5 Making Images from Interferometer Data

This chapter is devoted to the use of $\mathcal{AIPS}$ to make and improve images from interferometer visibility data. It begins with a brief description of the routes by which such data arrive in $\mathcal{AIPS}$. 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 $\mathcal{AIPS}$ is described. This entire process often isolates bad data samples, not previously removed from the data set. An interactive, baseline-based data editor called `EDITR` is described at the end of the chapter. You may find it more useful than `TVFLG` (§ O.1.6) for removing data at this stage in the processing. Task `IMAGR` now does "3-dimensional" imaging, `SCMAP` contains an editing option at each self-calibration cycle, and `EDITR` has replaced `IBLED` as the baseline-based editor of choice. A UV-plane based interactive editor called `UFLAG` is now available.

Lists of $\mathcal{AIPS}$ 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 § 4.3.5 and § 8.1, and imaging and self-calibration are also discussed in § 8.4 for spectral-line data and in § 9.7 for VLBI data.

5.1 Preparing *uv* data for imaging

$\mathcal{AIPS}$ 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 $\mathcal{AIPS}$ cataloged files, then you will need to import them. These data will normally arrive in $\mathcal{AIPS}$ 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 $\mathcal{AIPS}$.

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 C_R	to list all files on all disks.
> 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 (76) in the first (CAT) column of the listing. This number and the disk number (3) should be noted for future reference as they are useful when selecting this data set for further processing. Don't forget that the 2nd, 3rd, 4th, and 5th input and the output data name sets also have corresponding MCAT, UCAT, and CATALOG verbs. See § 3.3 and § 3.3.1.

5.1.3 Sorting the data — UVSRT

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

> TASK 'UVSRT' ; INP CR	to set the task name and list the input parameters.
> INDI *n* ; GETN *ctn* CR	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' CR	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' CR	to select the "XY" sort type required for image making.
> INP CR	to review the inputs you have selected. *N.B.,* check them carefully since the sort can be time consuming for large data sets.
> GO CR	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.

5.1.4 Combining data sets for imaging — DBCON and VBGLU

Multiple data sets may be combined — with some serious restrictions — to make a single image. DBCON only works properly if the two data sets are rather similar. Multi-source files must have been observed with the same antenna locations and names; DBCON then produces a multi-source output file with one subarray. For single-source files (after SPLIT) however, DBCON can create an output with multiple "subarrays" represented by multiple AN (antenna) tables. Thus data from multiple array configurations having the same polarization, spectral, and IF structure can be combined and imaged together. DBAPP does a similar operation for data sets that match more closely. DBCON allows you to scale the weights of the two data sets to control how they contribute to the final image. Task AVGWT will provide data on the average and rms of weights in selected portions of a data set.

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

5.2 Basic image making — IMAGR

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

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

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

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

5.2.1 Making a simple image

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

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

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

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

> DOCRT = -1 ; EXPLAIN IMAGR C_R	to print the long explain file, and
> HELP *xxx* C_R	

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

> INP C_R

then:

> MAPPR C_R	to run IMAGR.

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

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

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

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

> INDI 0 ; MCAT 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. In 31DEC25, adverb DOINVERS may be used to obtain an alternate set of field locations. Tasks CHKFC and FLATN may then be used to choose whichever is better for your imaging.

You specify the multiple-field information with:

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

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

> 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 DECSHIFT values. To simplify the coordinate computations, the shift parameters may also be given as right ascension and declination of the field center, leaving IMAGR to compute the correct shifts, including any rotation. BOXFILE may also be used to specify initial Clean boxes for some or all fields, values for BCOMP, spectral-channel-dependent weights, and even a list of facets to ignore totally.

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

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

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

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

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

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

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

5.2.3 Data weighting

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

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

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

> UVSIZE 0 ; UVWTFN ' ' 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

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

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

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

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

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

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

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

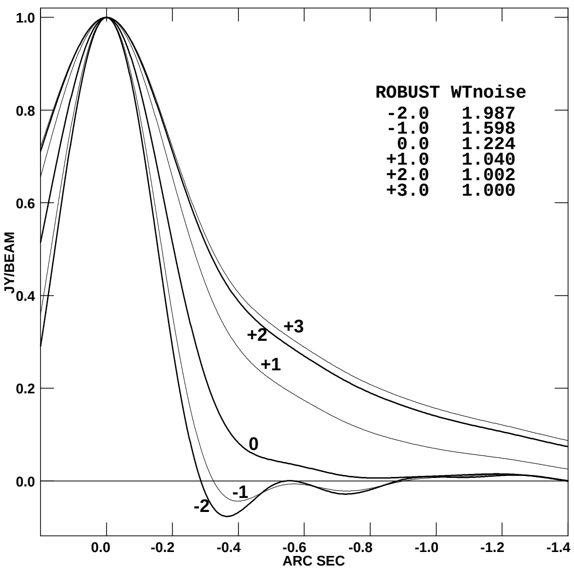
The option to average the input data in time over a baseline-length dependent interval adds further complications. It has the desirable effect of reducing the size of your work file, thereby speeding the imaging process. But it changes the distribution of data samples ahead of uniform weighting since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

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

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

5.2.4 Cell and image size, shifting

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



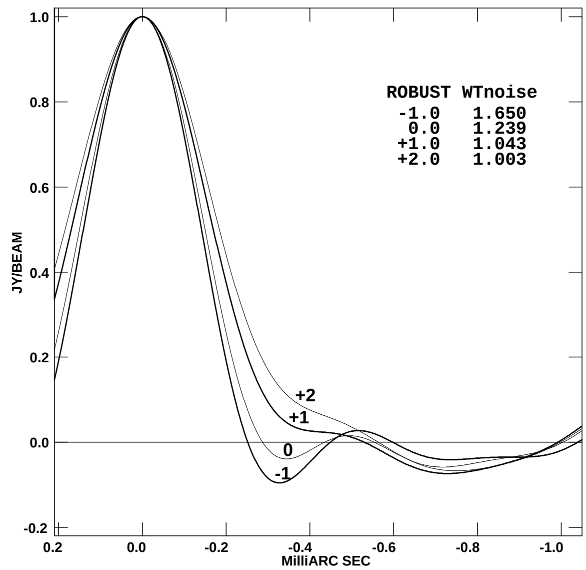


Figure 5.1: Slices taken through the centers of synthesized beams for various values of the ROBUST parameter. Plot at left for a VLA A- and B-array data set, while the plot at right is for a VLBA data set. Do not assume that these plots apply to your data sets, however. Tables give noise increase over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

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

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

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

5.2.5 Zero-spacing issues

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

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

5.3 Deconvolving images

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

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

5.3.1 Basic Cleaning with IMAGR

IMAGR implements a Clean deconvolution of the type devised by Barry Clark and enhanced by Bill Cotton and Fred Schwab. Clean components — point sources at the centers of cells — are found during "minor" iteration cycles by Cleaning the brightest parts of the residual image with a "beam patch" of limited size. More precise Cleaning is achieved at the ends of "major" iteration cycles when the Fourier transform of the Clean components is computed, subtracted from the visibility data, and a new residual dirty image computed. The rule for deciding when a major iteration should end in order to achieve a desired accuracy is complicated (see the Clark paper). IMAGR lets you vary the major iteration rule somewhat to suit the requirements of your image. Type DOCRT FALSE; EXPLAIN IMAGR C$_R$, 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 $\neq 0$, the specified value is used. OUTSEQ must be set to restart a Clean (see below).
> GAIN 0.1 C$_R$	to set the loop gain parameter, defaults to 0.1. Values of 0.2 or more may be suitable for simple, point-like sources, while even smaller values may be required for complex sources with smooth structure.
> FLUX f C$_R$	to stop Cleaning when the peak of the residual image falls to f Jy/beam.
> NITER n C$_R$	to stop Cleaning when n components have been subtracted. There is no default; zero means no Cleaning.
> BCOMP 0 C$_R$	to begin a new Clean — see below for restarting one.
> NBOXES 0 ; BOXFIL ' ' C$_R$	to specify no Clean search areas in advance; see § 5.3.3.
> CMETHOD ' ' C$_R$	to allow IMAGR to use DFT or gridded-FFT component subtraction at each major cycle, depending on which is faster. 'DFT' forces DFT and 'GRID' forces gridded subtraction at all iterations. Use the default. DFT is more accurate, but usually much slower; see the explain file for details.
> FACTOR 0 C$_R$	to use the "normal" criteria for deciding when to do a major cycle; see below.
> BMAJ 0 C$_R$	to have IMAGR use a Clean beam which is a fit to the central lobe of the dirty beam.
> DOTV 1 C$_R$	to have dirty and residual images displayed on the TV; see § 5.3.3.
> 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 MAPPR may be used for single-field Cleaning.

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

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

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

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

> BMAJ $bmaj$ 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.

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

Note that the number of Clean iterations, and many of the other Cleaning parameters, may be changed interactively while IMAGR is running by use of the *AIPS* SHOW and TELL utilities. Type SHOW IMAGR 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 TELLs.) Of particular interest is the ability to turn the TV display back on and to extend the Clean by increasing NITER. There are two ways to tell IMAGR that it has done enough Cleaning: by selecting the appropriate menu item in the TV display or by sending a OPTELL = 'QUIT' with TELL. The former can only be done at the end of a major cycle and only if the TV display option is currently selected, while the latter can be done at any time (although it will only be carried out when the current major cycle finishes).

IMAGR makes a *uv* "workfile" which is used in its Clean step to hold the residual fringe visibilities. Its name is controlled with the IN2NAME, IN2CLASS, IN2SEQ, and IN2DISK parameters. If the first three are left blank and 0, the workfile will be deleted when IMAGR terminates. Even if the workfile already exists, IMAGR assumes that its contents must be initialized from the main *uv* file unless the ALLOKAY adverb is set $\geq$ 2.. This file is useful if you suspect that there are bad samples in your data. Use LISTR (§ O.1.5) UVFND (§ 6.2.1), PRTUV (§ 6.2.1), UVPLT (§ 6.3.1) or even TVFLG (§ O.1.6) to examine the file. If you find data which you think are corrupt, remove them from the *input uv* data set with UVFLG. These workfiles may eventually use an annoying amount of disk if IN2SEQ is left 0. Be sure to delete old ones with ZAP in this case.

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

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

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

5.3.2 Multiple fields in IMAGR

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

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

> NFIELD n CR	to make images of n fields.
> IMSIZE i, j CR	to set the minimum image size in x and y to i and j, where i and j must be integer powers of two up to 8192.
> RASHIFT $x_1, x_2, x_3, \ldots$ CR	to specify the x shift of each field center from the tangent point; $x_n > 0$ shifts the map center to the East (left).
> DECSHIFT $y_1, y_2, y_3, \ldots$ CR	to specify the y shift of each field center from the tangent point; $y_n > 0$ shifts the map center to the North (up).

> FLDSIZ $i_1, j_1, i_2, j_2, i_3, j_3, \ldots$ 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 `RASHIFT` which is $(\alpha - \alpha_0)\cos(\delta)$. `IMAGR` has an optional `BOXFILE` text file which may be used to specify some or all of the `FLDSIZE`, `RASHIFT`, and `DECSHIFT` values. It is the only way to specify these parameters for fields > 64. To simplify the coordinate computations, the shift parameters may also be given as right ascension and declination of the field center, leaving `IMAGR` to compute the correct shifts, including any rotation. `BOXFILE` may also be used to specify initial Clean (and "UNClean") boxes for some or all fields, values for `BCOMP`, spectral-channel-dependent weights, a list of "star" positions to be added to plots, and even a list of facets to ignore totally.

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

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

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

There are a number of aspects of multi-field Clean that can trip up the unwary. The first is that the sidelobes of an object found in one field are *not* subtracted from the other fields in the minor Clean cycle. In fact, they are not even subtracted from pixels more than the beam patch size away in the same field. This can cause sidelobes of the strongest sources to be taken to be real sources during the current major cycle. (The `OVERLAP` ≥ 2 sequence reduces this effect significantly.) At the end of the major cycle, all components from all fields are subtracted from the *uv* data. At this point, all sidelobes of the components are gone from all fields, but the erroneously chosen "objects" with their sidelobes will appear (in negative usually). This is normally not a problem. During the next cycle, Clean will put components of the opposite sign on the erroneous spots and they will eventually be corrected. Nonetheless, it is a good idea to restrict the Cleaning to the obvious sources to begin with, saving Clean the trouble of having to correct itself, and to open up the search areas later in the Clean. The TV options make this easy to do in IMAGR; see § 5.3.3.

The situation is more complex if the multiple fields overlap. If a sidelobe in the overlap area is taken as a source in one major cycle, it will appear as a negative source in both fields at the start of the next major cycle (*only when* OVERLAP < 2). Clean will then find negative components in both fields and correct its original error twice, producing a positive "source" at the next major cycle. Such errors never get fully corrected. A simple rule of thumb is never to allow the search areas of one field to overlap with the search areas of another field — or use OVERLAP ≥ 2.. Even then, there is one other "gotcha." In the restore step, Clean only restores components to the fields in which they were found (again, unless OVERLAP > 0). Thus, a real source visible in two fields will be found in only one after Clean; your two images of the same celestial coordinate will be in substantial disagreement. Therefore, you must be careful about which parts of which images you believe to represent the sky. Instead, use OVERLAP ≥ 1 to have Clean components from all fields restored to all fields as needed. If OVERLAP ≥ 2 , the Clean and imaging are done in a fashion which greatly reduces the instabilities arising from Cleaning the same source (or sidelobe) from more than one field.

5.3.3 Clean boxes and the TV in IMAGR

Clean works better if it is told which pixels in an image are allowed to have components. The initial information on this is provided by the FLDSIZE adverb which gives the pixel dimensions of a rectangular window centered in each field in which Clean looks for components. This window can be nearly the full size of the image because the components are subtracted from the ungridded *uv* data. Cleaning windows or "boxes" can be specified with the adverbs:

> NBOXES n CR — to set the number of boxes in which to search for Clean components. Must be ≤ 50; if 0, one Clean box given via FLDSIZE is used and CLBOX is ignored.

> CLBOX $lx1, by1, rx1, ty1, lx2, by2, rx2, ty2, \ldots$ CR — to specify the pixel coordinates of the Clean windows as leftmost x, bottom-most y, rightmost x, topmost y for boxes 1 through NBOXES. Circular boxes may also be specified as -1, radius, center x, center y interspersed in any order with the rectangular boxes. Default is given by FLDSIZE(1).

> BOXFILE '$area:infilename$' CR — to specify the name of a text file listing the Cleaning windows. Blank means no file.

> OBOXFILE '$area:outfilename$' CR — 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. If you need to take your boxes to CASA, MASKS will make a mask image for you.

IMAGR will also create Clean boxes for you automatically. The adverbs IM2PARM(1) through IM2PARM(6) control the process, selecting the maximum number of boxes to be found at any one time, the cutoff levels as factors times the residual image robust rms, and more. IM2PARM(7) controls the starting list of Clean boxes for the next spectral channel — do they start with those input to IMAGR or also include those found in the current channel. See HELP IM2PARM 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 $\geq n > 0$ is the field number first displayed on the TV. Before each major cycle, the current residual image is displayed on the tv with a labeled step wedge (in 31DEC25) and a menu of options is offered to you. (Note that the residual image before the first major cycle is the un-Cleaned dirty image.) The image displayed is interpolated up or decremented down (by taking every $n^{\rm th}$ pixel in each direction) to make it fit on the display and the current Clean boxes are shown. If you do not select a menu option, IMAGR proceeds after 30 seconds, except initially where the default timeout is 10 minutes. IM2PARM(8) and (9) may be used to alter these timeouts.

The interactive options appear in two columns. To select an option, move the TV cursor to the option (remember the left mouse button — see § 2.3.2) and press buttons A, B, or C. Button D will get you some on-line help about the menu option. The basic options, in the order in which they are displayed, are:

ABORT TASK	to stop the task abruptly, destroying the output images and exiting as quickly as possible.
TURN OFF DOTV	to resume the Cleaning now and stop using the TV to display the residual images. To turn the TV display back on, if needed, use the `TELL IMAGR` verb with suitable adverbs, including `DOTV TRUE`.
STOP CLEANING	to stop the Clean at this point, restore the components to the residual images, write them on disk, and exit.
OFFZOOM	to turn off any zoom magnification
OFFTRANS	to turn off any black & white enhancement
OFFCOLOR	to turn off any pseudo-coloring
TVFIDDLE	to interactively zoom and enhance the display in black & and white or pseudo-color contours as in `AIPS`
TVTRAN	to enhance in black & white as in `AIPS`
TVPSEUDO	to select many pseudo-colorings as in `AIPS`
TVFLAME	to enhance with flame-like pseudo-colorings as in `AIPS`
TVZOOM	to set the zoom interactively as in `AIPS`
CURVALUE	to display the pixel value and x, y pixel coordinates at the TV cursor position as in `AIPS`
IMSTAT	to compute and display robust and normal statistics on the current residual image
SET WINDOW	to select a sub-image of the whole to be reloaded with better resolution — all boxes must be included.
RESET WINDOW	to select the full image and reload the display
TVBOX	to set the Clean boxes for this field beginning at the beginning as in `AIPS`. If existing boxes will be deleted, you are asked if you want that. If no, `REBOX` is done instead.
REBOX	to reset the current Clean boxes and create more as in `AIPS`
DELBOX	to delete some of the current Clean boxes as in `AIPS`
UNBOX	to make or re-make "UNClean" boxes for this field
CONTINUE CLEAN	to resume Cleaning now rather than wait for the time out period.
CHAR MULT	to change the character size used in the TV display.

A blank `OBOXFILE` adverb causes a temporary one to be created in `$HOME`. A `TELL` operation can change this adverb to a more permanent file. If `TVBOX`, `REBOX`, or `DELBOX` used, the new Clean boxes will be written to the text file, replacing any previously in that file. (All non-box cards in that file are preserved unchanged; a new `OBOXFILE` will be filled with the non-box cards from `BOXFILE`.)

If `NFIELD > 1`, a sufficient number of additional options appear of the form

SELECT FIELD n	to display field n, allowing its Clean boxes to be altered or
SELECT NEW FIELD	to prompt on the terminal for a new field number to be displayed, allowing its Clean boxes to be altered (when > 64 fields).
SELECT NEXT FIELD	to select the next higher numbered field (when > 64 fields).
SELECT LAST FIELD	to select the next lower numbered field (when > 64 fields).

Thus you can look interactively at the initial dirty images, place boxes around the brightest sources, and start the Clean. As it proceeds and weaker source become visible, you can expand the boxes and add more to include other sources of emission. Do be careful, however. Boxes that are too tight around a source can affect its apparent structure. The author once made Cas A into a square when stuck with a too-tight box. If `OVERLAP = 2`, the `SELECT FIELD` options are displayed when all residual images are current, *i.e.*, at the beginning, but are replaced by the options

REMAKE IMAGES	to re-compute all fields using the current residual *uv* data and then to display all fields on the TV.
THIS IS FLD n	to indicate that only field n is displayed and available to have its boxes altered.
FORCE A FIELD	to prompt on the terminal for a field number, exit TV, re-compute and display that field with current residual data (if needed) and then Clean that field (only available if `NFIELD` > 64 or `DOWAIT` true).
STOP FLD n	to mark field n as Cleaned sufficiently.
ALLOW FLD n	to allow further cleaning of previously stopped field n.
CHECK BOXES	to see if Clean boxes in one field overlap with Clean boxes in other fields and drop some to avoid this situation.

We encourage use of DOTV TRUE C_R when you are Cleaning an image, especially for the first time or when using the options described in the next section. Watching the TV display as the Clean proceeds will help you to gauge how to set up control parameters for future Cleans and how long to iterate. It may also warn you about instabilities in the deconvolution if you compare the appearance of extended structures early and late in the Cleaning process. The instabilities referred to in § 5.2.4 were first seen while Cleaning with the TV option.

5.3.4 Experimental variations on Clean in `IMAGR`

Experimental variations of the familiar Cleaning methods have been introduced in `IMAGR`. One deals with the so-called "Clean bias" which causes the fluxes of the real sources to be underestimated. The other two deal with the inadequacies of Clean in modeling extended sources.

5.3.4.1 Clean-component filtering

It has been found that Clean will eventually assign some components to noise spikes in regions which do not have real sources, producing the "Clean bias." Real source flux is underestimated, presumably because "sidelobes" of the noise "sources" get subtracted from areas of real sources. The magnitude of the effect is rather variable and is not understood. There are two older tasks discussed in § 5.3.6 which deal with the problem. However, it would be better to remove "weak, isolated" (presumably spurious) Clean components as Clean proceeds rather than only after the fact. One cannot do this at every Clean major cycle, since all components are likely to be weak and isolated initially. But it is a good idea to do it a few times while *uv*-plane based Cleans still have the ability to respond to the filtering. Note that this option may be used to remove negative "bowls" surrounding sources imaged with too little short-spacing data. However, it does not remove the bowl that sits on the source itself and so the final flux will be too low. You should consider the multi-scale Clean instead.

To have `IMAGR` filter Clean components set `IMAGRPRM(8)` $\neq 0$. Then `IMAGR` will select only those Clean components having $>$ abs(`IMAGRPRM(8)`) Jy within a radius of abs(`IMAGRPRM(9)`) cells of the component. Note that this rejects all areas of negative flux, unless `IMAGRPRM(8)`< 0, in which case the absolute value of the flux near the component is used. You can change these parameters with `TELL`, but only if `IMAGRPRM(8)` was non-zero to begin with. A copy of the input data has to be made for this option and it is only made if

IMAGRPRM(8) is non-zero. If this option is selected, the output CC files will have been merged. Note that IMAGRPRM(8) should always be ≤ 0 for images of Q, U, and V Stokes parameters since negative brightnesses are valid. Filtering is done on restarts, when requested from the TV, on certain Cleaning failures, and near the exit. Clean is continued after this last filtering if IMAGRPRM(9)< 0, but usually terminates fairly quickly. When this option is available, an addition TV option will appear:

FILTER COMPS	to exit TV, filter all Clean components, and then re-compute all fields using new residual data.

This is the only way to filter before the end of the current Clean.

5.3.4.2 SDI modification of Clean in IMAGR

A modified Clean algorithm that attempts (often successfully) to suppress the striping and bumpiness in Cleans of extended sources has been developed by Steer, Dewdney and Ito (1984, *Astron. & Astrophys.* **137**, 159). In the AIPS modification of the algorithm, Clean proceeds normally until the residual image becomes rather smooth. It then takes many components at once from all high-residual cells rather than trying to decide exactly which *one* cell is the highest. This algorithm attempts to cut the top off the plateau of emission found in the residual image(s) in a relatively uniform way. Unfortunately, it is very expensive to determine the correct weights to use for each pixel in this algorithm. The technique used by IMAGR does apply larger weights to isolated pixels and pixels at the edges of the "plateau" but these weights are still not quite large enough to avoid a tendency to make a slight rim around the plateau. The next cycle of SDI Clean does trim this down and the method converges well for the extended sources for which the algorithm was designed. IMAGR allows you to switch back and forth automatically between BGC and SDI depending on the contrast between the brightest and median residual pixel in the Clean windows. (SDI is used when the contrast is low.)

To allow IMAGR to use the SDI algorithm, specify IMAGRPRM(4) > 0. SDI Clean will be used when the fraction of pixels in the Clean windows exceeding half of the peak residual exceeds IMAGRPRM(4). IMAGRPRM(19) is used to limit the depth of an SDI Clean cycle. When this option is used, additional TV options appear:

FORCE SDI CLEAN	to force the next Clean cycle to use SDI method.
FORCE BGC CLEAN	to force the next Clean cycle to use Clark method.

These allow you to force the choice of SDI or BGC methods for the next Clean major cycle. After that, it reverts to selecting the method based on IMAGRPRM(4) and the histogram of residual values.

5.3.4.3 Multi-scale modification of Clean in IMAGR

Clean has problems with extended sources because the point-component model is so far from the reality for them. Greisen experimented in the 1970s with modeling sources as Gaussians rather than points, but found that there are always point sources in the image which cannot be modeled sensibly with an extended component. ("Bull's eyes" get painted around every point object.) Wakker and Schwarz (1998, *Astron. & Astrophys.* **200**, 312) proposed a scheme in which a smoothed image and a difference image were Cleaned. This still used point-source models although the Clean beam used for restoring the smoothed-image components was extended. Cornwell and Holdaway (July 1999, Socorro imaging conference) described a scheme in which an image is Cleaned simultaneously at several scales.

A *uv*-based variation of this last algorithm is available in IMAGR. The multi-field capability is used to image for each of NFIELD fields, images at NGAUSS scales specified in the array WGAUSS in arcsec. The full-resolution image is convolved with a Gaussian of width WGAUSS(i) while a dirty beam appropriate to a component of that width is constructed. One of the WGAUSS must be zero if a point-source model is desired; a warning is issued if none of the resolutions is zero. OVERLAP = 2 mode is used. See EXPLAIN IMAGR for details of this new option. Users have found that WGAUSS values increasing by factors of $\sqrt(10)$ are frequently optimal. The most important additional control parameter is IMAGRPRM(11) which down-weights higher scales to allow Clean to work on the higher-resolution images with roughly equal probability. FGAUSS is used to select the minimum brightness to be cleaned at each scale; higher values at higher scales are usually desired. Reasonable values of these three parameters will require running IMAGR to determine the point-source resolution and then to determine the peak brightnesses in each scale and the apparent noise levels after some Cleaning. The other available "knobs" for this algorithm may safely be left at zero.[3]

5.3.4.4 Spectral-index corrections

When using wide-band, multi-channel data to image continuum sources, serious effects are seen if the source intensities change as a function of frequency. If these changes can be modeled as spectral index images with or without "curvature," then IMAGR will allow the spectral-index effects to be compensated during Cleaning. First image each spectral channel (or group of closely-spaced channels) separately. Combine them into a cube with FQUBE, transpose the cube with TRANS, and solve for spectral index images with SPIXR. To use these images, set IMAGRPRM(17) to a radius (> 0) in pixels of a smoothing area and put the image name parameters in the 3rd and 4th input image names. Note that this algorithm is expensive, but that it can be sped up with judicious use of the FQTOL parameter.

5.3.5 Data correction options in IMAGR

There are a number of effects which degrade the usual image deconvolution, but which are, optionally, handled differently by IMAGR. These corrections are primarily for observations made with widely spaced frequencies over fields comparable to the single-dish field of view. If you have such data and hope to achieve high dynamic range images, then these corrections are for you. Otherwise skip to the next section.

5.3.5.1 Frequency-dependent primary-beam corrections

The primary-beam pattern of the individual telescopes in the interferometer scales with frequency. Therefore, each channel of multi-frequency observations of objects well away from the pointing center effectively observes a different sky. When a combined source model is produced, there will be residuals in the visibility data that cannot be Cleaned as the data does not correspond to a possible sky brightness distribution. If IMAGRPRM(1) is larger than 0, then a correction is made in the subtraction of Clean components from the *uv* data to remove the effects of the frequency dependence of the primary beam. If the array is known, *i.e.*, if it is the old VLA, new EVLA, ATCA, GMRT, or MeerKAT, the known primary beam parameters of the average telescope in the array are used. Otherwise, the primary beam is assumed to be that of a uniformly illuminated disk of diameter IMAGRPRM(1) meters. This correction is made out to the 5% power point of the beam with a flat correction further out. Note: this correction is only for the relative primary beam to correct to a common frequency and *does not* correct for the primary beam pattern at this frequency. Note that this algorithm is expensive, but that it can be sped up with judicious use of the FQTOL parameter. It adds essentially no cost when doing the spectral-index corrections, however.

5.3.5.2 Frequency-dependent correction for average spectral index

[3] See "Aperture Synthesis Observations of the Nearby Spiral NGC 6503: Modeling the Thin and Thick Disks" 2009, AJ, 137, 4718, by Greisen, E. W., Spekkens, K. and van Moorsel, G. A. for a detailed description of this algorithm and an example of its use.

If the sources observed do not have a flat spectrum, then the source spectrum will have channel-dependent effects on the Cleaning of a similar nature to the primary beam effects described above. This problem does not depend on position in the field except, of course, that the spectral index usually varies across the field. Normally, however, it varies around -0.8 rather than about 0. To the degree that the structure in the field can be characterized by a single spectral index, the amplitudes of the data can be scaled to the average frequency. This is done, before imaging, by scaling the amplitudes of the uv data to the average frequency using a spectral index of `IMAGRPRM(2)`. For optically thin synchrotron sources, this spectral index is typically between -0.6 and -1.0. This correction cannot remove the effects of variable spectral index but allows a single correction which should usually be better than no correction at all. Note that a much more expensive correction and more accurate may now be made (see above).

5.3.5.3 Error in the assumed central frequency

If the frequency used to compute the u, v and, w terms is in error, there will be a mis-scaling of the image by the ratio of the correct frequency to that used. Since central frequencies are frequently computed on the basis of unrealistic models of the bandpass shape, the "average" frequency given in data headers is frequently in error. If `IMAGRPRM(3)` is larger than 0, it is assumed to be a frequency scaling factor for the u, v, and w that is to be applied before imaging. Again, this can only correct for some average error. Since individual antennas will have different bandpass shapes, no single factor can correct all of the error.

5.3.5.4 Array mis-orientation effects

Images made with a coplanar array not oriented towards the instrumental zenith will have a distortion of the geometry which increases in severity away from the phase tracking center. For non-coplanar arrays, the image is distorted rather than just the geometry. VLA snapshots are misaligned coplanar arrays, whereas VLA synthesis images cannot be considered to have been made with a coplanar array. Images made with mis-aligned coplanar arrays can be corrected using task `OHGEO` to remove the effects of this misalignment. Since this correction requires the knowledge of the observing geometry, in particular, the average parallactic and zenith angles, `IMAGR` computes these values and leaves then as header keywords for `OHGEO` to use.

5.3.5.5 Non-coplanar effects

`IMAGR` has a `IMAGRPRM(4)` option to attempt to correct for non-coplanar effects in imaging. If this worked, it would be very very slow. At this writing, it is not believed to work at all and is disabled in the code. See the explain information for further details. The `DO3DIMAG` option removes a good part of the non-coplanar effects by rotating the projected baselines to make each field tangent at its center.

5.3.5.6 Units mismatch of residuals and Clean components

In principle, the units of the residuals are different from those of the restored components. Both are called Jy per beam area, but the beam areas differ; that of a dirty image is — in principle — zero. If the area of the central lobe of the dirty beam is similar to the restoring beam area, then this effect is negligible. Similarly, if the Clean has proceeded well into the noise then this difference is of little consequence. However, if there is significant flux left in the residual image, then this difference may be important. If `IMAGRPRM(5)`> 0, `IMAGR` will attempt to scale the residuals to the same units as the restored components. The principal difficulty is determining the effective area of the dirty beam. Operationally, this is done inside a box centered on the peak in the beam with half-width `IMAGRPRM(6)` in x and `IMAGRPRM(7)` in y. It may be better, in this regard, to use the default Clean beam and this option at this stage and change resolution and units later with CCRES; see § 7.6.4.

5.3.6 Manipulating Clean components

The list of Clean components associated with a Clean image can be printed with:

> TASK 'PRTCC' ; INP	to select the task and review its inputs.
> INDI *n* ; GETN *ctn* CR	to select the Clean image, where *n* and *ctn* select its disk and catalog numbers.
> BCOUNT n_1 ; ECOUNT n_2 CR	to list Clean components from n_1 to n_2.
> XINC n_3 CR	to list only every $n_3^{\rm th}$ component.
> DOCRT FALSE CR	to route the list to the line printer, or use TRUE to route the display to your workstation window.
> GO CR	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 CR	to clear input parameters.
> APARM(6) 1; APARM(10) 1 CR	to have the component flux summed and plotted on the y axis.
> GO TAPLT CR	to create the plot file.
> GO LWPLA CR	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 CR for details.

You can compare the source model contained in the CC file with the visibility data in a variety of ways. UVSUB allows you to subtract the components of some or all fields from the data, producing a residual visibility data set. OOSUB is similar, but allows for the frequency-dependent corrections performed in IMAGR and for limiting the components subtracted to those inside or outside of the primary beam. Of course, IMAGR's workfile already contains these residuals with the CC files of all fields subtracted. Various display options can be used on these *uv* files; see § 5.3.1. VPLOT, described in § 9.4.3, will plot a CC model against visibility data, one baseline at a time, n baselines per page.

The algorithm used by all *AIPS* Cleans assigns to a component only a fraction (GAIN) of the current intensity at the location of that component. As a result, the list of components contains many which lie on the same pixels. CCMRG combines all components that lie on the same pixel. This can reduce the size of the list greatly and, hence, the time required for model computations in tasks such as CALIB (§ 5.4) and UVSUB. Do this with

> TASK 'CCMRG' ; INP	to select the task and review its inputs.
> INDI *n* ; GETN *ctn* CR	to select the Clean image, where *n* and *ctn* select its disk and catalog numbers.
> INVERS *m* ; OUTVER *m* CR	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 input image and beam should have size an integer power of 2 and the reference pixel should be at $N_x/2$, $N_y/2+1$.

The subject of image deconvolution has been widely studied and many methods have been proposed for tackling it. Clean is renowned for yielding images that contain many artificial beam-sized lumps or stripes in smooth low-brightness regions. Point sources are a poor model for such regions. You should compare heavily Cleaned images with dirty, or lightly Cleaned, images to test that any features you will interpret physically have not been introduced by these Clean "instabilities." The *AIPS* Clean tasks have an optional parameter PHAT that will add a small-amplitude δ-function to the peak of the dirty beam in an attempt to suppress these instabilities as described by Cornwell (*Astron. & Astrophys.* **121**, 281 (1983).)

A modified Clean algorithm that attempts (often successfully) to suppress these instabilities has been developed by Steer, Dewdney and Ito (*Astron. & Astrophys.* **137**, 159 (1984)). In this algorithm, Clean proceeds normally until the residual image becomes rather smooth. It then takes many components at once from all high-residual cells rather than trying to decide exactly which *one* cell is the highest. The algorithm is embodied in the well-tested *AIPS* task SDCLN, which is actually an enhanced version of APCLN. The source must be contained in the inner quarter of the image area as in that task. Type EXPLAIN SDCLN C_R for information. SDCLN gives excellent results on extended sources, but is exceptionally CPU-intensive.

The most widely used, best understood, and probably most successful alternative to Clean is the Maximum Entropy Method ("MEM"). This is implemented in *AIPS* by the task VTESS. This requires a dirty image and beam, such as those produced by IMAGR with NITER set to 0, each twice the (linear) size of the region of interest (as for APCLN and SDCLN). The deconvolution produces an all-positive image whose range of pixel values is as compressed as the data allow. The final VTESS image is therefore stabilized against Clean-like instabilities while providing some "super-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.) To reduce the scatter, you may average the data over time and/or over spectral channel before plotting. UVPLT normally prepares the plot in memory before writing it to the TV or plot file. In this way, even very large data sets do not require large amounts of time. Use LWPLA to get hard copy of the plot file.
2. If you can use a point-source model for the first iteration, *i.e.*, if a range of baselines sufficient to calibrate all antennas is dominated by a single component (flat visibility function well above the noise), go to step 6 directly. This is frequently done with VLBI data, but is less common with arrays for which the initial calibrations are better, such as the VLA.
3. If you must use a more complicated model, obtain a Clean-component representation of it by making and Cleaning an image of the externally-calibrated data using IMAGR. Leave the *uv* data in "TB" sort order for CALIB; IMAGR will sort them if it has to. Note that you may want to use a somewhat higher loop GAIN in a Clean to be used as an input model for an early iteration of self-calibration than you would for final deconvolution of a very extended structure. Task FACES can prepare an initial model for wide-field observations particularly at long wavelength.
4. Consider running CCMRG to reduce the number of components in the model. This improves the speed of the calibration and makes the first negative component be a real negative rather than a minor correction to previous positive components. Remember that merging the components does alter the model which is used to compute the gains unless you were going to include all components anyway. Note that IMAGR often merges the components automatically.
5. Use PRTCC or TAPLT (as in the example in § 5.3.6) to help you decide how many components from this Clean to include in the CALIB model. CCFND is also helpful. When you have decided this, determine the appropriate *uv*-limits for the gain solution by referring to the hard copy of the visibility function you made at step 1.

6. Plan your CALIB inputs using the information given in the following two sections. The first few iterations are usually used to correct only phases; amplitude is normally corrected only in the last one or two iterations.

7. Use CALIB to calculate the gain corrections. If DOAPPLY ≥ 0, it will apply them to produce a new, (hopefully) improved data set, and will also catalog the gain corrections as an SN extension to the *input uv* data file. You may set DOFLAG to produce and use new data flags based on closure failures.

8. Use SNPLT on the input data file with DOTV = TRUE to review the gain corrections before proceeding further. Use DOBLANK = 1 to plot failed solutions as well as good ones. To take hard copy for future reference, run SNPLT with DOTV = FALSE and then run LWPLA on the plot files (usually more than one) produced. To plot the extrema of the gains use OPTYPE = 'SUM' in SNPLT.

9. Ask whether the gain corrections were believable — were they smaller than at the previous iteration of CALIB, if any? If not, is there a good reason why not? Did you change input parameters such as the model, the type of solution, or the solution interval, in a way that may have forced larger corrections than before? Proceed only if you are reasonably sure you understand what is happening at this point — otherwise consult a local expert at your site.

10. If the corrections were believable, run IMAGR to produce a new Clean image. Lower GAINs and higher NITER to produce deeper and more careful Cleans are appropriate as the self-calibration progresses.

11. Go back to step 4 and repeat the whole process if your new Clean image is a significant improvement over the previous one (with comparable Cleaning parameters on both occasions). You may want to go back to step 1 and repeat the process from there if you have been using amplitude self-calibration and wish to check that your amplitude calibration has not drifted significantly. If the new Clean image differs little from the previous one, do not continue on with further iterations of steps 4 through 10 unless you feel you can make an informed change to the CALIB input parameters at step 6. Tasks UVDIF (§ 6.2.1), DIFUV, and UVADD may help you to decide whether there have been significant changes to your data due to the previous iteration of CALIB.

12. For very high dynamic range imaging, it may be useful to check for baseline-dependent (non-closing) multiplicative complex gains. Tasks BLCAL and BLCHN perform this calibration which is normally run to get a single solution for the entire range of times in the data set. In 31DEC23, procedure BLPOLCAL extends this operation to the cross-hand visibilities. See § 9.5.9 for additional discussion.

13. Images may also be used as models (CMODEL='IMAG'). This works well (in 31DEC21) if the image is a "model image" rather than a Clean image. The Fourier transform of the latter does not agree with the visibility data due to the convolution with the Clean beam. One could experiment, in 31DEC21, with a Clean image in UVSUB with OPCODE='MODL' followed by DGAUS which (partially) corrects for the deconvolution. A UVADD with OPCODE='DIV' will construct an "already divided" data set which is understood by CALIB.

If you have used the spectral-index (§ 5.3.4.4) and/or primary beam (§ 5.3.5.1) options in IMAGR, you may continue to use those options in self-calibration. The procedure OOCAL will divide your multi-facet, channel-dependent model into the visibility data with task OOSUB. It then runs CALIB on that output and copies the resulting SN table back to the original input file.

The tasks SCMAP and SCIMG attempt to implement this sequence inside a single task. SCIMG contains almost all of IMAGR and all of CALIB. SCMAP is similar, but limited to a single field for simplicity. They attempt to make the decision about the number of merged components and the range of *uv* spacings to use in each self-calibration based on σ times the rms in the residual image of the current Clean, where you provide the σ. The process is somewhat less flexible, but also less painful, than running CALIB and IMAGR multiple times. They do not let you change imaging parameters while they are running, but they do provide automatic and interactive methods to change and save Clean boxes and to set a variety of Cleaning and self-calibration parameters including loop gain, solution interval, solution smoothing interval, and whether negative components are included or terminate the list. They let you switch from phase-only to amplitude and phase self-calibration or they will do it automatically when the phase only stops converging. Both tasks offer the full editing options of task EDITR (see § 5.5.2) displaying the input and current residual *uv* data with a wide variety of data selection and editing options. The CALIB, IMAGR, and EDITR process is similar, but conceptually simpler, so it is the one described here.

5.4.2 Self-calibration with CALIB

CALIB is the heart of the AIPS calibration package. The inputs to CALIB are extensive and spread over several screen pages. This is because the routines in CALIB are used in many situations — general calibration, real-time interferometry and VLBI. The task solves for antenna-based complex gains *i.e.*, "self-calibration," whether the source being calibrated is a "calibrator" source (usually taken to be a point) or a "program" source (usually taken to be complex). The solutions that CALIB generates are stored in SN "solution" tables which are attached to the *input* data file. The SN tables can be plotted with SNPLT and listed with LISTR. They can be edited themselves with SNEDT or be used to edit the *uv* data with EDITA. If the corrections are usually small, the OPTYPE='A&P' option in SNFLG may perform the flagging you would do with EDITA, but with very little effort on your part.

The following input parameters are used by CALIB for self-calibration of a single-source *uv* data set:

> TASK 'CALIB' ; INP C_R	to specify the task and review the inputs.
> INDI *n1* ; GETN *ctn1* C_R	to select the 'TB' sorted *uv* database.
> IN2D *n2* ; GET2N *ctn2* C_R	to select the Clean model image(s) to use.
> INVERS m C_R	to specify the CC file version number to use from *every* model image; 0 means the highest.
> NMAPS q C_R	to specify the number of images with CC files to use for the model. If $q > 1$, the image class names are assumed to have the first three characters of IN2CLASS with the field number one given in the last three characters as is done by IMAGR.
> NCOMP = $n_1, n_2, \ldots$ C_R	to cut off the model at the $n_i^{\rm th}$ Clean component in the $i^{\rm th}$ image.
> 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) $\rightarrow$ UVRANGE(2). WTUV = 0 is interpreted as zero weight and should *not* be used.
> REFANT n_r C_R	to select the reference antenna; for best results, choose one known to be good over most of the time range.
> SOLMODE 'A&P' C_R	to solve for amplitude and phase corrections simultaneously.
> SOLMODE 'P' C_R	to solve for phase weighted by amplitude, the default for single-source files.

> SOLMODE 'P!A' C_R	to solve for phase ignoring amplitude.
> SOLTYP ' ' C_R	to use a normal (non-linear) least squares solution.
> SOLTYP 'L1' C_R	to use an "L1" solution method in which a weighted sum of the moduli of the residuals is minimized. The computed gain solutions are less influenced by wild data points, but there is some loss of statistical efficiency and a modest increase in compute time. See F. R. Schwab, VLA Scientific Memo #136 for further details.
> SOLTYP 'GCON' ; SOLMOD 'GCON' C_R	to solve for amplitude and phase using least squares with a gain constraint — this requires `GAINERR` and `SOLCON` as well; see the help file.
> ANTWT $w_1, w_2, w_3, \ldots$ C_R	to apply additional weights to each antenna (in order) in generating the solutions; 0 implies 1.
> APARM(1) = x_5 C_R	to reject solutions from fewer than x_5 antennas; default is 6.
> APARM(2) = x_6 C_R	to tell `CALIB` whether the data have already been divided by a model ($x_6 > 0$) or not.
> APARM(3) = x_7 C_R	to solve for RR and LL separately ($x_7 \leq 0$) or to average RR and LL correlators before solving ($x_7 > 0$).
> APARM(5) = x_8 C_R	to make separate solutions for each IF ($x_8 \leq 0$) or to average all IFs to make a single solution ($x_8 > 0$). It is better to do separate solutions unless you are desperate for signal to noise.
> APARM(6) = x_9 C_R	to set the level of diagnostic information as 0 (very little), 1 (some including time and 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 thy 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 uv 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 table contains all of the corrections so far found. Both methods have things to recommend them; DOAPPLY controls the choice.

5.4.3 Considerations in setting CALIB inputs

In many cases, only a few input parameters to CALIB need be set, other than those selecting the *uv* data and the input model. The key parameters are NCOMP, UVRANGE, SOLINT and, if you are interested in polarization, REFANT.

It pays to be conservative when using NCOMP to select the number of Clean components which will comprise the input source model. Setting NCOMP too high will fossilize errors from the earlier calibrations in the model for the next one; after this, you are stuck with them as long as you continue feeding CALIB a model with as much Cleaned flux density. When calibrating Stokes I images, consider setting NCOMP in CALIB so that few negative Clean components are included. The first few iterations of CALIB should be phase-only calibration, since the tropospheric and ionospheric phase errors will almost always dominate amplitude errors due to the atmosphere or to system drifts. In these first iterations, it is prudent to be even more conservative, setting NCOMP so that the total Cleaned flux included in the model is between 50% and 80% of that at which the first negative Clean component appeared. CCFND will help you with this (§ 5.3.6). If your field is dominated by a few very strong, small-diameter regions, it is a good idea to make the first iterations of CALIB work on Clean components from these regions alone, restricting the range of baselines suitably by setting UVRANGE(1). Setting Clean windows in IMAGR or using CCEDT (§ 5.3.6) suitably will help you do this. Even later in the self-calibration cycle, it is probably still a good idea to eliminate weak, isolated Clean components. Try CCSEL for this.

It is always important to restrict the high-weight domain of the CALIB solution to the part of the *uv* plane that is described well by the model. In the early stages of self-calibration, the trustworthy part of your Clean model will almost always contain less flux density than was measured in the visibility function at the shorter baselines. Another way of putting this is that the large-scale structure of the source will be poorly represented by the model. You should therefore set UVRANGE(1) so that the total flux density in the input model (the sum of the Clean components up to the Clean iteration selected by NCOMP) exceeds the peak visibility amplitude in your data at a baseline of UVRANGE(1) kilo wavelengths (read this off a plot file output from UVPLT). It is also important to give some slight weight to the rest of the *uv* plane so that some solution may be found for most all antennas including those having no baselines in the high-weight region.

SOLINT sets the length of the time interval, in minutes, over which the model and the data are averaged when computing the gain corrections. This must be *short* enough that the gain corrections can track the fluctuations produced by the atmosphere over the longer baselines with sufficient accuracy. It must be *long* enough that the variances of the computed gain corrections (which depend on the signal-to-noise ratios in the data over the *uv* range in which the model is being compared with the data) are acceptably small. These constraints vary from source to source, frequency to frequency, and (because of the "weather") from day to day. They may not in fact be reconcilable for weak sources, especially in the wider VLA configurations and/or at the higher frequencies. In many combinations of these circumstances, you may not be able to self-calibrate your data. See Lecture 9 in *Synthesis Imaging in Radio Astronomy* for details of how to make this assessment. In VLBI imaging, it may be helpful to use a point-source model and quite small SOLINT for the first iteration of self-calibration to remove the gross and rapid changes due to atmospheric fluctuations. With that problem removed, it may then be possible to use longer SOLINTs and more complicated models.

REFANT selects the number of the reference antenna for the gain solutions. For total intensity continuum calibration, the choice of this CALIB input is unimportant. It is always best, however, to choose a reference antenna that was stable and present in all data throughout the run, if only because this prevents propagation of noise or glitches in the reference antenna through the gain solutions (and plots of them) for the other antennas. For polarization work, it is important to select an antenna for which both polarizations were always present; otherwise any polarization calibration which preceded CALIB may be seriously compromised.

Note that CALIB should almost always be run with SOLMODE set to phase-only calibration for the first iteration or two. Consider turning on amplitude calibration by setting SOLMODE 'A&P' only when either the phase adjustments being made are generally small (*i.e.*, the worst cases being a few tens of degrees) or the new re-Cleaned image is clearly dominated by amplitude errors — which will give symmetric Y-shaped patterns around strong point sources for VLA observations. In general, you will want to set CPARM(2) = 1 when using SOLMODE 'A&P', to prevent drifting of the flux-density scale during amplitude self-calibration.

CALIB has a number of new options to deal with difficult data. The adverb WEIGHTIT controls how the data are weighted when being processed by the gain-fitting routines. The default is $w = 1/\sigma^2$ which may cause too much contrast between the highest weighted points and the lowest. This problem is much worse when self-calibrating extended sources than when doing the primary calibration on point sources. If you encounter many failed solutions, try WEIGHTIT = 1 which uses $w = 1/\sigma$. If you have trouble with bad data and failed solutions, consider trying the "robust" forms of SOLTYPE selected with 'R', 'L1R', and 'GCOR'. A robust solution is one in which a solution is found, outlier data are temporarily flagged, a new solution found, and the process repeated while gradually tightening the omission criteria. This should make for more reliable solutions when there are bad correlators or antennas and, as a side benefit, allows more permanent flagging of the data under control of adverb DOFLAG.

CALIB will also edit out bad data according to the following criteria:

1. there are too few antennas (APARM(1)) to form a solution,
2. the solution does not converge, or
3. the signal-to-noise ratio for a given antenna (APARM(7)) is too low.

The signal-to-noise ratio is calculated from the post-fit scatter of the residuals from the gain model. Note that the scatter will contain contributions from thermal noise *and* unmodeled source structure. This is a good reason to restrict the *uv* range of the data. For further guidance and information on other CALIB inputs, type EXPLAIN CALIB 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 Evaluating the quality of the imaging

If your imaging were perfect, then subtracting the Clean component model from the visibility data should produce visibilities differing from zero only by the noise. Similarly, dividing the data by the model should produce "gains" of (1,0), namely a 1 "Jy" point source at the coordinate origin. There is a task called `EVAUV` to help you find out how close you have come to these ideal results. `EVAUV` first produces two data sets by subtracting the model from, and dividing the model into, your visibility data. It then determines the mean and rms of the real and imaginary parts of both data sets using a robust algorithm. Then it reports on the number of "bad" samples more than 3 rms from the mean. In `31DEC22`, if `SOLINT` is not zero, it will report rms closure phase and amplitude of the divided data set.

Various optional plots may be produced. They are: (1) The average and rms of the model subtracted data in bins of *uv*-plane radius. (2) The average and rms of the model divided - 1.0 data in bins of *uv*-plane radius. (3) The histogram of image pixels over a pixel value range selected by the user or the first of the model images. (4) The histogram of image pixels over a pixel range of plus/minus 5 times the rms of the first of the model images. (5) The real vs imaginary visibilities of the model subtracted data over a range of `APARM(1)` times the amplitude rms. The plot is a contour in logarithmic intervals of the 2-D histogram image controlled by `APARM`. Contours are drawn at 0.5, 1, 1.5, 2, 2.5, 3, etc in the log of the counts. (6) The real and imaginary gains minus (1,0) of the model divided data over a range of `APARM(2)` times the amplitude rms. Contours are again logarithmic and are displayed on the plot.

Closure phases and amplitudes are useful in evaluating the quality of a calibration source or a gain file produced by dividing a model into the visibility data with `UVSUB`, `OOSUB`, or `EVAUV`. Tasks `CLPLT` and `CAPLT` will plot closure phases and amplitudes, respectively, as functions of time for specified phase triangles and amplitude quadrangles. In `31DEC22`, task `EVACL` may be used to determine the rms closure phase and amplitude as a function of IF (spectral window). A good point source (or gain) should have values near zero for both. Task `CLOSE` will plot the rms phase or amplitude closure as a function of spectral channel. This may be useful in determining which channels should be used in continuum observations and which should be omitted.

5.4.5 Experimental extension of multi-field self-calibration

High-quality, multi-field images often suffer from position-dependent calibration effects. At lower frequencies, the strongest sources may well lie outside the main portion of the primary beam and so be effected by different instrumental gains including pointing errors, differences in atmosphere or ionosphere, etc. An experimental `RUN` file `PEELR` compiles a procedure by that name. One "interfering" field at a time, it performs a self-cal to improve that facet. After a list of fields is processed, it restores the original multi-field model to the corrected residual *uv* data. See `HELP PEELR` for details. It seems almost magical, but it really does improve the final images. A slight increase in the image rms is the price one pays for removing larger, more systematic problems.

5.5 More editing of *uv* data

5.5.1 General remarks on, and tools for, editing

There are many programs which aid in the processing, display, and editing of *uv* data. Summaries of this software may be listed on your terminal with:

`> ABOUT UV` C_R to list all *uv*-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 § 4.3.5 before investing large amounts of time in editing even at this stage.

There are facilities in CALIB, RFLAG, FLAGR, CLIP, CORER, UVMLN, FLGIT, DEFLG, and SNFLG to flag *uv* data in *AIPS* based on deviations from specified norms. There is also the task UVFLG to flag and unflag by antenna-IF or by correlator. The task UVPLT plots various combinations of *uv* data; see § 6.3.1. The task WIPER makes a similar plot on the TV and allows you to wipe away offending data. The task UVFND is also recommended for printing out suspicious portions of the database; see § 6.2.1. Note that CLIP examines the data correlator by correlator, but UVFND normally converts the data to Stokes components (using the same criteria as UVMAP) before checking that the amplitudes and/or phases are in range. To examine the correlators individually, use STOKES 'CORR' in UVFND, or to flag the data based on their values after conversion to true Stokes use STOKES = 'IQUV' in CLIP. Task FINDR is a companion to FLAGR intended to assist you in determining what is normal within your data.

CLIP is also useful for flagging discrepant data (*e.g.,* due to interference or malfunctions) on the basis of their deviations from the visibility predicted by a set of Clean components. In 31DEC23, it even offers an option to flag on excessive differences betwen the two parallel-hand polarizations. Tasks OOSUB or UVSUB will subtract the Fourier transform of a set of Clean components from visibility data. You may then use UVPLT to display the residual *uv* data set and CLIP to flag abnormally high points. You may wish to be cautious, and run UVFND to display such points before running an automatic CLIP task — be especially careful not to CLIP away evidence for real extended structure near the center of your *uv* plane! Before re-imaging, you must copy, with TACOP, the flag table produced by CLIP to the data set used for imaging. Note that IMAGR's workfile is also a *uv* data set from which the current Clean component model has been subtracted. It may also be used with UVPLT to help you to diagnose problems. If you run CLIP on it, you will need to append the resulting FG table to one on the imaging data set; use TAPPE.

FFT is another useful tool for finding suspicious data. Transform your image back into the *(u,v)* plane by running FFT and then display the results on the TV. Use image read-back verbs like CURVALUE and IMPOS (§ 6.4.5) to find the u and v values for abnormally high cells. Then use UVFND with OPCODE 'UVBX' to print the data surrounding these cells and UVFLG to delete any bad data. This method is particularly effective when applied to residual images from Clean. (You can instruct IMAGR to put out a residual image by setting BMAJ <0. CCRES can also make a residual image from a single-field normal image.)

Task UFLAG makes a uv-plane grid (with pill-box convolution) of your visibility data selected by adverbs much like those used in IMAGR. It shows you the vector and scalar averaged grids in amplitude and the vector averaged grid in phase and offers a variety of editing options. You may choose to mark an entire cell as "bad" or look at the visibilities contributing to cells you select and choose to mark some of them (rather than all) as bad. There are even automatic routines to do this last operation over cells meeting criteria that you set. The entire process is described in an *AIPS* Memo.[4]

There is a task called FLAGR which goes through a data set determining what are normal rmses and weights and then flagging those that deviate excessively including clipping all those that have amplitudes or weights outside specified normal ranges. FLAGR is intended for use eventually in pipeline data-reduction procedures, but at present should be considered experimental, but potentially very valuable. Before calibration, try it on your calibration sources using default values for most adverbs plus SOLINT set to 2.5 times the basic integration time and VECTOR set to 0. Task FINDR is a companion intended to determine

[4] Greisen, E. W. 2025, "Editing on a uv grid in *AIPS*" AIPS Memo 121 revised,
https://www.aips.nrao.edu/aipsdoc.html

what is normal in the data and then to print those values and return selected adverb values to AIPS for use by procedures.

There is a task for automatic flagging called RFLAG. Since RFI tends to be highly variable over time and/or frequency, RFLAG accumulates statistics about this variation which it plots along with recommended clip levels. Running it a second time, with DOPLOT ≤ 0, causes the clip levels to be applied to make a large flag table. This appears to be a very successful program with EVLA wide-band data (§ 4.3.13). Task REFLG may then be used to compress and extend the flag table.

TVFLG, SPFLG, FTFLG, WIPER, IBLED and EDITR are TV-based, interactive editors. TVFLG is most suitable for data sets with large numbers of baselines, *e.g.*, the VLA, but it can be used usefully for VLBI data experiments with 10 or more antennas. TVFLG allows you a global overview of your data and can display the data for all baselines simultaneously as a function of time. This task is documented extensively in A 2025 AIPS Memo[5] and § O.1.6 of the CookBook. SPFLG is a very useful task for data with a significant number of spectral channels. It is effective in examining data for frequency-dependent errors and interference and can be an effective data editor for interferometers with a small number of baselines; see the new Memo and § 10.2.2 and § 8.1. FTFLG is like SPFLG, but combines all baselines into one plot. This is a much faster way to look for global RFI, but it must flag all baselines with each flag generated. IBLED has a different philosophy; it plots one baseline at a time in a graphical rather than gray-scale (image) fashion. It is able to average data over time, spectral channels, and/or IFs to make a more manageable amount of data and to measure the "decorrelation index" which is a measure of how variable the phase is over the averaging intervals. The capability of averaging IFs and displaying decorrelation may be of special interest for VLBI data sets. Otherwise, IBLED has been replaced in 15APR98 by EDITR, which also uses the graphics planes rather than gray-scale images but which can plot multiple baselines to a chosen antenna and can display two data sets at the same time. This is obviously more useful for smaller arrays — *e.g.*, VLBI, MERLIN, and the Australia Telescope. This task is described below.

The editing task WIPER should be used with caution. It makes a plot like UVPLT of almost any parameter of a *uv* dataset against any other parameter. The plot is displayed on the TV and you may "wipe away" any points you do not like one point at a time or many at a time with a "fat brush." The task will be very useful for fields with a well-behaved visibility function seen with good signal-to-noise. It may also be useful with data sets from which a fairly good IMAGR model has been subtracted with UVSUB. It allows you to plot and edit any reasonable STOKES polarizations in one execution and displays which baselines enter into each plotted point.

The task FGCNT may be used to count the number of correlators flagged by a particular flag table. Many of the tasks above can generate very large flag tables and can, if the choice of parameters was not right, flag way too much data. This task provides a way to check on this.

5.5.2 Baseline-based *uv*-data editing — EDITR

EDITR is a very effective editing tool from the beginning of data analysis on data sets with modest numbers of antennæ. Since it can display two data sets at the same time for comparison purposes, EDITR may also be used to good purpose with larger data sets during the self-calibration and imaging stage. The visibility amplitude or phase or the amplitude of the visibility with a running vector average subtracted may be displayed. 31DEC25 added a coherence parameter defined as the amplitude of the scalar averaged visibility divided by the vector averaged visibility minus one. The data for the selected baseline are shown in an edit window at the bottom of the display. Optionally, a second observable (*e.g.*, phase) from the selected baseline is shown in the same color in a window directly above the edit window. This option is controlled by the DOTWO adverb. Data for 0 to 10 other baselines to the selected antenna may be displayed in a different color

[5] Greisen, E. W. 2025, "Interactive flagging in AIPS", AIPS Memo 127, https://www.aips.nrao.edu/aipsdoc.html

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 2 C_R	to allow all IFs and all polarizations to be displayed and edited at one time, with the 2 speeding the display.
> 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 § 4.3.11, we do not include a figure here; see Figure 4.2. The upper left corner of the display is reserved for displays of the selected data sample during editing while the bottom left corner is used for status information including flagging options. Menus, discussed below, appear down the left and right sides of the screen. The data are displayed in a stack of plots in the center of the screen. At the bottom are the data from the selected baseline in the primary observable; then the data from the primary baseline in a second observable (if DOTWO is true), and finally the data in the primary observable from 0–10 other baselines to the primary antenna. Data which have been flagged are shown in a different color. The data are plotted on a linear axis vertically, while the horizontal axis is monotonic but irregular in time. Tick marks are plotted at integer hours and the time interval of the edit area is indicated by times at the left and right ends of the axis. The time range displayed in all plots may be selected interactively and editing may therefore be done in crowded full time-range plots or in well separated short time-range plots. Surrounding the plot are various annotations describing the data plotted and the status of the various flags which control which data will be deleted on the next flagging command. If a second data set was specified, then data from that file are displayed in a different color in the same plot areas used for the primary data set.

The interactive session is driven by a menu which is displayed on the same screen as the data. Move the cursor to the desired operation (noting that the currently selected one is highlighted in a different color on many TVs) and press button `A`, `B`, or `C` to select the operation. Press button `D` for a short explanation of the selected operation. The right-hand column contains options to select which data are displayed and to select which data are flagged on the next flag command. The menus are changed to adapt to the input data in order to avoid, for example, offering options to select IF in a one-IF data set. The left-hand column contains 7 interactive modes for editing the data plus options to set the display ranges and scan averaging length, to turn on error bars in plotting samples, to review, alter, and re-apply the existing flag commands, to defer or force a TV display, to switch to entering commands from the keyboard instead of the menu, and to exit with or without applying the current flag commands.

The right-hand menu can contain

NEXT POL/IF	To switch to viewing the next correlator, switching to the other polarization and, if needed, incrementing the IF.
SWITCH POLARIZ	To switch to viewing and editing the other polarization, cycles through both if `CROWDED` was true.
SWITCH ALL POL	To switch functions from applying to one polarization to applying to both polarizations or vice versa.
ENTER IF	To select which IF is viewed and edited. When `CROWDED` is true, zero means all.
SWITCH ALL IF	To switch functions from applying to one IF, to applying to a range of IFs, to applying to all IFs.
SWITCH ALL TIME	To switch `FLAG ABOVE` and `FLAG BELOW` between all times and the time range of the frame.
ROTATE ALL ANT	To rotate functions from applying to (a) one baseline, (b) all baselines to the main antenna, and (c) all baselines.
SWITCH ALL SOURC	To switch between flagging only the current source and flagging all sources.
ENTER ANTENNA	To select the main antenna, baselines to which are displayed on the screen.
ENTER OTHER ANT	To select up to 11 other antennas to define the baselines to be displayed; enter 11 numbers, 0's are then ignored (to plot 5 enter the 5 plus 6 0's). The first one is used for the edit area.
NEXT BASELINE	To advance the list of other antennas, selecting the next one for the edit area.
NEXT ANTENNA	To select a new main antenna, one higher than the current main antenna. The "others" will also be adjusted if appropriate.
PLOT ALL TIMES	To display all data for the selected baselines.
SELECT FRAME	To select a window into the current data interactively.

NEXT FRAME	To select the next time range window of the same size as the current frame.
PREVIOUS FRAME	To select the previous time range window of the same size as the current frame.
SHOW AMPLITUDE	To display and edit amplitudes.
SHOW PHASE	To display and edit phases.
SHOW DIFF AMPL	To display and edit the amplitudes of the vector difference between the sample and its running mean.
SHOW COHERENCE	To display and edit the coherence parameter which is the ratio of the scalar average amplitude divided by the vector average amplitude - 1 over the "scan" length.
SHOW ALSO AMPL	To display amplitudes of the edit baseline for reference with the phase, difference amplitude, or coherence parameter edit window.
SHOW ALSO PHASE	To display phases of the edit baseline for reference with the amplitude, difference amplitude, or coherence parameter edit window.
SHOW ALSO DAMP	To display difference amplitudes of the edit baseline for reference with the phase, amplitude, or coherence parameter edit window.
SHOW ALSO COHE	To display the coherence parameter of the edit baseline for reference with the phase, amplitude, or difference amplitude edit window.
TV ZOOM	To alter the display zoom used while in the flag functions.
OFF ZOOM	To turn off any zooming.
2ND UV OFF	To disable the display of the 2^{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 1 or 2, then you may also choose to display and edit both polarizations and/or all IFs at the same time and input adverb `DO3COL` determines if polarizations and IFs are shown in different colors. The `NEXT CORRELATOR` cycles through all polarizations and IFs, show one at a time. The `SWITCH POLARIZATION` option switches the displayed polarization, the `ENTER IF` option prompts you if necessary for a new IF number, the `ENTER ANTENNA` option prompts you for a new primary antenna number, and the `ENTER OTHER ANT` prompts you for up to 11 other antenna numbers to select the main editing baseline and up to 10 secondary baselines to the primary antenna. (Note that you have to type in 11 numbers, but zeros are then ignored.) A flag command can apply to one or both polarizations and to one, some, or all IFs. It can apply to one baseline, to all baselines to the primary antenna, or to all baselines. The `FLAG ABOVE` and `FLAG BELOW` commands can apply only to the time range displayed in the data "frame" or they can apply to the full time range in the data set. The `SWITCH ALL POL`, `SWITCH ALL IF`, `ROTATE ALL ANT` and `SWITCH ALL TIME` options control these choices and the current state of these switches is displayed at the lower left of the TV screen. The task is able to zoom the display during interactive editing operations if you should need magnification to see what you are doing. The `TV ZOOM` and `OFF ZOOM` options let you control this. In larger data sets, however, a more useful display is obtained by interactively selecting a narrower time range with the `SELECT FRAME` option. To step forward and back through the frames, use the `NEXT FRAME` and `PREVIOUS FRAME` options, respectively. To display *uv* data amplitude, select `SHOW AMPLITUDE` and to display *uv* data phase, select `SHOW PHASE`. You may also display the difference between the current data sample and a running vector average of the data centered on the current sample and extending no more than plus or minus the "scan length" divided by two. To display the amplitude of the vector difference, select `SHOW DIFF AMPL`. Such displays are particularly sensitive to short-term problems while ignoring longer-term changes due to source structure. To display and edit the coherence parameter select `SHOW COHERENCE`. Note that a short-term disturbance in the visibility may cause an apparently longer-term disturbance in this parameter because of the time averaging. Since it takes time to compute things for, and display, the second data set, you may wish to turn it off part of the time. The `2ND UV OFF` and `2ND UV ON` options control this choice.

The left-hand menu can contain

FLAG TIME	To delete one time at a time.
FLAG TIME RANGE	To delete one or more time ranges.
FLAG BELOW	To delete all displayed times with data below a cutoff value.
FLAG ABOVE	To delete all displayed times with data above a cutoff value.
FLAG AREA	To delete one or more areas in the data-value *vs* time plane.
FLAG POINT	To delete one sample at a time using both horizontal and vertical cursor position.
FLAG QUICKLY	To delete samples using only mouse clicks
CLIP EVERYTHING	To delete all displayed times with data above a cutoff value. All baselines are flagged in the IFs and polarizations currently displayed in the edit window at whatever level was set.
ENTER AMPL RNG	To select the display range for amplitude plots. Use $0 - 1$ for zero to maximum, 00 for minimum to maximum.
ENTER PHASE RNG	To select the display range for phase plots.
ENTER DAMP RNG	To select the display range for plots of the amplitude of the visibility minus a running vector average visibility.
ENTER COHE RNG	To select the display range for plots of the coherence parameter.
PLOT ERROR BARS	To plot error bars based on data weights.
SET SCAN LENGTH	To set the averaging time used to determine the running average in seconds.
LIST FLAGS	To list all flags now in the Flag Command table.
UNDO FLAGS	To undo one of the flag operations in the `FC` table
REDO FLAGS	To reapply all remaining flags after one or more have been undone
SET REASON	To set the "reason" string to be put in the *uv*-data flag table.
USE EXPERT MODE	To control the task from the keyboard instead of the menu.
HOLD TV LOAD	To stop updating the TV display with every change of parameter; change several, then select
DO TV LOAD	To update the TV display now and with each change of display parameter.
REPLOT	To do the current plot over again, recomputing the differences from the running mean if appropriate.
CHAR MULT	To change the character size on the TV display.
EXIT	To exit `EDITR`, moving the `FC` table to a *uv*-data `FG` table.
ABORT	To exit `EDITR`, deleting the `FC` table.

The first eight items select interactive flagging modes to delete all selected data at a single time, over a range of times, over all values below or above a specified value, or within a range of times and values (respectively). When one of these options is invoked, the screen zooms (if set to do so), a line or box appears in the editing window, and a display of the sample (source, time, value) under the cursor appears at the upper left. Follow the instructions in the message window to select and edit data. Note that this is a very good way to look at your data values even if you do not want to delete anything. The `FLAG QUICKLY` method is very efficient, but it requires caution in its use. Whenever the left mouse button is depressed, the sample closest to the cursor position is flagged. The `CLIP EVERYTHING` option is particularly powerful and should be used with caution. The next four options set the range of amplitudes, phases, difference amplitudes, and coherence displayed. These default to the full range in the data (separately and differently for each baseline) and can be set back to default by entering `0 0`. The `SET SCAN LENGTH` option prompts you for a "scan" length in seconds used as the averaging interval for computing the running mean used in the difference displays. A longer scan length takes longer to compute, but is likely to be less noisy and more meaningful as an editing tool. If you are not using the difference display, set the scan length to a short interval. The running mean is not carried between sources and, as a result, is not normally carried across actual scan boundaries.

When you execute a flagging option, one or more lines are written to a flag command (`FC`) table attached to the input data set. If `EDITR` dies abnormally, this `FC` table can even be used in a later session. To list all of the flagging commands now in the table select the `LIST FLAGS` option. If you decide that you no longer want one of these flags, select `UNDO FLAGS` and enter the number (from `LIST FLAGS`) of the undesirable flag command. More than one flag command may apply to the same datum. After undoing flags, it is such a good idea to redo them that `REDO FLAGS` and tt REPLOT are now done automatically (in `31DEC20`). This is to make sure that everything is consistent. When the flag commands in the `FC` table are entered into a normal flag table, a 24-character "reason" is attached which is both descriptive and can even be used in `UVFLG` when removing entries in the `FG` table. The `SET REASON` command prompts you for the reason to be attached to subsequent flag commands. The default reason is the task name, time and date. Normally, `EDITR` updates the display whenever anything is changed. If you are about to change more than one display parameter (*i.e.*, polarization, IF, antenna, other antennæ, frame) before doing more editing, select `HOLD TV LOAD` to defer the display update until you select the `DO TV LOAD` option. If the display appears not to be current, select the `REPLOT` option. Finally, you may exit the program with the `EXIT` or `ABORT` options. The former applies your editing to a flag (`FG`) table attached to the input data set, while the latter discards any editing commands you may have generated.

Note that value-dependent flagging (`FLAG BELOW`, `FLAG ABOVE`, `FLAG AREA`, and `CLIP EVERYTHING`) use the values currently plotted to make a list of value-independent flag commands, namely a single time for the specified antennæ, IFs, polarizations, etc. When a value-dependent flag operation is undone with `UNDO FLAGS` or redone with `REDO FLAGS`, it is these value-independent flags which are undone or redone. You may have to undo more commands and then repeat flag commands to get the results you could have gotten by doing the now desired value-dependent command in the first place. You need also to be careful with the `ROTATE ALL ANT` setting with these value-dependent commands. If one baseline is set, then the commands only apply to the current baseline. If one antenna is set, the commands apply to all baselines to the current main antenna, while if all antennas is set, the commands apply to all baselines. The first three set an area or a clip level, below or above which data are deleted, based on the value of the observable in the baseline in the main edit window. The values of the data from other baselines and antennas are ignored. `CLIP EVERYTHING` ignores the setting of the `ROTATE ALL ANT` parameter and applies the clip based solely on the data value in each baseline.

Be careful when choosing `EXIT` versus `ABORT`. The former applies the flag commands to a flag table attached to the input uv data, the latter causes the flag commands to disappear without a trace. After `EXIT`, of course, one may use, edit, or ignore the output flag (`FG`) table. For single-source files, it may be necessary to run `SPLIT` to apply the `FG` table to the data since only some tasks know how to apply `FG` tables (those with `FLAGVER` as an adverb).

The colors used by `EDITR` are those of the various graphics planes when it begins to run. You may change them with the `AIPS` verb `GWRITE` to more desirable colors. The planes are:

```
Plane      Default RGB       Use
  1      1.00  1.00  0.00   Main editing and secondary windows
  2      0.06  1.00  0.00   Comparison baseline data windows
  3      1.00  0.67  1.00   Menu highlight
  4      0.00  1.00  1.00   Edit and frame window boundaries
```

```
  5     1.00  0.18  0.18   Flagged data in all windows
  6     0.60  0.60  1.00   Menu foreground
  7     1.00  0.80  0.40   Second uv data set if present
```

You may wish to change the colors to ones that you can see better.

5.6 Imaging with OBIT, Complex imaging

`AIPS` contains "verbs" that access programs in the `OBIT` software package written by Bill Cotton (`bcotton@nrao.edu`). If that package is installed on your computer, which is easy for Linux and not too hard for Mac systems, then you may use these programs. Two of the verbs are used to translate EVLA data into *AIPS* format; see § 4.1.1. The other four verbs provide increasingly elaborate access to `OBIT`'s `imager` program. They are called `OBITMAP`, `OBITIMAG`, `OBITSCAL`, and `OBITPEEL`. The first images a field of view in a single scale, the second allows many more imaging options including multi-scale, the third adds self calibration, and the fourth allows the self-calibration to be direction dependent. The *AIPS* group cannot take responsibility for the `OBIT` installation and programs, but much of the cleverness in *AIPS* was due to Bill and his later work deserves consideration. If nothing else, the tasks will run in a multi-threaded fashion which *AIPS* does not.

AIPS has a complex imaging option impletnted with the `RUN` file and procedure called `CXIMAGR`. It is a sequence of tasks `UVIMG`, `FFT`, and `CXCLN`. It is interesting, but the normal imaging described in this chapter is likely to be more satisfying.

5.7 Additional recipes

5.7.1 Cranberry banana bread

1. In a large saucepan, bring 2 cups **sugar** and 1 cup **water** to a boil, stirring to dissolve the sugar. Add 4 cups fresh **cranberries** and simmer over low heat for 10 minutes or until berries pop open. Cool. Drain the berries, reserving the juice and measuring 1 cup of berries for use in the bread.
2. Sift together 1 3/4 cup **flour**, 1/2 teaspoon **salt**, 2 teaspoon **baking powder** and 1/4 teaspoon **baking soda**.
3. In a large bowl, combine 2/3 cup **sugar**, 1/3 cup melted **butter**, 2 beaten **eggs**, 1/2 cup chopped **walnuts**, 1 cup mashed **banana**, and 1 cup cooked berries.
4. Add the flour mixture to the berry mixture, stirring until blended. Pour the mixture into a greased and lightly floured 9 x 5 x 3-inch loaf pan. Bake in a preheated, 350° F oven for 1 hour or until a toothpick inserted in the center comes out clean.
5. For a topping (optional), combine 1/4 cup **cranberry juice** from cooked berries, 2 tablespoons **sugar** and 2 tablespoons **Grand Marnier** in a small saucepan and stir over low heat until heated through. Poke a few holes in the baked loaf and pour on the topping.
6. Cool 10 minutes in the pan. Turn the loaf out on a rack and cool completely. Wrap in foil and store one day before slicing.

Thanks to Tim D. Culey, Baton Rouge, La. (`tsculey@bigfoot.com`).