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 (§ 4.4.3) for removing data at this stage in the processing. This chapter has been revised for the 15APR98 release of $\mathcal{AIPS}$ and significant portions of it do not apply to previous releases. In particular, 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. Tasks MX, HORUS, *et al.*, which are now obsolete, are no longer described.

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 § 4.4 and § 8.1, and imaging and self-calibration are also discussed in § 8.4 for spectral-line data and in § 9.6 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 tapes or disk files. FITS is the internationally recognized standard for moving astronomical data between different types of computers and different software packages. Pre-1990 VLA data may also be stored on EXPORT format tapes. This format was written by the now-deceased VLA DEC-10 and as an option by old versions of $\mathcal{AIPS}$.

5.1.1 Indexing the data — PRTTP

Bring your data tape to the $\mathcal{AIPS}$ processor and follow the tape mounting instructions in § 3.9. The program PRTTP reads a full tape and prints out a summary of all the *uv* and image data on tapes written in any of the supported formats. Type:

> TASK 'PRTTP' ; INP C_R	to list the required inputs on the terminal.
> INTAPE *m* C_R	to specify the tape drive number (*m*).
> NFILES 0 C_R	to print information for all the files.

> DOCRT -1 C_R	to print output on your system printer.
> PRTLEV 0 C_R	to select the level of reporting.
> GO C_R	to run the program.

The tape will be rewound if necessary and will then begin to move forward. All files will be read. A printout will appear on the system printer with a valuable summary of header information for each file on the data tape. The printout may be routed, instead, to your terminal by specifying `DOCRT 1` C_R before running `PRTTP` or it may be saved in a disk file by setting `OUTPRINT` (see § 3.10.1).

As `PRTTP` starts executing, look for the message `PRTTP BEGINS` on the *AIPS* "monitor" (the `MSG_SERVER` window on your workstation or, in its absence, your own *AIPS* window or some nearby terminal on antique systems). If you can see the active tape drive from your terminal, look also for movement of the tape. The `AIPS` prompt > should have already returned on your terminal, however, since `PRTTP` is running as a detached "task." As described in Chapter 3, tasks are the more complicated *AIPS* programs, run by the `GO` command after setting the task name with TASK '*taskname*'. They are shed from the terminal, as `PRTTP` has been here, allowing you to use `AIPS` for further processing (except running the same task at the same time).

The file at which the tape is currently positioned can also be "indexed" by the `AIPS` verb `TPHEAD`. This verb, which also works on FITS-disk files (§ 3.10.3), displays the data header to let you decide if you are pointing at the desired data file.

5.1.2 Loading the data — `FITLD` and `UVLOD`

`FITLD` copies FITS-format images and *uv* data from tape (or from an external FITS-format disk file) into your *AIPS* catalog on disk. The following shows inputs to `FITLD` for reading data from the third and fourth files on a tape mounted on tape unit number 2:

> TASK 'FITLD' ; INP C_R	to set the task name and review the required inputs.
> INTAPE 2 C_R	to specify the tape drive number; the tape must already be mounted as in § 3.9.
> NFILES 2 C_R	to skip to the third file on the tape.
> CLRONAME C_R	to use the file names on the tape.
> OUTDISK 3 C_R	to specify writing to disk 3, *e.g.*, to select a disk with sufficient free space. (See § 3.6 for help in monitoring free disk space).
> DOUVC 1 C_R	to use compressed *uv* disk format to save space.
> NCOUNT 2 C_R	to read 2 consecutive tape files.
> OPTYPE 'UV' C_R	to restrict reading to *uv* files.
> REWIND C_R	to rewind tape before skipping files.
> GO C_R	to run `FITLD`..

The tape will begin to move and appropriate messages should appear on the *AIPS* monitor. When the prompt > appears on your terminal, you are free to use `AIPS` for other purposes.

`FITLD` may also be used to read a FITS-disk file as:

> INFILE 'FITS:*filename* C_R	to read the FITS-disk file in the public area known as `$FITS` of name *filename*.
> GO C_R	to run `FITLD` with the adverbs set above — `NFILES` and `NCOUNT` are ignored when `INFILE` is not blank.

See § 3.10.3 for a discussion of FITS disk files.

If your data are in the old EXPORT format, you must use `UVLOD` instead. This task is restricted to *uv* files, but can read both FITS and EXPORT formats. Since the latter may have multiple sources, frequencies, and the like in each file, `UVLOD` has extra adverbs to let you specify source name, frequency band, source qualifier number, and, if all others fail, position within the file. See HELP UVLOD CR 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 CR

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

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

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

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

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

5.1.3 Sorting the data — UVSRT

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

> TASK 'UVSRT' ; INP 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`.

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

5.2 Basic image making — IMAGR

AIPS has several imaging tasks, each with distinctive capabilities. The older tasks UVMAP, MX, WFCLN, and HORUS will not be described here since it is our belief that they have all been superceded by IMAGR. See their help files if you wish to use them. The abilities of IMAGR include:

1. data calibration application for multi-source or self-calibrated single-source data sets.
2. data sorting if needed to fit the weighting, gridding, or Cleaning.
3. data weighting options far more general than those in any other task and including all those used in previous tasks.
4. data imaging in up to 64 simultaneous fields, each up to 8096x8096 in size.
5. Cleaning of all fields simultaneously with subtraction of the Clean components from the data at each major cycle followed by re-computation of the residual images — avoiding aliasing of sidelobes and allowing components almost to the edges of each field.
6. 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.
7. correction of Clean components for various wide-field and wide-bandwidth effects.
8. truly interactive TV display of residual images allowing you to alter the areas over which Clean components are sought.

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

We do not discuss imaging theory and strategy in much detail here because it is discussed fully in numerous lectures in *Synthesis Imaging in Radio Astronomy*[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 typical use of IMAGR at this stage is to construct an unpolarized (Stokes I) image at low resolution and wide field to search for regions of emission or at full resolution for deconvolution by image-plane techniques discussed in § 5.3.6. The following example assumes the use of an already calibrated, single-source data set:

[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)

> RESTORE 0 C_R	to set all adverbs to null values.
> TASK 'IMAGR'; INP C_R	to set the task name and see what to specify.
> INDI *m*; GETN *n* C_R	to select the desired *uv* database.
> STOKES 'I' C_R	to construct a total intensity image — use 'Q', 'U', 'L', 'R', etc. for other Stokes parameters one at a time.
> DOCALIB FALSE C_R	to apply no calibration.
> IMSIZE 1024 C_R	to make a square image 1024 pixels on each side.
> CELLSIZ 1 C_R	for 1 arc-second cells.
> UVTAP *xtap* *ytap* C_R	to specify the widths to 30% of the Gaussian taper in x and y in kλ (kilo-wavelengths).

Other inputs are defaulted sensibly, which is why we started with a `RESTORE 0`. In particular, Clean is turned off with `NITER = 0`, other calibrations are turned off, and all of the data (all IFs, channels, sub-arrays) will be used. Data weighting will be somewhere between pure "uniform" and pure "natural" (see § 5.2.3). Consider both:

> DOCRT = -1 ; EXPLAIN IMAGR 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:

> GO C_R to run `IMAGR`.

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

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

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

> INDI 0 ; MCAT 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    .IMAP    .    1 MA 22-MAR-1995 13:50:10
 43   76 3C138 A C    .IBEM    .    1 MA 22-MAR-1995 13:59:58
```

Note that the default beam class is `.IBEM`; the default image class will be `.IMAP`. The images produced with `NITER = 0` by `IMAGR` can be deconvolved by various image-plane methods (§ 5.3.6).

5.2.2 Imaging multiple fields and image coordinates

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

You specify the multiple-field information with:

> NFIELD n C_R	to make images of n fields.
> IMSIZE i, j C_R	to set the *minimum* image size in x and y to i and j, where i and j must be integer powers of two from 64 to 8192.
> FLDSIZ $i_1, j_1, i_2, j_2, i_3, j_3, \ldots$ C_R	to set the area of interest in x and y for each field in turn. Each i_n and j_n is rounded up to the greater of the next power of 2 and the corresponding IMSIZE. FLDSIZE controls the actual size of each image and sets an initial guess for the area over which Clean searches for components. (That area is then modified by the various box options discussed later.)
> RASHIFT $x_1, x_2, x_3, \ldots$ C_R	to specify the x shift of each field center from the tangent point; $x_n > 0$ shifts the field center to the East (left).
> DECSHIFT $y_1, y_2, y_3, \ldots$ C_R	to specify the y shift of each field center from the tangent point; $y_n > 0$ shifts the field center to the North (up).
> DO3DIMAG TRUE C_R	to specify that the (u, v, w) coordinates are re-projected to the center of each field.

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.

The OUTCLASS of the fields is controlled by IMAGR with no user assistance. For dirty images it is IMAP for the first field, IMAP01 for the second, and so forth. The I is replaced by Q, U, *et al.* for polarized images and the MAP is replaced by CLN when Cleaning. When DO3DIMAG is false, one beam of class IBEM is used for all fields. When DO3DIMAG is true, a beam of class IBEM*nn* is used for fields of class IMAP*nn*or ICLN*nn*, 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). The coordinate system for VLA images is the SIN projection, for which the image coordinates x and y relate to right ascension α and declination δ as

$$x = \cos\delta \sin\Delta\alpha$$

$$y = \cos\delta_0 \sin\delta - \sin\delta_0 cos\delta \cos\Delta\alpha$$

where $\Delta\alpha = \alpha - \alpha_0$ and the coordinates with subscript "0" are those of the tangent point that serves as the origin of the image coordinate system. 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). In either case, the RASHIFT and DECSHIFT are specified with respect to the reference coordinate of the input *uv* data set.

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

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 $W \propto 1/\sigma^2$. The 15APR98 version of FILLM offers the (default) option, for recent VLA data, of weighting data in this fashion using recorded system temperatures. Prior to this improvement, weights were simply based on integration times and assumed that each antenna in the array had the same system temperature. If the weights are not of this form, run 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 would contribute to a natural weighted image in this case.

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

> UVSIZE 0 ; UVWTFN ' ' 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 overline 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. `IMAGR` is new and the impact of all of these parameters on imaging is not well understood. You may wish to experiment since it *is* known — see figures 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.

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

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

5.2.4 Cell and image size, shifting

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

Actually, this is not the full story. If you have a large number of samples toward the outer portions of the *uv*-data grid, then the width of the main lobe of the dirty beam will not be correctly measured. Making the cell size smaller — raising the size of the *uv*-data grid (in wavelengths) — will change the apparent

Figure(s) omitted to shorten PostScript file.

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).

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

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

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

5.2.5 Zero-spacing issues

You help Clean to guess what may have happened in the unsampled "hole" at the center of the *uv* plane by including a zero-spacing (usually single-dish) flux density when you make the image. This gives Clean a datum to "aim at" in the center of the *uv* plane. Extended structure can often be reconstructed by deep Cleaning when the zero-spacing flux density is between 100% and ~125% of the average visibility amplitude at the shortest spacings (run `UVPLT` to estimate this average for your data set). If your data do not meet this criterion, there may be no reliable way for you to image the extended structure of your source without adding further information to your observations (*e.g.*, by adding *uv* data from a more compact array, by Fourier transforming a suitably tapered and deconvolved single dish image of the VLA primary beam, or by using such an image as the default image for a maximum entropy deconvolution as in § 5.3.6). 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 Cotton and Schwab. These enhancements involve going back to the original *uv* data at each "major cycle" to subtract the current Clean-component model and re-make the images. This allows for more accurate subtraction of the components, for Cleaning simultaneously multiple (perhaps widely spaced) smaller images of portions of the field of view, for Cleaning of nearly the full image area, for more accurate removal of sidelobes, and for corrections for various wide-field and wide-bandwidth effects. Of course, all these extras do come at a price. For large data sets with fairly simple imaging requirements, image-based Cleans, particularly `APCLN`, may be significantly faster.

The next section describes the basic parameters of Cleaning with `IMAGR`. The second section describes the use and limitations of multiple fields in `IMAGR`; the third section describes the setting of Clean "boxes" and the TV option in `IMAGR`; and the fourth section describes various wide-field and wide-bandwidth correction options. Clean component files are tables which can be manipulated, edited, and plotted both by general-purpose table tasks and by tasks designed especially for `CC` files. Some aspects of this are discussed in the fifth section. Images may also be deconvolved by other methods in *AIPS*. § 5.3.6 mentions several of these and describes the most popular alternatives, image-based Clean with `APCLN` and `SDCLN` and a Maximum Entropy method embodied in the task `VTESS`.

5.3.1 Basic Cleaning with IMAGR

IMAGR implements a Clean deconvolution of the type devised by Barry Clark and enhanced by Bill Cotton and Fred Schwab. Clean components — point sources at the centers of cells — are found during "minor" iteration cycles by Cleaning the brightest parts of the residual image with a "beam patch" of limited size. More precise Cleaning is achieved at the ends of "major" iteration cycles when the Fourier transform of the Clean components is computed, subtracted from the visibility data, and a new residual dirty image computed. The rule for deciding when a major iteration should end in order to achieve a desired accuracy is complicated (see the Clark paper). IMAGR lets you vary the major iteration rule somewhat to suit the requirements of your image. Type DOCRT FALSE; EXPLAIN IMAGR 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.
> INP C_R	to review the inputs — read carefully.
> GO C_R	to start IMAGR.

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

Two other subtle parameters which help to control the Clean may need to be changed from their defaults. MINPATCH controls the minimum radius in the dirty beam (in pixels) used during the minor cycles to subtract sidelobes of one component from other nearby pixels. If your dirty beam is complicated, with significant near-in sidelobes and your source extended, then the default 51 cells may be too small. IMAGR uses a larger patch during the first few major cycles, but will be reduced eventually to a MINPATCH patch. IMAGR normally creates a dirty beam twice the size of the largest field (or 2048 pixels whichever is smaller). This allows for a very large beam patch in the early cycles, letting widely spaced bright spots be Cleaned more accurately. If your image does not have widely spaced bright spots, you can save some compute time by reducing this beam size with CPARM(10); see the help file. MAXPIXEL controls the maximum number of image pixels searched for components during any major cycle. If MAXPIXEL were very large, IMAGR would spend all of its time examining and subtracting from pixels it is never going to use for components. If it is too small, however, then pixels that should be used during a major cycle will not be used and major cycles may end up using only a few components before doing another (expensive) component subtraction and re-imaging. Again, we do not know what to recommend in detail. The default (20050) seems good for normal 1024x1024 images, smaller values are better for smaller images of compact objects, and rather larger values may be good for extended objects. If the first Clean component of a major cycle is significantly larger than the last component of the previous cycle (and the messages let you tell this), then too few cells are being used.

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

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

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

Note that the number of Clean iterations, and many of the other Cleaning parameters, may be changed interactively while IMAGR is running by use of the *AIPS* SHOW and TELL utilities. Type SHOW IMAGR C_R while the task is running to see what parameters can be reset, and their current values. Then reset the parameters as appropriate and TELL IMAGR C_R to change its parameters as it is running. (The changes are written to a disk file that IMAGR checks at appropriate stages of execution, so they may not be passed on to the program immediately — watch your *AIPS* monitor for an acknowledgment that the changes have been received, perhaps some minutes later if the iteration cycles are long or your machine is heavily loaded. AIPS verb STQUEUE will show all queued 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 may be restarted to continue a Clean begun in a previous execution. To do this, you must set the OUTSEQ to the sequence number of file you are restarting. A good way to do this is

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

The other parameter that must be set to restart a Clean is BCOMP, the number of Clean components to take from the previous Cleans. A restart saves you much of the time it took IMAGR to do the previous Clean, although it will make a new file of residual visibilities. An image can be re-convolved by setting NITER = the sum of the 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. IMAGR writes over the Clean image file(s) as it proceeds to Clean deeper. You can preserve intermediate Clean images, however, either by copying them to another disk file with SUBIM or by writing them to tape with FITTP.

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

5.3.2 Multiple fields in IMAGR

IMAGR can also deconvolve components from up to 64 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 sources from the primary fields. To take advantage of this option, you must have prior knowledge of the location and size of the regions of emission that are important — yet another reason to make a low resolution image of your data first.

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

> NFIELD n C_R	to make images of n fields.
> IMSIZE i, j C_R	to set the minimum image size in x and y to i and j, where i and j must be integer powers of two up to 8192.
> FLDSIZ $i_1, j_1, i_2, j_2, i_3, j_3, \ldots$ C_R	to set the area of interest in x and y for each field in turn. Each i_n and j_n is rounded up to the greater of the next power of 2 and the corresponding IMSIZE. FLDSIZE controls the actual size of each image and sets an initial guess for the area over which Clean searches for components. (That area is then modified by the various box options discussed in § 5.3.3.)
> RASHIFT $x_1, x_2, x_3, \ldots$ C_R	to specify the x shift of each field center from the tangent point; $x_n > 0$ shifts the map center to the East (left).
> DECSHIFT $y_1, y_2, y_3, \ldots$ C_R	to specify the y shift of each field center from the tangent point; $y_n > 0$ shifts the map center to the North (up).

If ROTATE is not zero, the shifts are actually with respect to the rotated coordinates, not right ascension and declination. The actual x shift will be roughly RASHIFT divided by $\cos(\delta)$.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.

The manner in which the multi-field Clean is conducted requires some discussion. When DO3DIMAG is false, there is a single dirty beam for all fields. 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 DO3DIMAG is true, 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 — experimental — arrangement, selected by specifying `OVERLAP = 2` *and* `DO3DIMAG = TRUE`, 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, 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.

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 experimental sequence may reduce this effect somewhat.) At the end of the major cycle, all components from all fields are subtracted from the *uv* data. At this point, all sidelobes of the components are gone from all fields, but the erroneously chosen "objects" with their sidelobes will appear (in negative usually). This is normally not a problem. During the next cycle, Clean will put components of the opposite sign on the erroneous spots and they will eventually be corrected. Nonetheless, it is a good idea to restrict the Cleaning to the obvious sources to begin with, saving Clean the trouble of having to correct itself, and to open up the search areas later in the Clean. The TV options make this easy to do in `IMAGR`; see § 5.3.3.

The situation is more complex if the multiple fields overlap. If a sidelobe in the overlap area is taken as a source in one major cycle, it will appear as a negative source in both fields at the start of the next major cycle. Clean will then find negative components in both fields and correct its original error twice, producing a positive "source" at the next major cycle. Such errors never get fully corrected. A simple rule of thumb is never to allow the search areas of one field to overlap with the search areas of another field. Even then, there is one other "gotcha." In the restore step, Clean only restores components to the fields in which they were found. Thus, a real source visible in two fields will be found in only one after Clean; your two images of the same celestial coordinate will be in substantial disagreement. Therefore, you must be careful about which parts of which images you believe to represent the sky. The `15APR98` version of `IMAGR` now offers the `OVERLAP` option to solve these problems. If `OVERLAP` ≥ 1, Clean components from all fields are restored to all fields as needed. If `OVERLAP` ≥ 2 *and* `DO3DIMAG` > 0, the Clean and imaging are done in a fashion which greatly reduces the instabilities arising from Cleaning the same source (or sidelobe) from more than one field.

5.3.3 Clean boxes and the TV in IMAGR

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

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

> CLBOX $lx1, by1, rx1, ty1, lx2, by2, rx2, ty2, \ldots$ C_R

to specify the pixel coordinates of the Clean windows as leftmost x, bottommost y, rightmost x, topmost y for boxes 1 through NBOXES. Circular boxes may also be specified as -1, radius, center x, center y interspersed in any order with the rectangular boxes. Default is given by FLDSIZE(1).

> BOXFILE '*area* : *filename*' C_R — to specify the name of a text file listing the Cleaning windows. Blank means no file.

The BOXFILE text file is an optional means by which Clean windows may be entered at the start of a run of IMAGR for all fields, not just the first. It is also the only way to enter more than 50 boxes for the first field; the limit is 256 boxes per field with this option. The format of the file is one box per line beginning with the field number followed by the four numbers describing the box as in CLBOX above. Any line in which the *first* character is not a number is taken as a comment or a field definition (see § 5.2.2), *i.e.*, the field number must be left-justified in the line (leading zeros, not blanks, are allowed). NBOXES and CLBOX are overridden if any boxes for the first field are given in the file.

You can use the TV cursor in advance of running IMAGR to set the Cleaning boxes. First, load the TV display with either the dirty image or a previous version of the Clean image of the first field; see § 6.4.1. Then type:

> TVBOX C_R — to begin an interactive, graphical setting of up to 500 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.

You can also use the TV cursor in a very similar way to build and modify the BOXFILE text file. (You can also use your favorite text editor of course; see § 3.10.1 for general information about specifying and using external text files.) The verb FILEBOX reads the text file (if any) given by BOXFILE selecting those boxes (if any) already specified for the specified field number which fit fully on the current image on the TV. Which field number you want is given with the NFIELD adverb, or, if that is zero, deduced from the Class name of the image on the TV. (Be careful to load the TV with the desired image before running FILEBOX!) You then carry out a graphical setting or resetting of boxes in exactly the same manner as with REBOX. The new and changed boxes are then added to the end of the text file. Different portions of the current field and other fields may be done and redone as often as needed.

The real power of IMAGR becomes apparent if you set DOTV = n, where NFIELD $\geq n > 0$ is the field number first displayed on the TV. Before each major cycle, the current residual image is displayed on the TV and a menu of options is offered to you. (Note that the residual image before the first major cycle is the un-Cleaned dirty image.) The image displayed is interpolated up or decremented down (by taking every n^{th} pixel in each direction) to make it fit on the display and the current Clean boxes are shown. If you do not select a menu option, IMAGR proceeds after 30 seconds.

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

OFFZOOM — to turn off any zoom magnification

OFFTRANS — to turn off any black & white enhancement

OFFCOLOR — to turn off any pseudo-coloring

TVFIDDLE	to interactively zoom and enhance the display in black & and white or pseudo-color contours as in **AIPS**
TVTRAN	to enhance in black & white as in **AIPS**
TVPSEUDO	to select many pseudo-colorings as in **AIPS**
TVFLAME	to enhance with flame-like pseudo-colorings as in **AIPS**
TVZOOM	to set the zoom interactively as in **AIPS**
CURVALUE	to display the pixel value and x, y pixel coordinates at the TV cursor position as in **AIPS**
SET WINDOW	to select a sub-image of the whole to be reloaded with better resolution — all boxes must be included.
RESET WINDOW	to select the full image and reload the display
TVBOX	to set the Clean boxes for this field beginning at the beginning as in **AIPS**
REBOX	to reset the current Clean boxes and create more as in **AIPS**
CONTINUE CLEAN	to resume Cleaning now rather than wait for the time out period.
STOP CLEANING	to stop the Clean at this point, restore the components to the residual images, write them on disk, and exit.
TURN OFF DOTV	to resume the Cleaning now and stop using the TV to display the residual images. To turn the TV display back on, if needed, use the `TELL IMAGR` verb with suitable adverbs, including `DOTV TRUE`.
ABORT TASK	to stop the task abruptly, destroying the output images and exiting as quickly as possible.

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

SELECT FIELD n	to display field n, allowing its Clean boxes to be altered.

Thus you can look interactively at the initial dirty images, place boxes around the brightest sources, and start the Clean. As it proceeds and weaker source become visible, you can expand the boxes and add more to include other sources of emission. Do be careful, however. Boxes that are too tight around a source can affect its apparent structure. The author once made Cas A into a square when stuck with a too-tight box. 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.
DISPLAY ONLY n	to indicate that only field n is displayed and available to have its boxes altered.

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

5.3.4 Data correction options in `IMAGR`

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

5.3.4.1 Frequency-dependent primary-beam corrections

The primary-beam pattern of the individual telescopes in the interferometer scales with frequency. Therefore, each channel of multi-frequency observations of objects well away from the pointing center effectively observes a different sky. When a combined source model is produced, there will be residuals in the visibility data that cannot be Cleaned as the data does not correspond to a possible sky brightness distribution. If `CPARM(1)` is larger than 0, then a correction is made in the subtraction of Clean components from the *uv* data to remove the effects of the frequency dependence of the primary beam. The primary beam is assumed to be that of a uniformly illuminated disk of diameter `CPARM(1)` meters. This correction is made out to the 5% power point of the beam with a flat correction further out. Note: this correction is only for the relative primary beam to correct to a common frequency and *does not* correct for the primary beam pattern at this frequency.

5.3.4.2 Frequency-dependent correction for average spectral index

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

5.3.4.3 Error in the assumed central frequency

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

5.3.4.4 Array mis-orientation effects

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

5.3.4.5 Non-coplanar effects

IMAGR has a CPARM(4) option to attempt to correct for non-coplanar effects in imaging. If this worked, it would be very very slow. At this writing, it is not believed to work at all 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.

5.3.4.6 Units mismatch of residuals and Clean components

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

5.3.5 Manipulating Clean components

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

> TASK 'PRTCC' ; INP	to select the task and review its inputs.
> INDI *n* ; GETN *ctn* C_R	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^{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 from the data, producing a residual visibility data set. Of course, IMAGR's workfile already contains these residuals with the CC files of all fields subtracted. Various display options can be used on these *uv* files; see § 5.3.1. VPLOT, described in § 9.3.3, will plot a CC model against visibility data, one baseline at a time, *n* baselines per page.

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

> TASK 'CCMRG' ; INP	to select the task and review its inputs.
> INDI *n* ; GETN *ctn* C_R	to select the Clean image, where *n* and *ctn* select its disk and catalog numbers.
> INVERS *m* ; OUTVER *m* C_R	to select the input version of the Clean components and to replace it with the compressed version.
> GO C_R	to execute the task.

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.

5.3.6 Image-plane deconvolution methods — APCLN, SDCLN, VTESS

The previous sections have described the new task IMAGR which implements Clean by subtracting model components in groups from the ungridded *uv* data and re-imaging. This can be rather expensive. If you have a significant number of visibilities contributing to a fairly small image, it may be faster to use an image-plane deconvolution method. The venerable APCLN implements the Clark Clean in the image plane. Clean components are found during "minor" iteration cycles by Cleaning the brightest parts of the residual image with a "beam patch" of limited size, just as in IMAGR. More precise Cleaning is achieved at the ends of "major" iteration cycles when the Fourier transform of the Clean components is computed, multiplied by the transform of the beam, transformed back to the image plane, and then subtracted from the dirty image. This method does a good job Cleaning the inner quarter of the image area, but artifacts of the Cleaning and aliasing of sidelobes do interesting things to the remaining 75% of the image. Make the dirty image using

IMAGR and be sure to make it large enough to include all of the source in the inner quarter of the area. APCLN uses many of the now-familiar adverbs of IMAGR, including GAIN, FLUX, NBOXES, CLBOX, FACTOR, MINPATCH, MAXPIXEL, BMAJ, and more. APCLN recognizes only rectangular boxes and its DOTV option only displays the residual image with a pause for you to hit button D to end the Cleaning early.

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

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

The most widely used, best understood, and probably most successful alternative to Clean is the Maximum Entropy Method ("MEM"). This is implemented in *AIPS* by the task VTESS. This requires a dirty image and beam, such as those produced by IMAGR with NITER set to 0, each twice the (linear) size of the region of interest (as for APCLN and SDCLN). The deconvolution produces an all-positive image whose range of pixel values is as compressed as the data allow. The final VTESS image is therefore stabilized against Clean-like instabilities while providing some "super-resolution" wherever the signal-to-noise ratio is high. VTESS can also deconvolve multiple images simultaneously; see below.

There are three main reasons to prefer MEM deconvolution over all of the Clean deconvolution methods:

1. MEM can be much faster for images which have strong signals in many pixels. "Many" seems to be $\geq 512^2$ or so.
2. MEM produces smoother reconstructions of extended emission than does Clean.
3. MEM allows introduction of *a priori* information about the source in the form of a "default" image.

Because VTESS can produce excellent deconvolutions of extended sources in much less computation time than Clean, but requires careful control, we recommend studying the output of EXPLAIN VTESS C_R before using the task. Chapters 8 and 15 of the NRAO Summer School on *Synthesis Imaging in Radio Astronomy* also provide useful general background.

MEM can be used for quantitative work on regions of good (> 10) signal-to-noise ratio, if the dirty image is convolved with a Clean beam prior to deconvolution. Use the *AIPS* task CONVL for this purpose. The images may also be post-convolved, and added to the residuals, within VTESS. In many cases, the images of extended sources produced by SDCLN and VTESS are functionally identical. VTESS usually converges in *much less* CPU time, however, at the expense of leaving significantly larger residual sidelobes close to bright compact (point-like) features. To get around this deficiency of VTESS, first use Clean to remove the peaks of bright point-like features, then run VTESS on the residual image produced by this restricted Clean. (The *AIPS* Clean tasks will output a residual image if you set BMAJ < 0.)

VTESS can also combine information from different types of data. For example, single-dish data can be used to constrain the imaging of interferometer data, or many pointings covering one large object can be processed together. VTESS takes up to 64 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 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

If you decide to use self-calibration, a good sequence of steps is:

1. Use UVPLT to make a plot file showing the shape of the visibility function as a function of baseline length in the externally-calibrated data set. (See § 6.3, especially § 6.3.1, for information about plotting in AIPS.) *N.B.,* for large data sets, use XINC to reduce the number of points plotted to no more than a few thousand; otherwise it will take too long to make and plot the plot file. Use LWPLA to get hard copy of the plot file.

2. If you can use a point-source model for the first iteration, *i.e.*, if a range of baselines sufficient to calibrate all antennas is dominated by a single component (flat visibility function well above the noise), go to step 6 directly. This is frequently done with VLBI data, but is less common with arrays for which the initial calibrations are better such as the VLA.

3. If you must use a more complicated model, obtain a Clean-component representation of it by making and Cleaning an image of the externally-calibrated data using IMAGR. Leave the *uv* data in "TB" sort

order for CALIB; IMAGR will sort them if it has to. Note that you may want to use a somewhat higher loop GAIN in a Clean to be used as an input model for an early iteration of self-calibration than you would for final deconvolution of a very extended structure.

4. Consider running CCMRG to reduce the number of components in the model. This improves the speed of the calibration and makes the first negative component be a real negative rather than a minor correction to previous positive components. Remember that merging the components does alter the model which is used to compute the gains unless you were going to include all components anyway.

5. Use PRTCC or TAPLT (as in the example in § 5.3.5) to help you decide how many components from this Clean to include in the CALIB model. CCFND is also helpful. When you have decided this, determine the appropriate *uv*-limits for the gain solution by referring to the hard copy of the visibility function you made at step 1.

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

7. Use CALIB to calculate the gain corrections. It will apply them to produce a new, (hopefully) improved data set, and will also catalog the gain corrections as an SN extension to the *input uv* data file.

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

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

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

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

There is a new task called SCMAP which attempts to implement this sequence inside a single task. SCMAP contains all of IMAGR (for a single field) and all of CALIB. It attempts to make the decision about the number of merged components and the range of *uv* spacings to use in each self-calibration based on σ times the rms in the residual image of the current Clean, where you provide the σ. The process is somewhat less flexible, but also less painful, than running CALIB and IMAGR multiple times. SCMAP does not let you change imaging parameters while it is running, but it does provide interactive methods to change Clean boxes and to set a variety of Cleaning and self-calibration parameters including loop gain, solution interval and solution smoothing interval. It lets you switch from phase-only to amplitude and phase self-calibration or it will do it automatically when the phase only stops converging. SCMAP offers 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 and allows for multiple fields and DO3DIMAG, 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`.

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.
> 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 four characters of `IN2CLASS` with the field number minus one given in the last two characters as is done by `IMAGR`.
> NCOMP = $n_1, n_2, \ldots$ C_R	to cut off the model at the $n_i^{\rm th}$ Clean component in the $i^{\rm th}$ image.
> INVERS m C_R	to specify the `CC` file version number to use from *every* model image; 0 means the highest.
> SMODEL S, x, y, m C_R	to specify a point-source (or Gaussian or uniform spherical) model rather than a Clean component model. `CALIB` uses a source model (type m) of S Jy located at x, y arc-sec with respect to the pointing center. For a point model m `= 0`; see the help for details of the other types.
> SUBARRAY s C_R	to select the appropriate sub-array — `SUBARRAY = 0` implies all sub-arrays.
> UVRANGE = x_1, x_2 C_R	to give full weight (in doing the gain solutions) only to data from projected baselines between x_1 and x_2 in kilo wavelengths.
> WTUV w C_R	to set the weight for projected baselines outside the range `UVRANGE(1)` $\rightarrow$ `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.
> CPARM(2) = 1 C_R	to scale the gain corrections by the mean modulus of all gains to keep the flux density scale from drifting; ≤ 0 lets the gains float free.
> 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, osummaries 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%.
> 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.

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!

5.4.3 Considerations in setting `CALIB` inputs

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

It pays to be conservative when using `NCOMP` to select the number of Clean components which will comprise the input source model. Setting `NCOMP` too high will fossilize errors from the earlier calibrations in the model for the next one; after this, you are stuck with them as long as you continue feeding `CALIB` a model with as much Cleaned flux density. When calibrating Stokes I images, do not set `NCOMP` in `CALIB` so high that any negative Clean components are included. The first few iterations of `CALIB` should be phase-only calibration, since the tropospheric and ionospheric phase errors will almost always dominate amplitude errors due to the atmosphere or to system drifts. In these first iterations, it is prudent to be even more conservative, setting `NCOMP` so that the total Cleaned flux included in the model is between 50% and 80% of that at which the first negative Clean component appeared. `CCFND` will help you with this (§ 5.3.5). If your field is dominated by a few very strong, small-diameter regions, it is a good idea to make the first iterations of `CALIB` work on Clean components from these regions alone, restricting the range of baselines suitably by setting `UVRANGE(1)`. Setting Clean windows in `IMAGR` or using `CCEDT` (§ 5.3.5) suitably will help you do this. Even later in the self-calibration cycle, it is probably still a good idea to eliminate weak, isolated Clean components. Try `CCSEL` for this.

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

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

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

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

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

1. there are too few antennas (`APARM(1)`) to form a solution,
2. the solution does not converge, or

3.the signal-to-noise ratio for a given antenna (APARM(7)) is too low.

The signal-to-noise ratio is calculated from the post-fit scatter of the residuals from the gain model. Note that the scatter will contain contributions from thermal noise *and* unmodeled source structure. This is a good reason to restrict the *uv* range of the data. For further guidance and information on other CALIB inputs, type EXPLAIN CALIB C_R and/or read Lectures 9 and 16 in *Synthesis Imaging in Radio Astronomy*.

5.5 More editing of *uv* data

5.5.1 General remarks on, and tools for, editing

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

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

There are facilities in CALIB, CLIP, and CORER to flag *uv* data in *AIPS* based on deviations from specified norms. There is also the task UVFLG to flag and unflag by antenna-IF or by correlator. The task UVPLT plots various combinations of *uv* data; see § 6.3.1. The task UVFND is also recommended for printing out suspicious portions of the 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 are in range. To examine the correlators individually, use STOKES 'CORR' in UVFND, or to flag the data based on their values after conversion to true Stokes use APARM(5) = 1 in CLIP.

CLIP is also useful for flagging discrepant data (*e.g.*, due to interference or malfunctions) on the basis of their deviations from the visibility predicted by a set of Clean components. The task UVSUB will subtract the 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 of course run UVSUB again after doing the flagging or clipping, to add the transform of the Clean components back into the remaining data (using the input FACTOR = −1.0). Note that IMAGR's workfile is also a *uv* 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.

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

TVFLG, SPFLG, 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 § 4.4.3 of the *CookBook*. `SPFLG` is a very useful task for data with a significant number of spectral channels. It is effective in examining data for frequency-dependent errors and interference and can be an effective data editor for interferometers with a small number of baselines; see § 10.2.2 and § 8.1. `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 intevals. 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.

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

`EDITR` is a very effective editing tool from the beginning of data analysis on data sets with modest numbers of antennæ. Since it can display two data sets at the same time for comparison purposes, `EDITR` may also be used to good purpose with larger data sets during the self-calibration and imaging stage. The visibility amplitude or phase or the amplitude of the visibility with a running vector average subtracted may be displayed. The data for the selected baseline are shown in an edit window at the bottom of the display. Optionally, a second observable (*e.g.*, phase) from the selected baseline is shown in the same color in a window directly above the edit window. This option is controlled by the `DOTWO` adverb. Data for 0 to 10 other baselines to the selected antenna may be displayed in a different color in windows abive 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 and to help squeeze the data into memory, but will cause the data flags to be less selective in time. The program is more efficient if all data fit in memory, so `EDITR` will try to read all data into memory if it can. Failing that it will try to read all data for one IF or all data for one antenna for all or one IF. It will fail if this last case does not fit and you will need to use `TIMERANGE` or `SOLINT` to reduce the amount of data.

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-source or multi-source data set.
> DOCAL FALSE C_R	to apply no calibration. The `SN` or `CL` table from previous calibrations can be applied.
> 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 $1/60$ = 1 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.

> INP C_R — to review the other parameters, which we assume here to be set to their null values.

> GO C_R — to run the task.

You can average the data over spectral channels (the default will average all channels present). IFs are edited separately; the default will include all IFs after which you can choose the one to edit interactively.

Since the display used by `EDITR` is very similar to the one used by `EDITA` displayed in § 4.4.2, we do not include a figure here; see Figure 4.1. 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 anotations 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

SWITCH POLARIZ — To switch to viewing and editing the other polarization.

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. This can force a read of data if all IFs did not fit in memory.

SWITCH ALL IF — To switch functions from applying to one IF to applying to all IFs or vice versa.

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.

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 rotate 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" *may* also be adjusted.
PLOT ALL TIMES	To display all data for the selected baselines.
SELECT FRAME	To select a window into the current data interactively.
NEXT FRAME	To select the next time range window of the same size as the current frame.
PREVIOUS FRAME	To select the previous time range window of the same size as the current frame.
SHOW AMPLITUDE	To display and edit amplitudes.
SHOW PHASE	To display and edit phases.
SHOW DIFF AMPL	To display and edit the amplitudes of the vector difference between the sample and its running mean.
SHOW ALSO AMPL	To display amplitudes of the edit baseline for reference with the phase or difference amplitude edit window.
SHOW ALSO PHASE	To display phases of the edit baseline for reference with the amplitude or difference amplitude edit window.
SHOW ALSO DAMP	To display difference amplitudes of the edit baseline for reference with the phase or amplitude edit window.
TV ZOOM	To alter the display zoom used while in the flag functions.
OFF ZOOM	To turn off any zooming.
2ND UV OFF	To disable the display of the 2^{nd} *uv* data set.
2ND UV ON	To enable the display of the 2^{nd} *uv* data set.

The data displayed are of a single polarization, single IF, and 1–11 baselines to a single antenna. The `SWITCH POLARIZATION` option switches the displayed polarization, the `ENTER IF` option prompts you 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 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 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. The display the amplitude of the vector difference, select `SHOW DIFF AMPL`. Such displays are particularly sensitive to short-term problems while ignoring longer-term changes due to source structure. Since it takes time to compute things for, and display, the second data set, you may wish to turn it off part of the time. The `2ND UV OFF` and `2ND UV ON` options control this choice.

The left-hand menu can contain

FLAG TIME	To delete one time at a time.
FLAG TIME RANGE	To delete one or more time ranges.
FLAG BELOW	To delete all displayed times with data below a cutoff value.
FLAG ABOVE	To delete all displayed times with data above a cutoff value.
FLAG AREA	To delete one or more areas in the data-value *vs* time plane.
FLAG POINT	To delete one sample at a time using both horizontal and vertical cursor position.
FLAG FAST	To delete samples using only mouse clicks
ENTER AMPL RNG	To select the display range for amplitude plots.
ENTER PHASE RNG	To select the display range for phase plots.
ENTER DAMP RNG	To select the display range for plots of the amplitude of the visibility minus a running vector average visibility.
PLOT ERROR BARS	To plot error bars based on data weights.
SET SCAN LENGTH	To set the averaging time used to determine the running average in seconds.
LIST FLAGS	To list all flags now in the Flag Command table.
UNDO FLAGS	To undo one of the flag operations in the `FC` table
REDO FLAGS	To reapply all remaining flags after one or more have been undone
SET REASON	To set the 24-character "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.
EXIT	To exit `EDITR`, moving the `FC` table to a *uv*-data `FG` table.
ABORT	To exit `EDITR`, deleting the `FC` table.

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

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

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

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

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

```
Plane     Default RGB       Use
  1     1.00  1.00  0.00   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 Additional recipes

5.6.1 Banana-Rhubarb Crisp

1. Slice 2 large `bananas` into 1/4-inch rounds. Combine with $2\frac{1}{2}$ cups diced `rhubarb`, 2 tablespoon `sugar`, 1/4 teaspoon `cinnamon`, and a generous dash `nutmeg`. Spoon the mixture into a well-greased 9-inch pie plate or shallow baking dish (preferably glass or ceramic).
2. In a medium bowl, combine 1/2 cup white or whole-wheat pastry `flour`, 1/2 cup `graham cracker` crumbs, $1\frac{1}{2}$ teaspoons `baking powder`. With a pastry blender or two knives worked in a crisscross fashion, cut in 1/4 cup `butter` until the mixture is crumbly.
3. Combine 1 `egg` lightly beaten with 1/4 cup `milk` and stir into the flour mixture. Spoon the batter as evenly as possible over the fruit mixture. Sprinkle with 2 tablespoons `sugar`.
4. Bake in a pre-heated 400deg oven for 25–30 minutes.

Thanks to *Jane Brody's Good Food Book.*

5.6.2 Hawaiian banana cream pie

1. Preheat oven to 375 deg.
2. In a bowl, combine 1 cup chopped cashew or macadamia `nuts`, 1/2 cup flaked `coconut`, and 2 tablespoons `brown sugar`.
3. Beat 1 `egg white` until stiff; fold into nut mixture.
4. Press mixture evenly into an 8-inch pie plate, building up the sides slightly. Bake for 7 minutes or until crust is lightly browned. Crust will tighten as it cools (use a rack).
5. In a medium-sized saucepan, beat 3 `egg yolks`. Mix in 5 tablespoons `cornstarch` and 3/4 cup granulated `sugar`. Stir in 1.5 cups `milk`, 1/4 teaspoon `salt`, and 1 tablespoon unsalted `butter`.
6. Cook mixture slowly over medium heat, stirring constantly, for 5 to 7 minutes. Filling should be bubbling and thick.
7. Remove from heat and stir in 1 teaspoon `vanilla extract`. Transfer this custard to a glass bowl, cover with plastic wrap, and refrigerate for 2 hours.
8. Two hours before serving, whip 1/2 cup heavy `whipping cream` to stiff peaks and fold into custard. Peel and slice one `banana`, arranging evenly on bottom of crust. Spoon custard filling into crust. Cover again with plastic wrap and chill for 2 more hours.
9. Sprinkle 1/2 cup finely chopped cashew or macadamia `nuts` evenly over the filling. Peel, slice and arrange a second `banana` in a circular fashion around the outside top of the pie, placing a few slices decoratively in the center.