further details.

One may wish to solve only for the *antenna/i.f. phases* $\psi_k(t)$ rather than for the g_k if, for example, atmospheric phase

corruption is believed to be the dominant source of systematic error. In this case, one minimizes

$$S(\Psi) = \left(\sum_{1 \leq j < k \leq n} w_{jk} |\tilde{V}_{jk} - e^{i(\psi_j - \psi_k)} V_{jk}|^p \right)^{1/p},$$

or the version thereof incorporating time-averages.

Cornwell and Wilkinson [A new method for making maps with unstable radio interferometers, *Mon. Not. R. Astr. Soc.*, **196** (1981) 1067–1086] suggest adding to S terms which arise by assuming prior distributions for the g_k ; these “penalty terms” would be chosen so as to increase in magnitude as the solution parameter deviates from a prior mean which one might take, say, as the running mean of previous gain solutions. The widths of the prior distributions could be based on empirical knowledge of the behavior of the array elements. Such a modification can be useful when the array is composed of antenna elements of differing collecting area. This modification is used in order to constrain the moduli of the computed gains in one version of the AIPS task for self-calibration which is used primarily for VLBI data reduction (VSCAL).

Shannon sampling theorem — Suppose the complex-valued function f of the real variable t to be square-integrable, and assume that f is band-limited; *i.e.*, that its Fourier transform $\hat{f}(x) = \int_{-\infty}^{\infty} f(t) e^{2\pi i xt} dt \equiv 0$ for $|x| > a$. Then f is completely determined by its values at the discrete set of sampling points $n/2a$, $n = 0, 1, 2, \dots$, and f can be recovered via the *Shannon series* (also called the cardinal series)

$$f(t) = \sum_{n=-\infty}^{\infty} f\left(\frac{n}{2a}\right) \frac{\sin 2\pi a(t - n/2a)}{2\pi a(t - n/2a)}.$$

The series converges both uniformly and in the mean-square sense.

The Shannon series can be derived by expanding \hat{f} in a Fourier series, and then applying Fourier inversion—or it can be derived from the classical Poisson summation formula. It is sometimes referred to as Whittaker's cardinal interpolation formula or the Whittaker-Shannon sampling series, having first been studied in detail by E. T. Whittaker in 1915 and later introduced into the literature of communications engineering by Shannon in 1949. By the *Paley-Wiener theorem*, since f is band-limited, it can be analytically extended from the real line to the full complex plane, as an entire function of slow growth. The Shannon series, which converges for complex as well as real t , is one means of doing so. Whittaker referred to the series as “a function of royal blood in the family of entire functions, whose distinguished properties separate it from its bourgeois brethren.”

Suppose that $f(t)$ is “small” for $|t| > b$ (no nontrivial signal is both band-limited and time-limited). Then, assuming that b is integral, the number of terms in the Shannon series that really matter is $4ab$. This suggests that the space of “essentially band-limited” and “essentially time-limited” signals has dimension equal to the *time-bandwidth product* $4ab$. The precise sense in which this is so, together with a discussion of the *prolate spheroidal wave functions* (*q.v.*), which are relevant to the problem, is described by H. Dym and H. P. McKean [*Fourier Series and Integrals*, Academic Press, New York, 1972] and by David Slepian [Some comments on Fourier analysis, uncertainty and modeling, *SIAM Rev.*, **25** (1983) 379–393].

The multi-dimensional extension of the sampling theorem to rectangles implies that if an “unconfused” radio source $f(x, y)$ is confined to a small region of sky $|x| < x_0$, $|y| < y_0$ (radians), then it can be reconstructed unambiguously from a discrete set

G. Glossary

of visibility samples $\hat{f}(m\Delta u, n\Delta v)$, $m, n = 0, 1, 2, \dots$, with $\Delta u = 1/2x_0$ and $\Delta v = 1/2y_0$ wavelengths. See *cellsize* and *Nyquist sampling rate*. Other useful extensions of the sampling theorem—for example, to various multi-dimensional sampling configurations (e.g., 2-D hexagonal sampling lattices), to the case of stochastically jittered sampling, to derivative sampling (e.g., in 1-D, f can be recovered from samples of f and its derivatives through order r taken at intervals $(r+1)\frac{n}{2d}$), etc.—and sampling theorems for functions whose transforms of other than Fourier type are of *compact support*—are described in survey articles by A. J. Jerri [The Shannon sampling theorem—its various extensions and applications: a tutorial review, *Proc. IEEE*, **65** (1977) 1565–1596] and J. R. Higgins [Five short stories about the cardinal series, *Bull. (New Ser.) Amer. Math. Soc.*, **12** (1985) 45–89].

Shannon series — See *Shannon sampling theorem*.

shed — See *sub-task*.

SIVPSF — See *point spread function*.

slice — a one-dimensional cut across an *image*. E.g., the slice of a two-dimensional image f which passes through (x_0, y_0) and has orientation angle ϕ is the *subimage* h given by $h(t) = f(x_0 + t \cos \phi, y_0 + t \sin \phi)$. In AIPS, a slice may be excised from an image by issuing the *verb* command **SLICE**. Since AIPS deals only with digitized images, the program must interpolate to obtain data along the cut, except when the slice is taken along a row or column of the image.

slice file — in AIPS, an *extension file*, associated with an *image file*, in which a digitized *slice* (*q.v.*), or one-dimensional subimage, of the primary image is stored. In order to display a slice, one may issue the *verb* command **SL2PL**, which causes AIPS to read the contents of a slice file and generate a *plot file*.

snapshot — in earth-rotation aperture synthesis interferometry, an observation which is of such short duration that Earth's motion does not significantly enhance the *u-v coverage*, or a *map* derived from such a brief observation. Compare *full-synthesis map*.

For a thorough discussion of the use of the **VLA** in snapshot mode, see §5 of A. H. Bridle's Lecture No. 16 in the 1985 *Summer School Proceedings*.

software mount — a computer's reaction to the issuing of a command to it informing it that the *hardware mount* of some external storage module, such as a disk pack or a reel of magnetic tape, has occurred, and that the computer should open the channel of access to this module. See *hardware mount*.

sort order — the ordering of visibility measurements within a *u-v data file*. *Time-baseline order* is convenient for purposes of calibration, *baseline-time order* for data display, and so-called *x-y order* for gridding and subsequent mapping.

source editor — same as *text editor*. (Formerly, computers were used mainly for numerical computations and text editors primarily for the editing of program source code—hence the name *source editor*).

spatial resolution — In digital image analysis, this term refers rather imprecisely to the minimum size of details which can be discerned. The spatial resolution is determined by three factors: the inherent indeterminacy of whatever *image reconstruction* problem underlies the method by which the image was produced (and the properties of the image reconstruction algorithm which produced the image); the measurement noise;

and the *pixel size*—i.e., the size of the squares or the rectangles comprising the reconstruction matrix.

In radio interferometry, the inherent spatial resolution goes roughly in inverse proportion to the physical size scale D of the array (measured in wavelengths). For observations at a wavelength λ , the inherent spatial resolution, with a filled aperture, is essentially λ/D radians. However, with a synthesis array with large gaps in the *u-v coverage*, the effective resolution is somewhat coarser. Often, some measure of the spread of the central lobe of the *dirty beam* (say, the FWHM) is quoted as the spatial resolution. However, some reconstruction methods (e.g., the *regularization methods*) produce images in which the resolution of bright features may be much finer than that of dim features. This property of regularization methods may be viewed as either good or bad: *S/N* dependent spatial resolution complicates the interpretation of an image, but, on the other hand, one may gain additional contrast resolution—i.e., low surface-brightness features may become more readily discernible. An honest statement concerning the spatial resolution of an image must be based upon empirical knowledge of the reconstruction method that was used. See *super-resolution*.

spawn — See *sub-task*.

spheroidal function — an eigenfunction $\psi_{\alpha n}$ of a finite, weighted-kernel Fourier transform—more precisely, for given c and given $\alpha > -1$, one of the countably many solutions of the integral equation

$$\int_{-1}^1 e^{icnt} (1-t^2)^\alpha f(t) dt ;$$

equivalently, a solution of the differential equation $(1-\eta^2)f'' - 2(\alpha+1)\eta f' + (b - c^2\eta^2)f = 0$. The eigenfunction $\psi_{\alpha 0}$ of the equation above associated with the largest eigenvalue ν is termed the 0-order solution. The choice $\alpha = 0$ of weighting exponent yields the family $\{\psi_{0n} \mid n = 0, 1, 2, \dots\}$ of prolate spheroidal wave functions.

Weighted 0-order spheroidal functions $(1-\eta^2)^\alpha \psi_{\alpha 0}$ are optimal gridding convolution functions in the same sense that the *prolate spheroidal wave functions* (*q.v.*) are optimal, except that now the weighted concentration ratio

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

is maximized (see the paper by F. R. Schwab in the 1983 *Sydney Conference Proceedings*). The weighting exponent α is used to trade off the effectiveness of the aliasing suppression at the edge of the field of view, against that in the central region of the map. The choice $\alpha = 1$, with a *support width* of six *u-v* grid cells, yields an effective gridding convolution function, emphasizing aliasing suppression in the central region of the map; this function, ψ_{10} , with $c = 3\pi$, is the default function used in the AIPS mapping program. See *gridding convolution function*.

Stokes' parameters — the four coordinates relative to a particular basis for the representation of the polarization state of an electromagnetic wave propagating through space. Consider a wave propagating along the z -direction in a right-handed (x, y, z) Cartesian coordinate system. At a fixed point in space, let the instantaneous components of the electric field vector, in the x - and y -directions, be denoted by $E_x(t)$ and $E_y(t)$, respectively; and assume them to be stationary (in the weak sense, and square-integrable) stochastic processes. Form the matrix

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

Here, the bracketed expressions are expectation values, or correlation functions, in the lag variable τ , and $\hat{\cdot}$ denotes Fourier transform with respect to τ . Thus each element of S is a function of frequency ν . S is Hermitian (conjugate symmetric), owing to the stochasticity assumptions. The three Pauli spin matrices, together with the 2×2 identity matrix, form a basis for the algebra of 2×2 Hermitian matrices; i.e., each such matrix S can be represented in the form

$$\begin{aligned} S(\nu) = & \sigma_1(\nu) \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \sigma_2(\nu) \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \\ & + \sigma_3(\nu) \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} + \sigma_4(\nu) \begin{pmatrix} 0 & i \\ -i & 0 \end{pmatrix}. \end{aligned}$$

The four (real) coefficients, $\sigma_1, \dots, \sigma_4$, of the representation of S in this basis are called Stokes' parameters. They commonly are denoted by $I(\nu)$, $Q(\nu)$, $U(\nu)$, and $V(\nu)$, respectively. In other words,

$$S(\nu) = \begin{pmatrix} I(\nu) + Q(\nu) & U(\nu) + iV(\nu) \\ U(\nu) - iV(\nu) & I(\nu) - Q(\nu) \end{pmatrix},$$

with I , Q , U , and V real.

Stokes' parameter I measures the total intensity of the radiation field, Q and U the linearly polarized intensity, and V the circularly polarized intensity. I always is nonnegative. For a totally unpolarized wave, $Q = U = V = 0$; for a partially polarized wave, the ratio $\sqrt{Q^2 + U^2 + V^2}/I$ measures the total degree of polarization, $\sqrt{Q^2 + U^2}/I$ the degree of linear polarization, and $\frac{1}{2} \arctan \frac{U}{Q}$ the orientation angle of the linearly polarized component. $Q + iU$ is called the complex linear polarization. The IAU and IEEE orientation/sign conventions have the z -axis directed toward the observer, the x -axis directed north, and a $+i$ in the argument of the exponential kernel of the FT. Positive V corresponds to right circular polarization, and conversely. The polarization response of an interferometer can be described by forming the so-called cross-spectral density matrix, which is like the S above but is formed from measurements of the electric field taken at two points in space. For further details, including a description of the polarization response of an interferometer, for various feed configurations, see Carl Bignell's Lecture No. 6 in the 1982 *Summer Workshop Proceedings*.

Stokes' visibility functions — Stokes' visibility functions, V_I, V_Q, V_U , and V_V , are the Fourier transforms (FT's) of the radio brightness (spatial) distributions of Stokes' parameters, $I(x, y)$, $Q(x, y)$, $U(x, y)$, and $V(x, y)$. (Here, $V_I = \hat{I}$, $V_Q = \hat{Q}$, etc., where $\hat{\cdot}$ denotes FT.)

For a radio interferometer with ideal circularly polarized feeds, the relations between Stokes' visibility functions and the visibilities, V_{RR} , V_{LL} , V_{RL} , and V_{LR} , obtained by correlating right circular response with right, left with left, etc., are $V_I = \frac{1}{2}(V_{RR} + V_{LL})$, $V_Q = \frac{1}{2}(V_{LR} + V_{RL})$, $V_U = \frac{i}{2}(V_{LR} - V_{RL})$, $V_V = \frac{1}{2}(V_{RR} - V_{LL})$. Note that each of Stokes' visibility functions is *Hermitian*. On the assumption that circular polarization is absent (i.e., that $V(x, y) \equiv 0$), V_{RR} is equal to V_{LL} , and both are Hermitian.

Components of the systematic errors affecting visibility measurements are i.f.-dependent; hence **VLA** u - v data files usually do not contain Stokes' visibilities, but rather V_{RR} , V_{LL} , V_{RL} , and V_{LR} —as these are what is required for calibration purposes. Stokes' visibility functions generally are constructed only within the mapping programs. (But the AIPS visibility data format is designed to accommodate either type of visibility function, and the mapmaking tasks are able to recognize the form of their input data and deal with them appropriately.)

subimage — in AIPS parlance, any linear, rectangular, or hyper-rectangular section of an *image*.

sub-task — a task, or computer program, whose execution is initiated by the action of another program. The act of initiating the execution of the sub-task is called *task shedding* or *task spawning*. See *task*.

super-resolution — The problem of image reconstruction in radio interferometry is one of finding an approximation to an unknown function f (generally assumed to be of *compact support*) from *partial knowledge* of its Fourier transform \hat{f} —i.e., from a finite number of measurements of the visibility. Any of the techniques which are applied to the problem—the *Högbom Clean algorithm*, the *regularization method*, etc.—may be thought of as methods of smoothing, interpolating, and extrapolating the noisy measurements. *Super-resolution* is a term which refers to the extrapolation aspect: Cautious extrapolation yields an image whose *spatial resolution* is $\approx \lambda/D$, where D is the diameter of the largest centered region in the u - v plane which has been reasonably well sampled. Less cautious extrapolation yields super-resolution; spurious detail appears as caution is abandoned.

Super-resolution in a *Clean map* is effected by choosing an artificially narrow *Clean beam*. With regularization methods (in image reconstruction, and more generally), super-resolution comes about by choosing a small value of the *regularization parameter*. The spatial resolution achieved by a regularization method may be signal-to-noise dependent—bright features may be super-resolved, and dim ones not.

support — The closure of that subset of the domain of definition of a function f (or of a generalized function, or distribution) on which the function assumes a nonzero value is called the *support* of the function, and is denoted by $\text{supp}(f)$. I.e., $\text{supp}(f) = \{x \mid f(x) \neq 0\}$.

For example, the support of the function $f(x) = x$ is the whole real line, even though $f(0) = 0$. And the support of

$$f(x, y) = \begin{cases} 1, & x^2 + y^2 < 1, \\ 0, & \text{otherwise,} \end{cases}$$

is the *closed unit disk*, $\{(x, y) \mid x^2 + y^2 \leq 1\}$.

In Euclidean space, a function f whose support is bounded—i.e., such that $f \equiv 0$ “far-out”—is said to be of *compact support*. The Fourier transform of a nontrivial function of compact support (such as a *u*-*v* *measurement distribution* or a *gridding convolution function*) cannot itself be of compact support; i.e., it has “sidelobes” extending to infinity.

support width — of a function whose *support* is a rectangle or a hyper-rectangle (e.g., the Fourier transform of a band-limited function), the linear measure of one of the edges of its *support*.

SVPSF — See *point spread function*.

Synthesis Imaging in Radio Astronomy — A collection of lectures from the 1988 (Third) NRAO Synthesis Imaging Summer School edited by R. A. Perley, F. R. Schwab and A. H. Bridle. (Astronomical Society of the Pacific Conference Series, Volume 6 (1989)). A very useful reference book for the reduction of radio interferometric data. This volume supersedes the proceedings from the earlier workshops.

synthesized beam — in radio interferometry, the *beam*—but always ignoring instrumental effects. Hence, the synthesized beam is fully determined by the *u*-*v sampling distribution*, the *u*-*v*

G. Glossary

weight function, the *u-v taper function*, and the *gridding convolution function*. See *beam*.

tape blocking efficiency — Data are stored on magnetic tape in units of *blocks*. An *inter-record gap*—essentially wasted space—separates one block from the next. The *tape blocking efficiency*, or the fraction of unwasted space, is the ratio

$$\frac{\text{block length}}{\text{recording density}} \cdot \frac{\text{block length}}{\text{recording density}} + \text{length of an inter-record gap}$$

The length of an inter-record gap is about $\frac{3}{4}$, $\frac{3}{5}$, and $\frac{3}{10}$ inch at recording densities of 800, 1600, and 6250 bpi, respectively.

taper — See *u-v taper function*.

task — used in two senses: 1) the execution of a computer program and 2) the program itself. Thus, if two computer users are (independently) running the same program at the same time, it may be said either that two tasks are running, or that two incarnations of the same task are in existence. A *sub-task* (*q.v.*) is a task whose execution is initiated by the action of another program. Many of the more complicated and the more specialized functions of AIPS are accomplished by the action of sub-tasks shed by the AIPS program. (Simpler functions are invoked by the issuance of *verb* commands—see *POPS symbols*.)

t-b order — See *time-baseline order*.

TEK screen — a cathode ray tube (CRT) terminal and display device appropriate for pictorial display of data, in the form of contour plots, graphs, etc., as well as for display of textual data. The Tektronix company's Model 4012 terminal (with a green P4 phosphor, hence the synonymous term *green screen*) is the canonical device of this type. The "make copy" button on this device can be used to produce a copy, on paper, of the image shown on the CRT screen. Each of the NRAO's AIPS data reduction computers is outfitted with a *TEK screen*.

TEK4012 — same as *TEK screen*.

Telex 6250 tape drive — a model of tape drive used on the [VLA](#) Vaxes, capable of operation at 1600 and 6250 bpi.

terminal page — Many modern computer terminals contain a semiconductor memory with a capacity of several CRT screen loads (≈ 24 lines) of character data. A *terminal page* is a unit of one screen load of such data. Certain terminal keys allow one to cause data which previously appeared on the CRT screen to reappear—this feature is called *terminal scroll* (*q.v.*). A typical terminal at the NRAO has three terminal pages of memory.

terminal scroll — that feature present on certain models of computer terminals which allows data which previously appeared on the CRT screen to be made to reappear. Often, depressing one key on the terminal will cause earlier information to reappear line-by-line (this is termed *line scroll*), while the action of another key will cause a whole earlier screen load to reappear (this is termed *page scroll*).

text editor — a computer program designed for the creation, manipulation, and modification of computer files containing textual data such as reports, documentation, alphanumeric command lines, and program source code. Generally, one or more text editors are supplied by the computer manufacturer. Three text editors are in widespread use on the Vax—*SOS*, *EMACS* and *EDT*. *vi*, *edt* and *emacs* are used on NRAO's Convex computers. See *line editor* and *screen editor*.

text file — a computer data file containing only textual data, as might be written by a *text editor* (*q.v.*). Programs such as the AIPS tasks sometimes write messages, especially progress report messages, into a text file—see *message file*.

Third NRAO Synthesis Imaging Summer School — The 1988 Summer School on Synthesis Imaging which was held in Socorro, New Mexico in June 1988. The lectures were formally published in *Synthesis Imaging in Radio Astronomy*.

thrashing — See *memory thrashing*.

time-baseline order — An ordered set of visibility measurements $\{V_{ij}(t_k) \mid 1 \leq i < j \leq n, k = 1, \dots, l\}$ recorded with an n element interferometer at times $t_1 < t_2 < \dots < t_l$ is said to be in *time-baseline order* if the ordering is such that all of the data obtained at time t_1 , sorted into the canonical ordering by baseline, occur first, followed by the data obtained at time t_2 , again ordered canonically, etc., etc. (The canonical ordering by baseline is the order $V_{12}, V_{13}, \dots, V_{1n}, V_{23}, \dots, \dots, V_{n-1,n}$.) Compare *baseline-time order*.

Time-baseline ordering of a *u-v data file* is convenient for calibration purposes. The AIPS task for self-calibration requires that its input *u-v* data file be time-baseline ordered.

time smearing — in a radio interferometer map, the space-variant broadening of the *point spread function* (or *beam*) which is due to time averaging of the data. When, for example, the visibility data along a *u-v* track are averaged, with equal weight, over time intervals of width Δt sec., the visibility amplitude of a point source is reduced by a factor $\approx \frac{\sin \gamma}{\gamma}$ —where $\gamma \equiv \pi(u'x + v'y + w'z)\Delta t$, where the primes denote the time rate of change of the spatial frequency coordinates (u, v, w) along the track (wavelengths/sec.), and where (x, y, z) denotes the direction cosines of the location of the point source with respect to the *phase tracking center*. For further details, see A. R. Thompson's Lecture No. 2 and Alan Bridle and Fred Schwab's Lecture No. 13 in the *Third NRAO Synthesis Imaging Summer School*. Compare *bandwidth smearing*.

trackball — a spherical ball mechanism, about the size (10 cm., or so, in diameter) of a tennis ball, which may be oriented manually by the interactive user of a television display device such as the I^2S . The ball can be rotated about any axis, and its orientation, which is sensed by the computer, typically is used to control the enhancement or the coloration of the displayed data (i.e., to control the *TV transfer function(s)*), or to position the *TV cursor*, in order to point out to a program features in the displayed image which are of particular interest.

trackball button — On the unit which houses the trackball for the I^2S Model 70 TV display device are the four *trackball buttons*, labeled A, B, C, and D. These are switches that are used, in conjunction with the display routines, to exert additional control over the TV display. Occasionally these buttons are put to other use in AIPS, such as stopping the Clean deconvolution program.

transfer function — a transform which can be used to describe the output of a device (say, an electrical transducer) as a function of the input to the device. See *TV look-up table*.

TRC — *top right corner*, the corner of an image diagonally opposite the **BLC**. See *m × n map*.

true color display — a type of *false color display*, (*q.v.*).

TU77 tape drive — a model of tape drive used on the NRAO's Vaxes, capable of operation at 800 and 1600 bpi.

TU78 tape drive — a model of tape drive used on the [VLA](#) Vaxes, capable of operation at 1600 and 6250 bpi.

TV blink — a feature of a computer-controlled TV display device, such as the I^2S , intended to facilitate the comparison of a pair of images stored on two different *image planes*. The TV display is made to alternate between the two images. The AIPS implementation of blinking allows the user, by manipulating the *trackball*, to control the rate of alternation and the fraction of time that each image is displayed.

TV cursor — See *crosshair*.

TV image catalog — See *image catalog*.

TV look-up table — a memory within the control unit of a TV display device which is used for storage of the *transfer functions* controlling the intensity of the display, as a function of pixel value. Within AIPS, the transfer functions may be altered through the use of interactive verbs and manipulation of the *trackball*.

TV roam — a feature of a computer-controlled TV display device such as the I^2S which allows contiguous parts of a single large image, stored on more than one *image plane*, to be displayed as if the image were stored on a single, larger image plane. On the I^2S unit, the portion of the image to be displayed on the TV screen is selected by manipulation of the *trackball*. See *image plane*.

TV scroll — a feature of a computer-controlled TV display device such as the I^2S which allows the display of an image stored on a single *image plane* to be moved about the display screen. This feature, which also is called panning, commonly is used in combination with the *TV zoom* capability. On the I^2S unit, the scroll ordinarily is controlled by manipulation of the *trackball*. Compare *TV roam*.

TV zoom — a magnification feature of a computer-controlled TV display device such as the I^2S . On the I^2S , the three available magnification factors (which multiply the linear dimensions of the original display of the image by a factor of 2, 4, or 8) generally are selected by depressing one of the *trackball buttons*. Since the magnification is achieved by pixel replication (i.e., by piecewise linear interpolation)—rather than by a smooth interpolation—the visual impression may be somewhat displeasing. The entire magnified image may not fit on the TV screen, so zoom usually is used in combination with the *TV scroll* feature.

uniform weighting — A *dirty map* obtained by computing the inverse Fourier transform (FT) of a weighted *u-v measurement distribution* in which each visibility sample has been weighted in inverse proportion to the local density of the *u-v coverage* is said to have been computed using *uniform weighting*. When a radio map is computed via the fast Fourier transform algorithm, uniform weighting may be achieved by computing normalized discrete convolution summations $\sum_{i=1}^N C(u - u_i, v - v_i) \bar{V}_i / N$, where (u, v) denotes the spatial frequency coordinates of a given *u-v* grid cell, where C is an appropriately chosen *gridding convolution function*, and where the \bar{V}_i are the N visibility measurements obtained at positions (u_i, v_i) in some neighborhood of (u, v) , the size of which is determined by the *support* of C . The uniform weighted map is given by the inverse discrete FT of data interpolated and smoothed in this manner, onto the lattice points of a rectangular grid. So-called *natural weighting* is achieved by using unnormalized convolution sums, rather than by dividing

by N . The AIPS mapmaking tasks use a weighting scheme which is slightly more complicated than that described here.

Since the density of *u-v* coverage typically is greater in the inner regions of the *u-v* plane, a map computed using uniform weighting has finer *spatial resolution* than one computed with natural weighting. With natural weighting, low surface-brightness extended features may be more easily discernible than with uniform weighting. Essentially the same effect can be achieved with uniform weighting, when accompanied by use of a *u-v taper function*.

UNIX — a “universal” computer operating system developed at the Bell Telephone Laboratories. Its virtue is that program packages such as AIPS—once having been made to run under one UNIX-based operating system—ought to run on any other such system, even on a computer of different manufacture, with no alterations. Many Vaxes operate under UNIX, though not the NRAO’s. The Convexes C-1 in Charlottesville and at the AOC operates under UNIX. See *operating system*.

user-coded task — an AIPS *task* written by a user, rather than by a professional programmer or a member of the AIPS programming group. One of the design goals for AIPS, not yet fully realized, is that it should be relatively easy for a user who is not an experienced programmer to write an AIPS task suited to his own needs—i.e., that it should be fairly simple for him to make some sense of the AIPS database, and to get at his data and manipulate it as he sees fit. The AIPS task named **FUDGE** is intended to serve as a paradigm for user-coded tasks for manipulation of *u-v data files*; two other tasks, **TAFFY** and **CANDY**, are paradigms for *image file* manipulation. A useful reference is the manual by W. D. Cotton and a ‘cast of AIPS’ [*Going AIPS! A Programmers Guide to the NRAO Astronomical Image Processing System*, NRAO, Charlottesville, VA, 1990].

The addition to AIPS of new *verbs*, and modification of the functioning of existing verbs, requires modifying the AIPS program itself; this is best left to the AIPS programming group.

u-v coverage — the support of the *u-v sampling distribution* (*q.v.*). Also see *conjugate symmetry*.

u-v data file — in AIPS, a *primary data file* designed to accommodate the measurements of the visibility function of a radio source.

u-v data flag — In an AIPS *u-v data file*, each visibility measurement is accompanied by a real-valued weight, which ordinarily is (positive and) proportional to the length of the integration period over which the measurement was obtained. A non-positive weight represents a *u-v data flag*, which signifies that the visibility measurement ought to be ignored. See *flagging* and *clipping*.

u-v FITS format — an extension of the *FITS format* (originally designed for the interchange of image data) to accommodate radio interferometer visibility data [E. W. Greisen and R. H. Harten, An extension of FITS for groups of small arrays of data, *Astron. Astrophys. Suppl. Ser.*, **44** (1981) 371–374]. See *FITS format*.

u-v measurement distribution — in radio interferometry, a linear combination of shifted Dirac δ -functions, one located at the position in the *u-v* plane of each visibility measurement, and each weighted by the visibility measurement obtained at that location. Denoting the *u-v* coverage by $\{(u_i, v_i)\}_{i=1}^n$, the visibility function by V , and the measured visibility by \bar{V} , the (two-dimensional) *u-v* measurement distribution S is given by

G. Glossary

$S(u, v) = \sum_{i=1}^n \tilde{V}(u_i, v_i) \delta(u - u_i, v - v_i)$. Compare *u-v sampling distribution*.

This definition may be modified to incorporate two types of weight function, yielding a *weighted* and/or *tapered* measurement distribution—see *u-v taper function* and *u-v weight function*.

The visibility measurements $\{\tilde{V}(u_i, v_i)\}$ are not actual samples of V , but rather are error-corrupted samples of a function which represents some sort of *local average* of the visibility—this is a distinction which it is worthwhile to note, and then to ignore. Various systematic errors affecting the measurements may be corrected by proper calibration—see *antenna/i.f. gain* and *instrumental polarization*.

u-v sampling distribution — in radio interferometry, a linear combination of shifted Dirac δ -functions, one located at the position in the *u-v* plane of each visibility measurement. Sometimes termed *u-v transfer function*. See *beam*.

If $\{(u_i, v_i)\}_{i=1}^n$ (the *u-v coverage*) is the set of spatial frequency coordinates at which the source visibility has been sampled, then the (two-dimensional) *u-v sampling distribution* S is given by $S(u, v) = \sum_{i=1}^n \delta(u - u_i, v - v_i)$.

Occasionally the term *u-v sampling distribution* is used in the same sense as the term *u-v measurement distribution* (*q.v.*).

u-v taper function — an even, real-valued weight function (typically, an elliptical Gaussian), smooth and peaked at the origin, which may be incorporated into the definition of *u-v measurement distribution* or *u-v sampling distribution*, above, serving to control the spatial resolution of the radio map or the beam; i.e., to enhance the response to extended features in the radio source brightness distribution by giving relatively higher weight to the measurements at short *u-v* spacings. Compare *u-v weight function*.

u-v transfer function — same as *u-v sampling distribution*, but always explicitly incorporating any *u-v weight function* or *u-v taper function*.

u-v weight function — a real-valued function which may be incorporated in the definition, above, of *u-v measurement distribution* or *u-v sampling distribution*, serving to weight each measurement either according to an estimate of the statistical measurement error, or according to the local density of sampling, or both. Compare *u-v taper function* and see *uniform weighting*.

Varian printer — an electrostatic printer/plotter manufactured by the Varian Corp.

Variational Method — the name which applies to Tim Cornwell's AIPS implementation (in the program VM) of the *maximum entropy method*, to solve the image deconvolution problem $g = b * f$, where g and b are given, and f is unknown. The regularizing term $S(\tilde{f})$ (see *regularization method*), a function of the computed approximate solution \tilde{f} , is given by the negative of an entropy expression, of the form

$$H(\tilde{f}) = - \int_A \tilde{f}(x) \log \frac{\tilde{f}(x)}{h(x)} dx .$$

Here A denotes the (assumed known) support of f , and h is a prior estimate of f ; when $h \equiv \text{constant}$, this agrees with the standard formulation of the maximum entropy method. A weighted sum $\chi^2(\tilde{f}) + \lambda S(\tilde{f})$ of a χ^2 error term and S is minimized, and the regularization parameter λ is chosen so that the r.m.s. residual corresponding to the final iterate is approximately equal to an input value. For optical data the χ^2

term is taken as $\|g - b * \tilde{f}\|^2$, whereas for radio data the χ^2 term is evaluated in the visibility domain, where the measurement errors may more properly be assumed to be statistically independent. Also, $\int_A \tilde{f}$ is constrained to be near an estimate of the *zero-spacing flux* which is supplied by the user. The minimization is done using a Newton-type method, with a diagonal approximation to the Hessian of the objective function and intricate control of the steplength. In terms of execution speed, this method is competitive with the *Clark Clean algorithm*—at least in the case of large objects of complex structure observed with the VLA—and superior results usually are obtained for this class of objects. See [T. J. Cornwell, Deconvolution with a maximum entropy type algorithm, VLA Scientific Memo. No. 149].

verb — See *POPS symbols*.

Versatec printer — an electrostatic printer/plotter manufactured by the Versatec Corp., and used on the NRAO's AIPS computer systems.

Very Long Baseline Interferometry — Techniques and Applications — Proceedings of the NATO Advanced Study Institute held at Castel S. Pietro Terme, Bologna, Italy in 1988. Edited by M. Felli and R. E. Spencer. Kluwer Academic Publishers, Dordrecht (1989). This volume contains much useful information on the planning and execution of VLBI observations as well as on the reduction of VLBI data.

vi — a moderately sophisticated text editor (a *screen editor*) used on computers which run the UNIX operating system. See *text editor*.

virtual memory page — on a computer running under a virtual memory operating system, one unit of *virtual memory storage*. At a typical Vax installation, the size of a virtual memory page is 512 bytes.

virtual memory page swapping — on a computer running under a virtual memory operating system, the action (initiated automatically by the operating system) of reading new virtual memory pages into the physical memory, and storing on disk (i.e., in the *virtual memory*) the data which thus have been displaced. Each occurrence of the displacement of a memory page is referred to as a *page fault*. See *memory thrashing*.

virtual memory storage — computer storage—typically disk storage—in an area apart from the *physical memory* of a computer. Access to virtual memory storage is controlled by the operating system, in a way intended to give the programmer the illusion that a large amount of physical memory is present. Access to virtual memory may be much slower than access to physical memory, and the operating system may incur a significant amount of overhead in managing the virtual memory. See *memory thrashing*.

visibility phase tracking center — In a correlating-type radio interferometer usually the *fringe stopping center* and the *delay tracking center* coincide. When this is the case, both are referred to as the *visibility phase tracking center*.

VM — See *Variational Method*.

VMS — (Virtual Memory System) the operating system used on the NRAO's Vax computers. See *virtual memory storage* and *operating system*.

wedge — a legend, or scale—generally in the form of a bar graph with gradations in *intensity* and *chromaticity*—which

may be displayed adjacent to a photographic or video display of a digitized *image*. The wedge is a visual representation of the *transfer function* that was used in generating the display. The wedge is either colored or gray, depending on whether the display is a *pseudo-color display* or a *gray-scale display*.

Note that a *false color display* would require more than one wedge (or a multi-tiered wedge) to display the several transfer functions, as well as an additional wedge to display the possible color mixtures.

window Clean — an application of the *Högbom Clean algorithm*, with an explicit specification, by the user, of the Clean window. Generally the user should specify a Clean window whenever it is possible to make a reasonably valid and restrictive estimate of the *support* of the true radio source brightness distribution. At the termination of the algorithm, it is prudent to examine a display of the residual map for the presence of large residuals outside of the Clean window; their presence could suggest that an inappropriate window was selected. See *Clean window*.

working set size — on a computer running under a virtual memory operating system, the amount of *physical memory* allocated to a task. Any program memory requirement in excess of the working set size is relegated to *virtual memory storage*. At a typical Vax installation, the working set size is set at $\frac{1}{4}$ or $\frac{1}{2}$ megabyte.

x-y order — An ordered set of visibility samples $\{V(u_i, v_i, w_i)\}$ arranged according to descending absolute value of the spatial frequency coordinate u — i.e., with $|u_1| \geq |u_2| \geq \dots \geq |u_n|$ — is said to be in *x-y order*.

x-y order is a convenient ordering for the operation of gridding convolution; hence the AIPS mapping tasks require that their input *u-v data files* be sorted accordingly. See *sort order*.

Y-routine — in AIPS, a subroutine designed to aid in the use of a specific model of TV display device, such as the I²S Model 70. AIPS requires a relatively small core of Y-routines implementing basic TV display functions; complicated display functions then are accomplished by combining these basic functions that are supposed to be common to many models of TV display device. At present there are approximately 25 Y-routines for use at those AIPS installations equipped with an I²S. Compare *Z-routine*.

zero-spacing flux — The visibility $V(u, v) \equiv \hat{f}(u, v)$ ($\hat{}$ denotes Fourier transform) of a source brightness distribution f in a neighborhood of $u = v = 0$ is inaccessible to an interferometer composed of elements of finite collecting area. The *zero-spacing flux* is equal to the total, or integrated flux density of the source—i.e., it is given by $V(0, 0) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) dx dy$. Because the hole in the *u-v coverage* in the neighborhood of the origin may be fairly large, image reconstruction methods, such as the *Högbom Clean algorithm*, may do a poor job, within this central region, of interpolating the measured data. This frequently is manifested by the appearance of a *negative bowl artifact*—a negative ‘baseline’ beneath the reconstruction of f —owing to the reconstruction method having underestimated the zero-spacing flux. The *Variational Method* for maximum entropy reconstruction requires that the user supply an estimate of $V(0, 0)$. The Clean algorithm, too, may benefit if a datum at $u = v = 0$ is included when the *dirty map* is constructed.

A zero-spacing estimate can be derived from single-dish measurements. Providing a proper estimate is difficult, because of contamination of single-dish measurements by ‘confusing

sources.’ The estimate ought to correspond to a telescope with the same primary beam response as the array elements; and it is not just a single datum $V(0, 0)$ which is missing, but rather a region—so proper weighting of the zero-spacing information is tricky. See Tim Cornwell and Robert Braun’s Lecture No. 8 in the *Third NRAO Synthesis Imaging Summer School*.

zoom — See *TV zoom*.

Z-routine — in AIPS, a subroutine—generally designed to perform some routine, often needed function—written for a specific model of *host computer* or for a specific host computer operating system. The implementation of certain basic functions, especially those for file access and file management, generally is machine dependent and operating system dependent. The typical AIPS installation requires 50–100 Z-routines. Compare *Y-routine*.

1978 Groningen Conference Proceedings — *Image Formation from Coherence Functions in Astronomy. Proceedings of IAU Colloquium No. 49 held at Groningen, the Netherlands, August 10–12, 1978*, edited by C. van Schooneveld, D. Reidel, Dordrecht, Holland, 1979—contains many papers on aperture synthesis techniques, including some of the early papers on *hybrid mapping*.

1982 Summer Workshop Proceedings — *Synthesis Mapping. Proceedings of the NRAO-VLA Workshop held at Socorro, New Mexico, June 21–25, 1982*, edited by A. R. Thompson and L. R. D’Addario, NRAO, Green Bank, WV, 1982—a collection of the fifteen lectures which comprised this short course on aperture synthesis techniques—a useful introduction to *VLA* data reduction methods.

1983 Sydney Conference Proceedings — *Indirect Imaging: Measurement and Processing for Indirect Imaging. Proceedings of an International Symposium held in Sydney, Australia, August 30–September 2, 1983*, edited by J. A. Roberts, Cambridge Univ. Press, Cambridge, 1984—contains a number of interesting papers on aperture synthesis techniques.

1985 Summer School Proceedings — lecture notes from the second NRAO summer short course on radiointerferometric imaging (in preparation). This volume supersedes the 1982 *Summer Workshop Proceedings*.

1988 Summer School Proceedings — lecture notes from the third NRAO Summer School on radio interferometric imaging. The lectures have been published as *Synthesis Imaging in Radio Astronomy* (q.v.).

4012 — See *TEK screen*.

I INDEX

— 1 —

1978 Groningen Conference Proceedings	G - 21
1982 Summer Workshop Proceedings	G - 21
1983 Sydney Conference Proceedings	G - 21
1985 Summer School Proceedings	G - 21
1988 Summer School Proceedings	G - 21
1990 AIPS Site Survey	1 - 1

— 2 —

2-Gigabyte limit	9 - 1
----------------------------	-------

— 3 —

3C138	4 - 35, 35
3C286	4 - 35, 35
3C48	4 - 35

— 4 —

4012	G - 21
----------------	--------

— A —

ABORT	3 - 1
ABOUT	3 - 11, 11
	6 - 5
ABOUT ADVERB	13 - 1
ABOUT ANALYSIS	13 - 10
ABOUT AP	13 - 13
ABOUT ASTROMETRY	13 - 14
ABOUT BATCH	13 - 14
ABOUT CALIBRATION	13 - 15
ABOUT CATALOG	13 - 20
ABOUT COORDINATES	13 - 23
ABOUT EDITING	13 - 23
ABOUT EXTENSION APPLICATIONS	13 - 24
ABOUT FITS	13 - 25
ABOUT GENERAL	13 - 25
ABOUT HARDCOPY	13 - 28
ABOUT IMAGE	13 - 29
ABOUT IMAGE-UTILITIES	13 - 28
ABOUT IMAGING	13 - 29
ABOUT INFORMATION	13 - 32
ABOUT INTERACTIVE	13 - 33
ABOUT MODELING	13 - 35
ABOUT OBSOLETE	13 - 36
ABOUT ONED	13 - 36
ABOUT OOP	13 - 37
ABOUT OPTICAL	13 - 38
ABOUT PARAFORM	13 - 38
ABOUT PLOT	13 - 39
ABOUT POLARIZATION	13 - 42
ABOUT POPS	13 - 42
ABOUT PROCEDURE	13 - 45
ABOUT PSEUDOVERB	13 - 47
ABOUT RUN	13 - 47
ABOUT SINGLEDISH	13 - 48
ABOUT SPECTRAL	13 - 49
ABOUT TABLE	13 - 50
ABOUT TAPE	13 - 51
ABOUT TASK	13 - 52
ABOUT TV	13 - 62
ABOUT TV APPLICATIONS	13 - 64
ABOUT UTILITY	13 - 66
ABOUT UV	13 - 67
ABOUT VERB	13 - 72
ABOUT VLA	13 - 78
ABOUT VLBI	13 - 79
ACCOR	9 - 36
ACFIT	9 - 44, 44
ACLIP	4 - 16
ACSCL	9 - 40, 41
ADDBEAM	10 - 1
ADDISK	12 - 5, 14
adverb	3 - 2
	12 - 1, 8
	13 - 1
	G - 1
AFARS	7 - 10
AGAUS	6 - 24
	7 - 10
	8 - 14
AHIST	7 - 15
AIPS	1 - 1, 1, 1
AIPS*	1 - 1
AIPS	3 - 1
	12 - 1
AIPS Letter	3 - 11
AIPS monitor	G - 1
aliased response	G - 1
aliasing	G - 1
ALLDEST	11 - 2
ALTDEF	8 - 9
	10 - 1
ALTSW	8 - 9
ALU	G - 1
analysis	7 - 12, 15
	8 - 14
	10 - 19, 20, 22, 22
ANBPL	4 - 32, 32
	6 - 3, 7
ANTAB	9 - 2, 32, 32, 67
	C - 5, 10, 11
antenna coordinates	4 - 29

antenna file	4 – 1	baseline-time order	G – 2
.	G – 1	BASFIT	4 – 29
antenna/i.f. gain	G – 1	batch	12 – 5, 5, 5
antenna/i.f. phase	G – 1	13 – 14
antenna residual delay	G – 1	batch editor	G – 2
antenna residual fringe rate	G – 1	batch job	G – 2
ANTNUM	9 – 20, 21	batch processor	G – 2
AP	13 – 13	batch queue	G – 2
.	G – 1	BATCLEAR	12 – 5
AP-120B array processor	G – 1	BATEDIT	12 – 5
APCAL	9 – 35, 41, 41, 41	BATLIST	12 – 5
APCLN	5 – 24	BATQUE	12 – 5
.	6 – 19, 24	“battery-powered” Clean algorithm	G – 2
.	8 – 6	BCOMP	5 – 11
APROPOS	3 – 11	BDAPL	E – 20
array processor	13 – 13	BDF2AIPS	4 – 46
.	G – 1	E – 1, 1, 11
array processor microcode	G – 1	L – 1
associated file	G – 2	BDFLIST	E – 1, 1
astrometry	13 – 14	L – 1
ATLOD	4 – 2	beam	G – 3
auto re-boot	G – 2	beam patch	G – 3
AVEOT	3 – 15	beam squint	G – 3
AVFILE	3 – 15	beam-switched continuum	10 – 1, 4, 5, 15, 15, 15
.	8 – 1	10 – 15, 15, 21
AVSPC	4 – 41	BLANK	6 – 24
.	8 – 1, 1, 1	7 – 5
.	9 – 65	8 – 14
.	C – 4	blanked pixel	7 – 3, 5
.	8 – 14
.	G – 3

— B —

backup	F – 1, 4, 4, 6	BLAPP	9 – 56, 56
back-up	G – 2	BLC	G – 3
Backups	11 – 2	BLCAL	9 – 64
BADDISK	3 – 10	BLCHN	E – 20
BAMODIFY	12 – 5	9 – 64
bananas	2 – 12, 12	E – 20
.	3 – 20	BLEW CORE!	12 – 9
.	6 – 28, 28	BLING	9 – 2, 56
.	7 – 17, 17, 18, 18, 18	blink	G – 3
.	8 – 17, 17, 18, 18, 18	BLOCKING	F – 4, 5
.	9 – 73, 73, 74, 74, 74	blotch	7 – 4, 5
.	11 – 6	BLSUM	6 – 4, 12, 24
.	12 – 23, 24, 24, 24	7 – 4, 8
.	13 – 81, 81, 82, 82, 82	8 – 14, 14
.	A – 6, 6, 6	BMAJ	5 – 11
.	B – 8, 8	BMIN	5 – 11
.	C – 13, 13, 14, 14, 14	boot	G – 3
.	D – 7, 8, 8, 8, 8	BOT marker	G – 3
.	E – 22, 22, 23, 23	BOXES	5 – 14, 16
.	L – 12, 12, 12	7 – 8
.	F – 6, 6	BOXFILE	5 – 6, 14, 16
bandpass	4 – 42, 42, 42	BPA	5 – 11
.	9 – 2, 40, 40, 43, 63, 63	BPASS	4 – 42, 42, 42
bandwidth smearing	G – 2	9 – 40, 63
bandwidth synthesis	8 – 1	E – 6
.	G – 2	L – 1
baseline error	4 – 29, 29		

BPEDT	4 – 42	calibration file	F – 3
.	6 – 24	CALRD	4 – 13
.	9 – 40, 63	CANDY	12 – 19, 21
.	E – 6	CAPLT	6 – 7
BPERR	4 – 42	carriage-return key	9 – 23
bpi	G – 3	casa	G – 3
BPLOT	4 – 42	CATALOG	1 – 1
.	6 – 15	catalog entry	3 – 4
.	E – 12, 20	catalog file	5 – 2
BPSMO	4 – 42	catalog slot	G – 3
BPWAY	5 – 7	CCEDT	5 – 22
.	E – 20	CCFND	9 – 68
BPWGT	E – 20	CCMOD	5 – 22
broadband mapping technique	G – 3	CCMRG	7 – 16
BSAVG	10 – 21	CCNTR	5 – 22, 26
BSCAN	9 – 38	CCRES	6 – 9
BSCLN	10 – 15	CCSEL	5 – 11, 24
BSCOR	10 – 15, 15	CELGAL	7 – 15
BSGEO	10 – 15, 15	cell-averaging	5 – 22, 29
BSGRD	10 – 15, 15	cellsize	9 – 68
BSMOD	10 – 13	CENTR	8 – 9
BSPRT	9 – 56	cereal bowl map defect	G – 4
BSTST	10 – 15	characteristic function	G – 4
b-t order	G – 3	character strings	12 – 8
bug	G – 3	CHARMULT	6 – 21
button A	2 – 11	CHKFC	5 – 14
button B	2 – 11	CHKNAME	12 – 9
button C	2 – 11	chromaticity	G – 4
button D	2 – 11	C.I.E. chromaticity diagram	G – 4
byte	G – 3	CIRCLEVS	6 – 5

— C —

CALDIR	4 – 11, 13	CL	F – 3
CALIB	4 – 14, 31, 35	Clark Clean algorithm	5 – 11, 11
.	5 – 25, 26, 27, 27, 29, 29	CLBOX	G – 4
.	9 – 65, 68, 68	CLCAL	5 – 16
.	A – 2, 2	4 – 32, 32, 35, 35
.	D – 1, 1, 1	9 – 28, 28
.	E – 6	A – 2
.	L – 1, 1, 10	D – 1, 1
calibration	4 – 1, 1, 11, 14, 31, 32	E – 4, 6
.	4 – 32, 32, 32, 35, 35	L – 1, 1
.	4 – 35, 40, 40, 42, 42, 42	4 – 29, 35, 35
.	4 – 45, 46, 47, 47	9 – 2, 14, 38, 67
.	9 – 2, 2, 26, 27, 28	A – 5
.	9 – 28, 30, 31, 32, 32, 32	CLEAN	5 – 11
.	9 – 32, 36, 36, 36, 38, 38	8 – 6
.	9 – 40, 41, 41, 43, 44, 44	CLEAN beam	G – 4
.	9 – 46, 47, 47, 48, 49, 49	CLEAN box	G – 4
.	9 – 49, 49, 49, 49, 49, 56	CLEAN component	G – 4
.	9 – 56, 59, 59, 59, 61, 61		
.	9 – 62, 64, 65, 65, 67, 68		
.	9 – 68, 70, 72, 72		
.	10 – 11		
.	13 – 15		
.	A – 2, 2		
.	C – 5, 8, 9, 11, 11		
.	D – 1, 1, 1, 1		

Clean map	G - 4	CPASS	4 - 42
Clean speed-up factor	G - 4	crash	G - 5
Clean window	G - 4	crosshair	G - 5
CLIP	4 - 16 5 - 31, 31 8 - 1 10 - 7 E - 9	CSCOR	10 - 11
clipping	G - 4	CTRL C	2 - 2
closure amplitude	G - 4	CTRL D	11 - 3
closure phase	G - 4	CTRL Q	2 - 2
CLPLT	6 - 7 9 - 23	CTRL S	11 - 3
CLRMSG	3 - 3 11 - 2	CTRL Z	2 - 2
CLRNAME	3 - 4 11 - 2	cube	8 - 5, 7, 7, 9
CLRSTAT	3 - 6	CUBIT	G - 5
CLVLB	9 - 41	cursor	7 - 16
CMETHOD	5 - 11	CURVALUE	8 - 14
CNTR	6 - 8, 9, 9	cvel	2 - 11
COHER	9 - 25, 59	database	G - 5
color contour display	G - 5	data cube	5 - 31
color printer	6 - 21, 25 Z - 1	data file	6 - 20
color triangle	G - 5	data flag	8 - 1, 1
COMB	7 - 1, 2, 3 8 - 5, 14 10 - 24	DBAPP	9 - 43, 43
compact support	G - 5	DBCON	E - 17
components file	G - 5	DAT (4mm)	F - 5
COMPRESS	12 - 9	data cube	Z - 3
compressed format	F - 2, 4, 4	data file	G - 6
conjugate symmetry	G - 5	data flag	G - 5
CONPL	6 - 16	DBCON	G - 6
Conrac monitor	G - 5	DCONV	10 - 3, 24
continuum	8 - 1, 1, 5, 14	deconvolution	10 - 24
control character	2 - 2	DECSHIFT	G - 6
CONVL	7 - 2, 15 8 - 14 10 - 24	DEFAULT	9 - 19
convolution theorem	G - 5	DEFCOLOR	3 - 8
convolving function	G - 5	DEFLG	4 - 8
<i>CookBook</i>	0 - ii 1 - 3, 3, 5 A - 2	DELAY	6 - 9
coordinate reference pixel	G - 5	delay beam	4 - 16
coordinates	5 - 6 6 - 2 12 - 2 13 - 23	DELBOX	5 - 31
COPIXEL	6 - 20	delete	9 - 34
CORER	5 - 31 6 - 3	DELZN	12 - 9
correction for average spectral index	5 - 21	DFILEBOX	G - 6
correlator offset	G - 5	DFT	6 - 19
COSTAR	6 - 21	DFTIM	6 - 7
Cotton/Schwab Clean algorithm	5 - 11		7 - 17

DFTPL	6 - 7	EHEX	3 - 17
Dicomed Image Recorder (Model D47)	G - 6	12 - 3
DIFUV	10 - 7	ELFIT	7 - 17
direct Fourier transform	G - 6	emacs	2 - 10
dirty beam	G - 6	G - 7
dirty map	G - 6	ENDBATCH	12 - 5
discrete Fourier transform	G - 6	ENDEDIT	12 - 9
disk hog	G - 6	ERASE	12 - 9
disk space	3 - 10	EVASN	4 - 32
.	9 - 1	EVLA	E - 1, 1, 1, 1, 4, 4
DISKU	3 - 10	E - 5, 5, 6, 6, 9
.	11 - 4	E - 9, 11, 11, 12, 12, 12
.	F - 3	E - 12, 16, 16, 17, 17, 20
DISMOUNT	3 - 15	E - 20, 20, 23, 23
DISPLAY	12 - 14	L - 1, 1, 1, 1, 1, 1
DO3DIMAG	5 - 6, 14, 22	L - 1, 1, 1, 10, 10
DOBAND	4 - 42	Exabyte (8mm)	F - 5
DOCRT	3 - 3	Z - 3
.	4 - 32	EXIT	11 - 3
.	6 - 3	EXPLAIN	3 - 11
DOFARS	7 - 10	6 - 5
Doppler	10 - 3, 12	explain file	G - 7
DOTV	5 - 11, 16	EXPORT format	G - 7
.	6 - 5	EXPORT tape	G - 7
.	8 - 10	expressions	12 - 1, 8
DOWAIT	11 - 3	exp × sinc function	G - 7
.	12 - 4	EXTAB	12 - 17
DSKEW	6 - 2	EXTDEST	4 - 8, 19, 35
DTSIM	7 - 16	6 - 5
DTSUM	6 - 3	7 - 8
.	9 - 21	L - 1
dynamic range	G - 6	extended hexadecimal	3 - 17
		12 - 3
		extension file	G - 7
		extension files	3 - 6
		4 - 1
		12 - 2
		13 - 24
		A - 2
		F - 3
		EXTLIST	6 - 4, 5
		7 - 8

— E —

EDIT	12 - 9
EDITA	4 - 16, 20, 20, 47
.	5 - 27, 33
.	6 - 24
.	9 - 34
.	E - 6, 11
.	L - 1
editing	4 - 9, 10, 16, 16, 19, 19
.	4 - 20, 20, 23, 23, 23
.	4 - 23, 41
.	5 - 31, 33, 33, 33, 33
.	8 - 1
.	9 - 2, 24, 34, 34, 70
.	10 - 7, 7, 7, 7
.	13 - 23
.	C - 5
EDITR	4 - 16, 16
.	5 - 1, 31, 33, 33, 33
.	5 - 33
.	6 - 24
.	9 - 24, 34
.	10 - 7, 7
.	C - 5
EDT	G - 7

— F —

F2PS	12 - 17
.	Z - 4
FACES	5 - 26
FACTOR	5 - 11
false color display	G - 7
FARAD	4 - 35
FARS	7 - 10
.	8 - 14
fast Fourier transform algorithm	G - 7
FFT	5 - 31
.	G - 7
fg	Z - 5
FGCNT	5 - 31
.	E - 9

FGDIF	4 - 16	flagging	4 - 10, 16, 16, 19, 19, 20
FGPLT	4 - 16		4 - 20, 23, 23, 23, 23
	6 - 15		4 - 41
FILEBOX	5 - 16		5 - 31, 31, 33, 33, 33, 33
	6 - 19		8 - 1
FILEZAP	3 - 16		9 - 24, 34, 34, 70
	12 - 13		10 - 7, 7, 7, 7
FILIT	5 - 16		G - 8
	6 - 19	FLAGR	4 - 16
FILLM	3 - 19		5 - 31
	4 - 3, 3, 40, 46	FLAMLEVS	6 - 5
	A - 2	FLATN	5 - 14
	D - 1		7 - 3
	F - 1, 1	FLDSIZE	5 - 6, 14, 16
filtering	7 - 15	FLGIT	5 - 31
FINDR	4 - 16		8 - 1
	5 - 31, 31	flux density	7 - 4
FINISH	12 - 9	font	1 - 5
FITAB	3 - 18	FOR	12 - 8
	4 - 48	FQCENTER	9 - 49
	6 - 5	FQCUBE	12 - 21
	11 - 2	FQ number	4 - 3, 3, 8
	F - 4, 6	FQUBE	5 - 20
FITDISK	3 - 18		7 - 3
FITLD	3 - 18		8 - 7
	4 - 2, 7		E - 20
	5 - 2	FREE	3 - 10
	6 - 1	FREESPAC	11 - 4
	8 - 1		Z - 6
	9 - 2, 6, 6, 6, 6	FREQID	4 - 8
	10 - 4	FRING	9 - 2, 38, 38, 45, 46, 49
	E - 1		9 - 49, 49, 49, 58, 59, 59
FITS	1 - 1		9 - 61
	3 - 18		E - 4
	13 - 25		L - 1
FITS-disk	3 - 16, 18	fringe-fitting	9 - 36, 38, 38, 45, 46, 47
	6 - 1		9 - 47, 49, 49, 49, 49, 49
	12 - 16, 17		9 - 49, 49, 49, 56, 56, 56
FITS format	G - 7		9 - 59, 59, 61, 61, 62
FITS tape	G - 8		C - 8
fitting	7 - 6, 7, 8, 8	fringe rotator	G - 8
	8 - 14	FRMAP	6 - 7
FITTP	4 - 48	FRPLT	6 - 7
	6 - 5		9 - 26
	11 - 2	FTFLG	4 - 16
	12 - 4		5 - 31
	A - 5		E - 9
	L - 10		L - 1
	F - 4, 6	ftp	12 - 17
FIXBX	5 - 14, 16	FUDGE	12 - 19, 21
FIXWT	5 - 7	full-synthesis map	G - 8
		FXPOL	9 - 13, 13
		FXVLB	9 - 6

— G —

GAIN	5 – 11
gain file	G – 8
GAL	6 – 16
.	7 – 16
.	8 – 14
GAMMASET	6 – 25
Gaussian	7 – 7, 8, 8, 16
.	8 – 14
.	10 – 22, 22
Gaussian-tapered sinc function	G – 8
general information	13 – 25
geometric transformation	7 – 12
Gerchberg-Saxton algorithm	G – 8
GET	3 – 8
.	12 – 5, 9
GET2N	3 – 5
GET3N	3 – 5
GET4N	3 – 5
GETHEAD	8 – 9
GETJY	4 – 14, 31
.	A – 2
.	D – 1
.	E – 6
.	L – 1
GETNAME	3 – 5
GETONAME	3 – 5
.	5 – 11
GETPOPSN	12 – 9
GETTHEAD	12 – 9
GETVERS	12 – 13
Gibbs' phenomenon	G – 8
GIPSY	G – 8
global fringe fitting algorithm	G – 8
GNU	2 – 10
GO	3 – 1
.	12 – 4
Going AIPS	1 – 3
.	12 – 12, 19, 21, 21, 23
graphics functions	6 – 26, 27, 28
graphics overlay plane	G – 8
graphics plane	G – 8
gray-scale display	G – 9
gray-scale memory plane	G – 9
GRCLEAR	6 – 18
GRDROP	11 – 1
GREAD	6 – 18
Green Book	G – 9
green screen	G – 9
GREYS	6 – 8, 9, 9
.	8 – 14
.	F – 3
gridding convolution function	G – 9
gridding correction function	G – 9
GRINDEX	11 – 1
GRIPE	11 – 1
.	Z – 3
.	G – 9

gripe file	G – 9
GRLIST	11 – 1
GROFF	6 – 18
GRON	6 – 18
GWRITE	5 – 33
.	6 – 18, 25

— H —

HA2TI	5 – 3
Hanning smoothing function	G – 9
hard copy	13 – 28
hardware mount	G – 9
header	8 – 9
header record	G – 9
HELP	3 – 1, 2, 11
.	6 – 5
HELP ADVERBS	3 – 2, 11
HELP ARRAYS	3 – 11
help file	G – 9
HELP NEWTASK	12 – 12
HELP PROCS	3 – 11
.	12 – 9
HELP PSEUDOS	3 – 11
HELP REALS	3 – 11
HELP STRINGS	3 – 11
HELP VERBS	3 – 2, 11
Hermitian function	G – 9
HGEOM	7 – 3, 12
high-frequency VLA data	D – 1, 1, 1, 1, 1
HINOTE	3 – 8
HISEQ	7 – 15
histogram	7 – 4
.	8 – 6
history file	3 – 8
.	F – 3
.	G – 9
HITEXT	3 – 8
Hogbom Clean algorithm	G – 9
host computer	G – 9
HSA	C – 1
hue	G – 10
HUINT	6 – 16, 21
.	8 – 14
HYB	9 – 68
hybrid mapping algorithm	G – 10

— I —

I ² S	G – 11
IBLED	5 – 31
.	10 – 7
IIS	G – 10
IM2HEAD	3 – 6
IM2TV	6 – 18
IM2UV	10 – 24
IM3HEAD	3 – 6
IM4HEAD	3 – 6

image	13–28, 29	instrumental polarization	G–10
	G–10	intensity	G–10
image catalog	G–10	interactive	13–33
image file	G–10	Internet	3–15, 18
image plane	G–10	INTYPE	3–5
image reconstruction	G–10	invisible distribution	G–11
imaging	13–29	IPL	G–11
IMAGR	4–49, 50	IRING	6–16
	5–1, 3, 4, 5, 6, 7	isoplanaticity assumption	G–11
	5–7, 10, 10, 11, 11, 11	ISPEC	6–4, 12
	5–14, 14, 16, 16, 21, 22		7–8
	5–22, 26		8–14
	6–16, 19, 24		
	8–1, 6		
	9–68, 68, 70		
	L–1, 10		
IMCENTER	7–4, 6	JMFIT	6–4, 19
IMDIST	7–4		7–7
IMEAN	6–14	jobs	Z–5
	7–4		
	8–6		
IMFIT	6–4, 19	KLEENEX	11–1, 3
	7–7	KNTR	6–8, 9
IMFRING	E–20		8–10
IMHEAD	3–6		10–23
	4–3		F–3
	8–1		
	9–21		
	E–1		
IMLIN	8–5, 14	l ₁ solution algorithm	G–11
	10–20	l ₂ solution algorithm	G–11
IMLOD	6–1	LASTEXIT	3–8
IMOHEAD	3–6	LAYER	6–16
import	Z–2	LGEOM	7–12, 12
IMPOS	5–31	line editor	G–11
	6–20	LINIMAGE	E–17
IMSCAL	E–20	LIST	12–9
IMSIZE	5–6, 14	LISTR	4–8, 10, 13, 14, 16, 19
IMSTAT	7–4		4–19, 32, 35, 35, 35, 47
IMTXT	6–4		5–11, 27
IMVIM	6–14		6–3, 4
IMXY ; IMVAL	6–20		9–21, 49
INCLASS	3–5		A–2, 2, 5
	12–2		E–1
INDXR	4–2		L–1
	9–6, 12, 21	lobe rotator	G–11
	10–1, 11	LOCIT	4–29
	E–1		6–16
information	13–32	log in	2–1
INNAME	3–5	loop gain	G–11
	12–2	LPCAL	9–61
INP	3–2	lpq	Z–4
INPFIT	7–7	lprm	11–5
INPUTS	3–8	LTYPE	Z–4
inputs file	G–10		5–11
INSEQ	3–5		6–8
	12–2	luminance	G–11

LWPLA	0 – ii	MODSP	7 – 16
.	4 – 42, 45	MODVF	8 – 14
.	6 – 5, 5	MOMNT	8 – 14, 14
.	8 – 14	monitor	G – 11
.	10 – 5, 23	MORIF	E – 6, 17
Motif		Motif	2 – 11
.		MOUNT	3 – 14, 15, 15
.	9 – 6
M2CAT	3 – 4	11 – 5
M3CAT	3 – 4	12 – 16
M3TAR	9 – 17	mouse	2 – 3, 11, 11
M4CAT	3 – 4	MOVE	3 – 11
magnetic tape	3 – 14, 15, 15	MSGSRV	2 – 3
.	11 – 5, 5	3 – 3
.	13 – 51	MSORT	9 – 11, 19
.	F – 4, 4	<i>m</i> × <i>n</i> map	G – 11
.	Z – 2	MULTI	4 – 2
major cycle	G – 11	multi-band delay	9 – 48, 49
man aips	2 – 3	multi-source data sets	4 – 1
man readline	2 – 10	MWFILT	7 – 15
map	G – 11	MX	6 – 19, 24
MAPPR	5 – 5	G – 12
MATCH	9 – 6, 13	N	— N —
MATHS	7 – 13	NAMEGET	12 – 2, 13
MAXFIT	7 – 6	natural weighting	5 – 7
Maximum Entropy Method	5 – 24	NBOXES	G – 12
.	G – 11	negative bowl artifact	5 – 16
MAXPIXEL	5 – 11	NFS	G – 12
MBDLY	9 – 49	NINER	3 – 10
MCAT	3 – 4	NITER	7 – 12, 15
.	5 – 5	NOIFS	5 – 11, 11
MCUBE	8 – 7, 7	non-closing offset	E – 6
.	10 – 15	Nyquist sampling rate	G – 12
memory page	G – 11	O	— O —
memory paging	G – 11	OBEDT	13 – 37
memory thrashing	G – 11	OBIT	9 – 46
message file	3 – 3	OBITIMAG	E – 1, 1, 11
.	G – 11	OBITMAP	5 – 38
message monitor	3 – 3	OBITPEEL	5 – 38
message terminal	G – 11	OBITSCAL	5 – 38
MF2ST	6 – 8	Object-based tasks	5 – 16
.	7 – 8	OBXFILE	9 – 16
MFITSET	6 – 19	OBPLT	13 – 36
MFPRT	7 – 8	obsolete	9 – 16
.	10 – 22	OBTAB	3 – 4
microcode	G – 11	OCAT	6 – 19
minimum-match	1 – 5	OFFPSEUD	6 – 19
.	12 – 1	OFFTRAN	6 – 19
minor cycle	G – 11	OFFZOOM	6 – 18
MINPATCH	5 – 11	OFLAG	4 – 3
MK3IN	9 – 2, 17, 17	OFMCONT	6 – 19
MK3TX	9 – 17	OFMDIR	6 – 19
MkIII	9 – 6, 17, 17, 20, 21	OFMGET	6 – 19
MODAB	7 – 16	OFMLIST	6 – 4
model	13 – 35		
modeling	7 – 6, 7, 8, 8, 16		
.	10 – 13, 22, 22		
MODIFY	12 – 9, 9		
MODIM	7 – 16		

OFMSAVE	6 – 19	PCFIT	9 – 36
OFMTWEAK	6 – 19	PCFLG	9 – 36
OGEOM	7 – 12	PCLOD	9 – 36, 36
	8 – 14	PCNTR	6 – 8, 9, 9
	10 – 15, 15	PCPLT	9 – 36
OHGEO	5 – 22	PCVEL	E – 17
	7 – 3, 12	PEELR	5 – 31
	8 – 14	PGEOM	7 – 12
ONEBEAM	5 – 6, 14	phaseless reconstruction	G – 12
one-dimensional	13 – 36	phase referencing	9 – 30, 47, 49
on-the-fly	10 – 2, 3, 4, 7, 12, 15	phase tracking center	G – 12
	10 – 15, 15	physical memory	G – 12
OOCAL	5 – 26	pillbox	G – 12
	E – 20	pixel	G – 12
OOFRING	E – 20	pixel coordinates	G – 12
OOP	13 – 37	PLAYR	6 – 21
OOSUB	5 – 22, 31	PLCOLORS	6 – 5, 9
	E – 17	PLCUB	6 – 12
opacity	7 – 2	PLGET	8 – 10, 14
operating system	G – 12	plot file	3 – 8
optical	6 – 14		6 – 5
	13 – 38	plot file	6 – 5, 7, 7, 9, 9, 12
OTF	10 – 2, 3, 7, 12, 15, 15		6 – 14, 16
OTFB5	10 – 2, 4		10 – 5, 23
OTFIN	10 – 2		F – 3, 4
OTFUV	10 – 2, 3, 3		G – 12
OUTCLASS	3 – 5	PLOTR	6 – 16
	12 – 2	plots	13 – 39
OUTFILE	Z – 2	PLROW	6 – 12
OUTNAME	3 – 5		8 – 14
	12 – 2	point source response	G – 12
OUTPRINT	6 – 3	points per beam	G – 12
	12 – 5, 17	point spread function	G – 12
	Z – 4	polarization	4 – 32, 35, 35, 35, 35
OUTPUTS	3 – 8		5 – 24
	4 – 35		6 – 9
OUTSEQ	3 – 5		7 – 1, 1
	12 – 2		9 – 1, 2, 13, 13, 17, 35
OVRSWTCH	5 – 14		9 – 47, 61, 61, 62, 62
			13 – 42
			A – 5
			E – 12, 12, 12

— P —

page	G – 12	<i>POPS</i>	3 – 1, 2
page swapping	G – 12		12 – 1, 1, 6, 8, 9, 9
Paley–Wiener theorem	G – 12		12 – 9, 9, 12, 12
paraform tasks	12 – 19, 19, 20, 21, 21		13 – 42
	13 – 38		G – 12
parallactic angle	4 – 35	<i>POPS</i> procedure	G – 13
PATGN	7 – 13	<i>POPS</i> symbols	G – 13
	10 – 24	POPSYM	3 – 1
PBCOR	7 – 13	portability	1 – 1
PCAL	4 – 35, 35	POSSM	4 – 42, 42, 45
	9 – 62, 62		6 – 7, 15
	A – 5		8 – 14
	E – 12, 12, 12		9 – 22, 27, 40, 40, 49, 49
PCASS	9 – 36		10 – 5
PCCOR	9 – 36, 36		E – 6, 12, 17, 20
PCEDT	9 – 36		L – 1

PostScript	0 – ii	pseudoverb	12 – 1
.	6 – 5, 5, 21, 25	13 – 47
.	8 – 10, 14	G – 13
.	10 – 5, 23	PSF	G – 13
.	12 – 17	PUTHEAD	8 – 9
.	Z – 2	9 – 13
primary beam correction	G – 13	10 – 1
primary-beam corrections	5 – 21	— Q —	
primary data file	G – 13	Q2HEADER	3 – 6
principal solution	G – 13	Q3HEADER	3 – 6
PRINTER	3 – 3	Q4HEADER	3 – 6
.	6 – 3, 4, 5	QCCREATE	12 – 4
.	11 – 5	QGETVERS	12 – 13
.	Z – 1	QHEADER	3 – 6
print job	Z – 4	QINP	3 – 8
PROC	12 – 9	QHEADER	3 – 6
procedure	12 – 9, 9, 9, 12, 12	Q-routine	G – 13
.	13 – 45	QUACK	4 – 10, 20
.	G – 13	9 – 34
PROFL	6 – 12	QUEUES	12 – 5
prolate spheroidal wave function	G – 13	QUFIX	E – 12
prompt character	G – 13	quick boot	G – 14
PRTAB	4 – 1, 19, 35	QUOUT	E – 12
.	6 – 4	QUXTR	7 – 10
.	9 – 21	— R —	
PRTAC	6 – 5	RAINLEVS	6 – 5
PRTAN	4 – 8, 35	RANCID	G – 14
.	6 – 4	RASHIFT	5 – 6, 6, 14
.	9 – 19, 21	READISK	3 – 18
.	A – 2	12 – 18
.	E – 1	readline	2 – 10
.	L – 1	re-boot	G – 14
PRTCC	5 – 22, 26	REBOX	5 – 16
.	6 – 4	REBYTE	6 – 19
PRTHI	3 – 8	RECAT	3 – 18
.	6 – 5	12 – 18
PRTIM	5 – 11	recipe	3 – 6
.	6 – 4	2 – 12, 12
PRTMSG	3 – 3	3 – 20
.	5 – 2	6 – 28, 28
.	6 – 4	7 – 17, 17, 18, 18, 18
PRTOF	4 – 3	8 – 17, 17, 18, 18, 18
PRTSD	10 – 1, 5	9 – 73, 73, 74, 74, 74
PRTSY	E – 11	11 – 6
PRTTP	3 – 15	12 – 23, 24, 24, 24
.	4 – 48	13 – 81, 81, 82, 82, 82
.	5 – 1	A – 6, 6, 6
.	6 – 5	B – 8, 8
.	A – 2	C – 13, 13, 14, 14, 14
PRTUV	5 – 11	D – 7, 8, 8, 8, 8
.	6 – 3	E – 22, 22, 23, 23
Prussian helmet Clean algorithm	G – 13	L – 12, 12, 12
pseudo-AP	G – 13	F – 6, 6
pseudo array processor	3 – 20	REFANT	5 – 29
pseudo-array processor	G – 13		
pseudo-color display	G – 13		
pseudo-continuum u-v data file	G – 13		
PSEUDOVB	12 – 19		

REFLG	4 – 16	ROTATE	5 – 5, 6, 14
	E – 9	rotation measure	7 – 2
regularization method	G – 14	rotation-measure synthesis	7 – 10
regularization parameter	G – 14	RSPEC	6 – 12
REHEX	3 – 17	RSTOR	5 – 11, 24
	12 – 3	RUN	3 – 17
REIFS	E – 6, 17		12 – 3, 3
re-IPL	G – 14		13 – 47
REMAG	7 – 3, 5	run file	G – 14
	10 – 24		
REMDISK	12 – 14		
REMHOST	3 – 15		
	12 – 16		
REMOTE tapes	3 – 15	SABOX	5 – 16
	12 – 16	SAD	6 – 4, 8, 9, 24
remote user	12 – 14, 14, 15, 17		7 – 8
REMOVIE	8 – 10		10 – 22
REMTAPE	3 – 15	sampling theorem	G – 14
	12 – 16	saturation	G – 14
RENAME	3 – 6	SAVE	3 – 8
	10 – 21	SBCOR	12 – 5, 9
RENUMBER	3 – 6	SCIMG	9 – 20
REROAM	6 – 17	SCLIM	5 – 26
residual delay	G – 14	SCMAP	6 – 24
residual fringe rate	G – 14		9 – 68, 70
resolution	G – 14	SCLIM	6 – 16
RESTORE	3 – 8	SCMAP	5 – 1, 26
restoring beam	G – 14		6 – 24
RETURN	12 – 9		9 – 68, 70
REWAY	5 – 7	scratch	G – 14
	E – 16, 16	scratch file	G – 14
	L – 1	SCRDEST	3 – 10
REWIND	3 – 15		11 – 4
RFARS	7 – 10	screen editor	G – 15
RFI	6 – 3	scroll	G – 15
RFLAG	4 – 16	SDCAL	10 – 11
	5 – 31	SDCLN	5 – 24
	E – 5, 9, 9, 9	SDGRD	6 – 24
	L – 1, 1, 1	SDIMG	10 – 13
RGBMP	6 – 16, 21	SDLSF	10 – 12
	8 – 14	SDMOD	10 – 13
RLCAL	E – 12	SDTUV	10 – 4
RLCOR	4 – 35	SDVEL	10 – 1, 3, 12
RLDIF	4 – 35, 35, 35	self-calibration	5 – 25, 26, 27, 27, 29, 29
	A – 5		8 – 4
	E – 12, 12		9 – 68, 68, 68, 70
RLDLY	9 – 61, 61	self-calibration algorithm	G – 15
	E – 12	self-calibration gain solution algorithm	G – 15
RM	7 – 10	SERCH	8 – 14
RM2PL	6 – 24	SETFC	5 – 5, 6, 14
	7 – 10		L – 10
RMFIT	6 – 24	SETJY	4 – 8, 8, 11, 40
	8 – 14		9 – 43
RMSD	7 – 3, 5		A – 2
	8 – 14		D – 1
roam	G – 14		L – 1
ROAMOFF	6 – 17	SETMAXAP	3 – 20
ROBUST	5 – 7	SETRGBL	6 – 5, 28

SETSLICE	6 – 19	SOLTYP	5 – 27
	7 – 8	sort order	G – 16
SETEXWIN	6 – 19	source editor	G – 16
SG2RUN	3 – 8, 17	SOUSP	4 – 31
	12 – 3, 18		E – 6
SHADW	7 – 15	spatial resolution	G – 16
Shannon sampling theorem	G – 15	spawn	G – 16
Shannon series	G – 16	SPCAL	9 – 61
shed	G – 16	SPCOR	7 – 13
SHOUV	6 – 3	SPECR	8 – 1
	9 – 22	spectral index	4 – 11, 31, 42
SHOW	5 – 11		7 – 2
single-dish	10 – 1, 2, 3, 4, 5, 5		E – 6, 6, 12, 17, 20
	10 – 7, 7, 11, 11, 12, 13	spectral-line	4 – 40, 41, 42, 45
	10 – 13, 15, 15, 15, 15, 19		8 – 1, 1, 1, 4, 5, 6
	10 – 20, 22, 22, 24		8 – 7, 7, 9, 10, 14, 14
	13 – 48		8 – 14
SIVPSF	G – 16		9 – 2, 43, 44, 44, 47, 59
SIZEFILE	3 – 20		9 – 59, 59
SKYVE	7 – 12		10 – 3, 3, 7, 7, 12, 20
SL2PL	6 – 12, 15		10 – 20, 22
	7 – 8, 8		13 – 49
	8 – 14	SPFLG	4 – 16, 16, 41
SLFIT	7 – 8		5 – 31
SLICE	7 – 8		6 – 24
	8 – 14		8 – 1
	G – 16		10 – 7, 7
slice file	G – 16		L – 1, 1
SLPRT	7 – 8	spheroidal function	G – 16
SMODEL	5 – 27	SPIXR	5 – 20
snapshot	G – 16		7 – 3
SNCOP	9 – 59		E – 20
SNEDT	4 – 7	SPLAT	L – 1
	5 – 27	SPLIT	4 – 35, 49
	6 – 24		8 – 1, 1, 4
	9 – 28, 41, 41, 49		9 – 44, 65
	E – 11		A – 5
SNFLG	4 – 16		C – 9
	5 – 31		E – 16
	9 – 34		L – 1
	D – 1		F – 6
	E – 6	SPMOD	7 – 16
SNIFS	6 – 15	SPY	3 – 10
SNP2D	E – 6, 16	SQASH	7 – 10
SNPLT	4 – 13, 14, 16, 32, 47		8 – 5, 14
	5 – 26, 27	STACK	7 – 3
	6 – 15	STALIN	3 – 8
	9 – 24, 36, 49	STARS	6 – 8
	E – 11	statistics	7 – 4
	L – 1, 1	status	3 – 6
SNREF	4 – 35	STEPLEVS	6 – 5, 28
SNSMO	9 – 28, 28, 44, 49	Stokes' parameters	G – 16
software mount	G – 16	Stokes' visibility functions	G – 17
Solar data	4 – 46, 47, 47	STORE	3 – 8
SOLCL	4 – 46, 47	STQUEUE	5 – 11
SOLINT	5 – 29	structure of AIPS	1 – 5
	9 – 49	SUBIM	3 – 11
SOLMODE	5 – 27		8 – 7

subimage	G - 17	TASAV	4 - 35, 46
SUBMIT	12 - 5	9 - 27
sub-task	G - 17	A - 2, 2, 5
SUFIX	E - 1	C - 5
sum	7 - 2	F - 6
SUMIM	7 - 3	task	3 - 1
super-resolution	G - 17	12 - 18, 19, 20, 21, 21, 21
support	G - 17	G - 18
support width	G - 17	tasks	13 - 52
SVLBI	9 - 2, 16, 46, 58	TASRT	4 - 1
SVPSF	G - 17	TBDIF	6 - 4
<i>Synthesis Imaging in Radio Astronomy</i>	4 - 3, 46	TBIN	12 - 17
	5 - 4, 24, 25, 29, 29	t-b order	G - 18
	7 - 12	TBOUT	12 - 17
	8 - 1	TDEPEND	5 - 16
	9 - 49	TECOR	9 - 2, 14, 14
	G - 17	C - 4
synthesized beam	G - 17	TEK4012	G - 18
SYSOL	E - 11	TEK screen	G - 18
SysPower	E - 11	TEKSRV	2 - 3
		6 - 5, 26

— T —

TAB	2 - 10	television	1 - 1
TABED	4 - 1, 19, 41	Telex 6250 tape drive	G - 18
TARGET	12 - 9	TELL	5 - 11
tables	4 - 1	12 - 5
	9 - 6, 24, 72, 72	terminal page	G - 18
	13 - 50	terminal scroll	G - 18
TAPPUT	9 - 11	TeX	0 - ii
TACOP	4 - 1, 41, 45	text editor	G - 18
	6 - 5	text file	G - 18
	8 - 4	text files	3 - 16, 16
	L - 1	TFILE	9 - 17
TAFFY	12 - 19, 21	TGET	3 - 8
TAFLG	4 - 1	12 - 4, 5
	5 - 22	TGINDEX	3 - 8
TAMRG	4 - 1	Third NRAO Synthesis Imaging Summer School	G - 18
	9 - 20	thrashing	G - 18
tape	3 - 14, 15, 15	TI2HA	5 - 3
	11 - 5	time-baseline order	G - 18
	13 - 51	time smearing	G - 18
	F - 4, 4	TKASLICE	6 - 27
	Z - 2	TKERASE	6 - 26
tape blocking efficiency	G - 18	TKGUESS	6 - 27
tape mount	Z - 5	TKMODEL	6 - 27
taper	G - 18	TKPL	6 - 5, 27
TAPES	3 - 14	7 - 4
	12 - 16	TKPOS	6 - 27
TAPLT	5 - 22, 26	TKRESID	6 - 27
	6 - 15	TKSET	6 - 27
	9 - 16, 17	TKSLICE	6 - 27
TAPPE	4 - 1	8 - 14
	5 - 31	TKVAL	6 - 27
	8 - 4	TKWIN	6 - 27
TARS	7 - 10	TKXY ; IMVAL	6 - 27
		TPHEAD	3 - 15
		6 - 1
		8 - 1

TPMON	11–5	TVLABEL	6–8, 21
	12–16	TVLINE	6–21
	Z–5	TVLOD	6–17, 21
TPUT	3–8		7–8
	12–4, 5	TV look-up table	G–19
trackball	G–18	TVMAXFIT	6–20
trackball button	G–18	TVMODEL	6–24
TRANS	7–3	TVMOVIE	6–21
	8–5, 9, 10		8–10
	10–20	TVNAME	6–19
transfer function	G–18	TVOFF	6–18
TRC	G–18	TVON	6–18
true color display	G–18	TVPHLAME	6–17, 19
TU77 tape drive	G–18	TVPL	6–5
TU78 tape drive	G–19	TVPOS	6–18
TV3COLOR	6–21	TVPSEUDO	6–17, 19
TVALL	6–17, 17, 19	TVRESID	6–24
	7–7, 8	TVRGB	6–21
	8–10		8–14
TVANOT	6–21	TVROAM	6–17
TVASLICE	6–24	TV roam	G–19
TVBLINK	6–21	TVSAD	6–24
TV blink	G–19		7–8
TVBOX	5–16	TV scroll	G–19
	6–19	TVSET	6–24
TCLEAR	6–18		7–8
TCOLORS	6–9	TVSLICE	6–24
TCOMPS	6–24		7–8
TCPS	0–ii		8–14
	6–5, 25	TVSPC	6–24
	8–10		7–8, 10
	10–7		8–14
	Z–2	TVSTAR	6–21
TVCUBE	6–21	TVSTAT	6–20
	8–10		7–4
TV cursor	2–11	TVTRAN	6–19, 21
	G–19	TVWEDGE	6–17
TVDIST	6–20	TVWIN	6–19
TVFLG	4–16, 23, 23, 23, 23, 23		7–4
	4–46	TVWINDOW	7–7
	5–11, 31	TVWLABEL	6–17, 21
	6–24	TVZOOM	6–18
	A–2	TV zoom	G–19
	E–5, 16	twm	2–11
TVFLUX	6–20	TXPL	6–5, 5
TV functions	6–5, 16, 17, 19, 21, 21		12–15
	6–21, 25	TYAPL	4–7
	13–62, 64		5–7
TVGUESS	6–24		E–11, 16
TVHELIX	6–17, 19		L–1
TVHLD	6–17	TYCOP	9–41
TVHUEINT	6–21		E–11
	8–14	TYSMO	4–7
TVHUI	6–16, 21		9–41
	8–14		E–11
TVILINE	6–21		
TV image catalog	G–19		
TVINIT	6–18		

— U —

U2CAT	3 – 4	UVHIM	6 – 7
U3CAT	3 – 4	UVIMG	6 – 7
U4CAT	3 – 4	UVLIN	8 – 1, 5
UBAVG	5 – 3	UVLOD	4 – 2, 7
UCAT	3 – 4	UVLSF	5 – 2
.	5 – 2	E – 1
UFLAG	5 – 31	UVLSF	8 – 1, 1, 5
.	6 – 24	E – 17
.	E – 20	UVMAP	6 – 24
undefined pixel	7 – 3, 5	8 – 6
.	8 – 14	u-v measurement distribution	G – 19
uniform weighting	5 – 7	UVMLN	5 – 31
.	G – 19	8 – 1
UNIX	G – 19	UVMOD	7 – 16
UNQUE	12 – 5	UVPLT	4 – 14, 46
UOCAT	3 – 4	5 – 10, 11, 26, 31, 31
URL	1 – 1	6 – 7
.	3 – 11	9 – 67
user-coded task	G – 19	10 – 5
USUBA	9 – 6, 11	UVPRT	A – 2
UTESS	5 – 24	UVRANGE	6 – 3
utilities	13 – 66	u-v sampling distribution	4 – 14
UV	13 – 67	UVSIZE	5 – 7
UV2MS	4 – 2	UVSRT	5 – 3
UVAVG	9 – 20, 65, 65	9 – 6, 19
.	E – 20	10 – 7
UVBAS	8 – 5	UVSUB	5 – 22, 31
UVBOX	5 – 7	UVTAPER	5 – 7
UVBXFN	5 – 7	u-v taper function	G – 20
UVCON	7 – 16	u-v transfer function	G – 20
UVCOP	3 – 11	u-v weight function	G – 20
.	8 – 1	UVWTFN	5 – 7, 7
.	9 – 34
.	A – 2
.	E – 17
u-v coverage	G – 19	Varian printer	G – 20
UVCRS	9 – 67	Variational Method	G – 20
u-v data file	G – 19	VBCAL	9 – 67
u-v data flag	G – 19	VBGLU	9 – 13
UVDIF	5 – 26	E – 1
.	6 – 3	VBMRG	9 – 13
UVFIL	12 – 19, 21	velocity	9 – 43
UVFIT	7 – 16	VERB	12 – 19
.	9 – 67	13 – 72
u-v FITS format	G – 19	G – 20
UVFLG	4 – 10, 16, 19	verbs	3 – 2
.	5 – 31	Versatec printer	G – 20
.	8 – 1	VERSION	3 – 8
.	10 – 7	12 – 4
.	A – 2	Very Long Baseline Interferometry —	–
.	L – 1	Techniques and Applications	G – 20
UVFND	5 – 11, 31, 31	VGET	3 – 8
.	6 – 3	VGINDEX	3 – 8
UVFRE	8 – 1	vi	G – 20
UVHGM	6 – 7	virtual memory page	G – 20
.	virtual memory page swapping	G – 20
.	virtual memory storage	G – 20

— V —

visibility phase tracking center	G - 20	VLBI 9 - 1, 1, 2, 2, 6, 6
VLA	9 - 33		9 - 6, 6, 10, 11, 11, 13
	13 - 78		9 - 13, 14, 14, 16, 17, 17
	C - 5, 11, 11		9 - 19, 20, 20, 21, 22, 22
VLACALIB	4 - 13, 14, 32		9 - 23, 24, 25, 26, 27, 28
	A - 2		9 - 28, 30, 31, 32, 33, 34
VLACLAL	A - 2		9 - 34, 35, 36, 38, 38, 41
VLAMP	9 - 33		9 - 43, 44, 44, 45, 47, 47
VLANT	4 - 29		9 - 48, 49, 49, 49, 49, 49
	D - 1		9 - 56, 56, 56, 59, 59, 61
	E - 4		9 - 62, 62, 64, 65, 65, 67
VLAPROCS	4 - 1		9 - 68, 70, 72
	A - 2		13 - 79
VLARESET	4 - 32		C - 1, 4, 4, 5, 5, 5
VLARUN	A - 1, 1		C - 8, 8, 8, 9, 11, 11
VLBA	9 - 1, 2, 2, 6, 6, 6		C - 11, 12, 13
	9 - 6, 10, 11, 13, 13, 16	VLOG	9 - 31
	9 - 20, 31, 32, 36, 43	VM	G - 20
	C - 1, 5, 8, 8, 11, 12	VMS	G - 20
	C - 13	VPFLG	4 - 16
VLBAAMP	9 - 41	VPLOT	5 - 22
	C - 2, 5		6 - 7
VLBABPASS	C - 5		9 - 23, 27, 34, 49, 49, 49
VLBABPSS	C - 2		9 - 65
VLBACALA	C - 10, 11		D - 1
VLBACCOR	9 - 36	VPUT	3 - 8
	C - 2	VTESS	5 - 24, 24
VLBACPOL	C - 2		6 - 24
VLBACRPL	9 - 22	VTEST	10 - 12
	C - 2, 5		
VLBAEOPS	9 - 16		
	C - 2		
VLBAFIX	9 - 6, 10, 11, 11, 26	WAIT	11 - 2
	C - 2	wc	Z - 4
VLBAFPOL	9 - 13	wedge	G - 20
VLBAFRGP	C - 2, 8	WETHR	4 - 16
VLBAFRNG	C - 2, 8		6 - 15
VLBALOAD	9 - 6	wild-card	12 - 2
	C - 2, 4	window Clean	G - 21
VLBAMPCL	9 - 38, 38	window management	2 - 11
	C - 2, 5	window manager	2 - 1
VLBAPANG	9 - 35	WIPER	4 - 16, 16
	C - 2, 8		5 - 31, 31
VLBAPCOR	9 - 36		6 - 24
	C - 2, 5	workfile	5 - 11, 22, 31
VLBARUN	9 - 5, 5, 5	working set size	G - 21
	C - 3, 3, 3	workstation	2 - 1
VLBASNPL	C - 2, 5, 8	World-Wide Web	0 - ii
VLBASUMM	9 - 21		1 - 1
	C - 2, 4		3 - 11
VLBATECR	9 - 14, 14	WRDISK	3 - 18
	C - 4		12 - 18
VLBATRCR	C - 2	WRTPROCS	12 - 18
VLBAUTIL	9 - 5, 5, 6, 10, 11, 11	WTSUM	10 - 13, 21
	9 - 13, 14, 16, 20, 21, 22		
	9 - 24, 26, 35, 36, 36, 38		
	9 - 41, 49		
	C - 3, 3, 4, 5, 8, 8		

— X —

XAS	2–3, 11
.	6–5, 16, 21
XBASL	6–24, 28
.	8–5, 14
.Xdefaults	2–2, 3
.	3–3
XG2PL	6–24
.	7–10
XGAUS	6–24
.	7–10
.	8–14, 14
XHELP	3–11
XMOM	8–14, 14
.	10–22
XPLOT	6–24, 28
.	7–10
.	8–10, 14
XSMTH	8–14
xv	Z–2
x-y order	G–21

— Y —

Y-routine	G–21
---------------------	------

— Z —

ZAMAN	6–24
.	7–10
.	8–14
ZAP	3–6
.	5–11
.	11–2, 4
ZEMAN	6–24
.	7–10
.	8–14
ZEROSP	5–10
zero-spacing flux	G–21
zoom	G–21
Z-routine	G–21