ANALYSIS OF AUSTRALIA TELESCOPE COMPACT ARRAY DATA WITH AIPS

Neil Killeen, ATNF
ATNF manual No. K2
Version 4
November 27 1993

Updates to manual on WWW.

Contents

1 INTRODUCTION

This document describes how to calibrate and image Australia Telescope Compact Array data with *AIPS* at the ATNF laboratory in Epping and at the ATCA observatory in Narrabri.

I have tried to provide two paths through the analysis procedure. First, a detailed exposition which attempts to explain what to do as well as why. Second, since this approach is unlikely to satisfy all users, I have also provided a more skeletal approach through the use of flow-charts and *AIPS* 'adverb boxes'. The latter show *examples* of most of the major *AIPS* tasks that you will encounter and should help illuminate a less detailed path to success. However, some useful information which might help to avoid some pitfalls is necessarily lost with this simple guide. The calibration and imaging procedures are not trivial, and there are too many branches to provide a completely 'black-box' approach for those who want it. I encourage all new users to take the time to learn the salient points of the relevant analysis tecvhniques.

This manual does not try to teach you how to use *AIPS*. You should use the *AIPS* Cookbook for that or ask a colleague for instruction. Other documentation that may be useful to you is the "Guide to Computing at the Radiophysics Laboratory" for general information about the computing facilities available at Epping. Also, the NRAO synthesis workshop books are an excellent reference for information about aperture synthesis techniques. Finally, there is the detailed text book by Thompson, Moran, and Swenson called "Interferometry and Synthesis in Radio Astronomy".

There is no path through *AIPS* for those who wish to make a full polarimetric calibration of ATCA data. Furthermore, even if you are interested only in total intensity, a full polarimetric calibration is required for the best results. To make such a calibration, you must use the alternative radio astronomy reduction package called *MIRIAD*. See § 3.3 for some help on when to use *AIPS* and when to use *MIRIAD*. *MIRIAD* also offers some other capabilities that do not exist in *AIPS*. In particular, a suite of tasks for multi-frequency synthesis work designed specifically for ATCA data. However, *AIPS* performs better in other areas, most notably those of visibility editing and image display and analysis (although the gap is steadily closing). Regardless, it is now quite feasible to analyze your data entirely within *MIRIAD*, whether you are doing spectral line or continuum work. A *MIRIAD* User's Guide for ATCA users is available from Bob Sault or myself. Please come and talk to either of us if you are unsure of what package or combination of packages is most suitable for your needs.

Before you can do any analysis, you must of course first obtain a computer account if this has not already been organized prior to your visit. This account will provide access to all relevant machines (Convex and workstations). Make sure that the member of teh computer group that sets up your account puts you in the *aipsusr* group. You should also have a workstation and disk booking before you visit Epping. These are made via Henrietta May (hmay@atnf.csiro.au) or myself (nkilleen.atnf.csiro.au); ideally you should arrange this booking before you visit the ATNF, otherwise you may find resources are not available.

In addition, before you try to run *AIPS*, you should have an *AIPS* user number. If you have not been allocated one, please contact Henrietta May or myself at Epping. If you already have a favcurite number, please check with us to make sure it is available for use here (we are not necessarily consistent with NRAO's allocation system).

All criticisms and suggestions will be given due consideration and all plagiarism in this document is implicitly (and occasionally explicitly) acknowledged. My thanks to Dorothy Goddard for inserting tooth picks under her eyelids and wading through this manual.

Last update: 27/11/93

2 OVERVIEW AND BACKGROUND INFORMATION

2.1 Overview

Let us begin with a rough outline of the steps that you might take in analysing your data in *AIPS*.

1. Load your data into *AIPS*.
2. Inspect the raw data and edit it to remove badly corrupt data.
3. Determine the complex antenna gains as a function of time (and possibly frequency) from the calibrator sources in your observation.
4. Apply the calibration to all desired sources.
5. Image (and deconvolve) desired sources.
6. Improve initial calibration with self-calibration of strong sources.

The succeeding sections describe these steps in detail. In addition, a flow chart showing a suggested path through the analysis maze is provided at the end of this section. The steps shown can sometimes be moved about in order.

You may want to make as good a calibration as possible with *MIRIAD* (see § 3.3 for a discussion on this). Although you could work entirely within *MIRIAD*, it may be preferable to combine the best features of *AIPS* with those of *MIRIAD*. This generally means loading your data into *AIPS*, editing it to remove bad data, and then shipping the data to *MIRIAD* for calibration and imaging. Generally, the TV display and image analysis software is better in *AIPS* than *MIRIAD*, so you may then return to *AIPS*(although *MIRIAD* does offer reasonable software in all these areas). See the *MIRIAD* manual for more information.

Before actually loading your data into *AIPS*, it is useful to discuss some of the general methods employed by the *AIPS* calibration package. I will assume some working knowledge of *AIPS*. If you are inexperienced with *AIPS*, then read the *AIPS* Cookbook, or ask for help from an unfortunate colleague or from your ATNF support scientist (if you are at Epping). There are also fairly extensive `HELP` files in *AIPS*, and even more extensive `EXPLAIN` files. Appendix A presents a list of some tasks that are relevant to the calibration and imaging procedures. Longer lists can be found in the *AIPS* Cookbook.

In general, I will not explain all *AIPS* adverbs that you must fill in for each task. If they are self-evident or well documented, then I leave it up to you. If I think they need specific discussion then I will do so; in particular, a group of adverbs used for most of the calibration tasks is discussed in Appendix B.

Your initial aim in *AIPS* will be to build what is called a multi-source visibility file. This file contains visibility data for many sources (possibly with different frequencies and bandwidths). The concept of the calibration package is that you never touch the actual data in the multi-source file, you just derive tables that are applied to the data in order to correct them. These tables are attached to the multi-source file and are specific types of what are called extension files. The number and type of extension files present can be ascertained with the general header viewing task, `IMHEAD`. Multiple versions of extension files can be kept, and there is a suite of tasks to delete, copy, and generally manipulate them (see Appendix A).

Examples of extension files that are not tables are the history (HI) and plot (PL) files. The tables applications tasks will not function on non-tables extension files. A correct use of the myriad of tables that the calibration package produces is essential. Appendix C explains what the most common tables you will encounter in *AIPS* are for and what some of their contents mean. Please take the time to read this.

As well as multi-source visibility files, *AIPS* also supports single-source files. As indicated by the name, these contain data for one source only. After you have finished calibrating with multi-source files, it is often convenient to apply the calibration directly to the data and split sources off into single-source files. The calibration tables are eliminated, as the actual visibilities will have been corrected. This saves you the book-keeping that goes along with keeping track of the calibration tables. Extension files such as the AN, HI, and PL files can still be attached to single-source files. The only calibration table that will generally become associated with single-source files is the solution (SN) table, because this is used in the self-calibration procedure.

It is also possible, depending on exactly what sort of imaging you wish to do, to work entirely with the multi-source files and not bother with splitting off sources into single-sources files. This may be advantageous if you have a large number of sources in your observation.

2.2 Some useful tips for new *AIPS* users

- *AIPS* makes the distiction between *tasks*, *verbs*, `procedures`, *pseudo verbs* and *adverbs*. A task is a process that is shed by the *AIPS* program and runs in the background so that control of the *AIPS* program is returned straight away to the user. They are generally used for long comptational jobs. Verbs are processe that are actually run by the *AIPS* program itself. They do not run in the background, and control of the *AIPS* program is not returned until they are finished. They are typically used for short computational jobs or interactive graphics processes. An example is the verb to list an image header, `IMHEAD`. Procedures are essentially scripts or macros that the user defines within *AIPS*. They are built from standard *AIPS* commands and are very useful for chaining together tasks and verbs. Pseudo verbs are commands that set some characteristic that the *AIPS* program then uses in the future. An example is the command `DEBUG true` which instructs *AIPS* that should a task be compiled and linked with the debugger, then it should be started in debug mode. Adverbs are the arguments to tasks, verbs and pseudo verbs which pass information from the user to the process.

  Tasks, verbs and procedures can all be activated with the `GO` command such as `GO ATLOD` (a task), `GO IMHEAD` (a verb), or `GO ATCALIB` (a procedure). In addition, verbs and procedures can also be activated by commanding the verb or procedure name only without the `GO`, but some people would rather not know this. Pseudo verbs can only be activated by commanding the pseudo verb's name followed by its argument, if any. Adverbs are passive only and don't `GO` anywhere. You assign values to them with statements such as `optyp='load'` (strings are quoted) or `nfiles=10`.

  I will probably use words such as task, verb, and command interchangeably in this manual.

- Note that all *AIPS* adverbs are *global*; they are not attached to any task in particular. If you wish to recover your inputs for a task as you last ran it, then use the very useful `TGET` command (such as `TGET ATLOD`). The golden rule is always to review your adverbs to any *AIPS* task with the `INPUTS` command before setting it loose.

- Rather than always having to say `GO TASK`, you can set the adverb `task` equal to the task you are interested in and then activate the task with the `GO` command alone. For example, setting `task='split'` and the entering `GO` would activate the task `SPLIT`. Similarly you could enter `INPUTS` at this point and the adverbs for `SPLIT` would be listed. Note that procedures and verbs do not honour the `task` adverb. Entering `task='atcalib'` followed by `GO` will not activate the procedure. You must activate it and verbs as decribed above.

- *AIPS* is not case sensitive. You can speak to *AIPS* in either lower or upper case and internally, it will convert to upper case. This means, that whenever *AIPS* needs to access a file, say, from the host operating system (such as Unix), that file must have an upper-case file name. Since the path to that file (in Unix) might also have lower or mixed case, you must use an upper-case environment variable which specifies the path to the file. This is done with adverbs such as `infile` or `outfile` where you would set them to something like `infile='myarea:myfile'`. *AIPS* would convert all this to upper case, so `myarea` must in fact be an upper case environment variable set under Unix before you started *AIPS* with a command such as

  ```
  setenv MYAREA /DATA/MENSA_1/miriad/nkilleen
  ```

  and `myfile` is really an upper-case file called `MYFILE`.

- *AIPS* also likes to write whatever you do into the message file (if you are a student we pass these message files onto your supervisor). At Epping, this message file resides by default on Lynx which is not the computer you are running *AIPS* on; it is a file server. As described in the next section, there is a network penalty for *writing* to a remotely mounted disk. If you would prefer that *AIPS* did not write anything at all into your message file, you can issue the command (really a pseudo verb if you must know) `MSGKILL true` which remains active until either you exit from *AIPS* or you issue the `MSGKILL false` command. You will note that commands such as `IMHEAD` to list the header of a file will run much much faster without this overhead. It is up to you whether you want to keep track of what you have done with the message facility or not. Most people find that they enter the command `CLRALL` rather often (to clear the entire message file) as *AIPS* starts to bleat if it gets too full. However, each *AIPS* file also has a history file (viewed with the command `PRTHIS`) associated with it so that you can always work out how your file got to be in its current state if need be.

  You can list the contents of the message file with the verb `PRTMSG` and clear bits of it (rather than all of it) with `CLRMSG`.

2.3 File system management and network use at the ATNF

In this section I will describe the networking environment you will encounter at the ATNF. This description is true at both Epping at Narrabri.

2.3.1 *File systems*

Each of the workstations, and the Convex (a low-end workstation) have a number (usually 2 or 3) of disks available for data storage purposes. The file systems are called /DATA/HOST_N where 'HOST' is the computer name and N is the disk number. Some of these workstations and their disks are bookable by users. Others are all-user scratch disks which are wiped regularly, others are all-user longer term storage areas. For example, the data file system on the bookable disk on PUPPIS is /DATA/PUPPIS_3. For the Convex the first of the two all-user disks is /DATA/ATELES_1.

To find out what the status of all the available disks is, enter the command

```
aipsdisks
```

at the Unix prompt level and you will be enlightened.

Under these top level file systems on each disk, are several more directories. These are 'AIPS', 'aips++', 'miriad', and 'other'. *AIPS* files go into the 'AIPS' directory (*AIPS* will put them there for you when it creates them). The other directories are used appropriately.

For temporary storage of large data files on disk, you should use the 'other' directory (except for files apropriate to the other standard named directories; do not clutter the 'AIPS' directory with RPFITS files). Change to this directory with a command such as

```
cd /DATA/PUPPIS_3/other
```

You should then create a directory for your own use with a command such as

```
mkdir ehubble
```

(if you are less famous try something more appropriate) and then enter that directory with

```
cd ehubble
```

You can then use this area for your workspace.

2.3.2 *How to use the network*

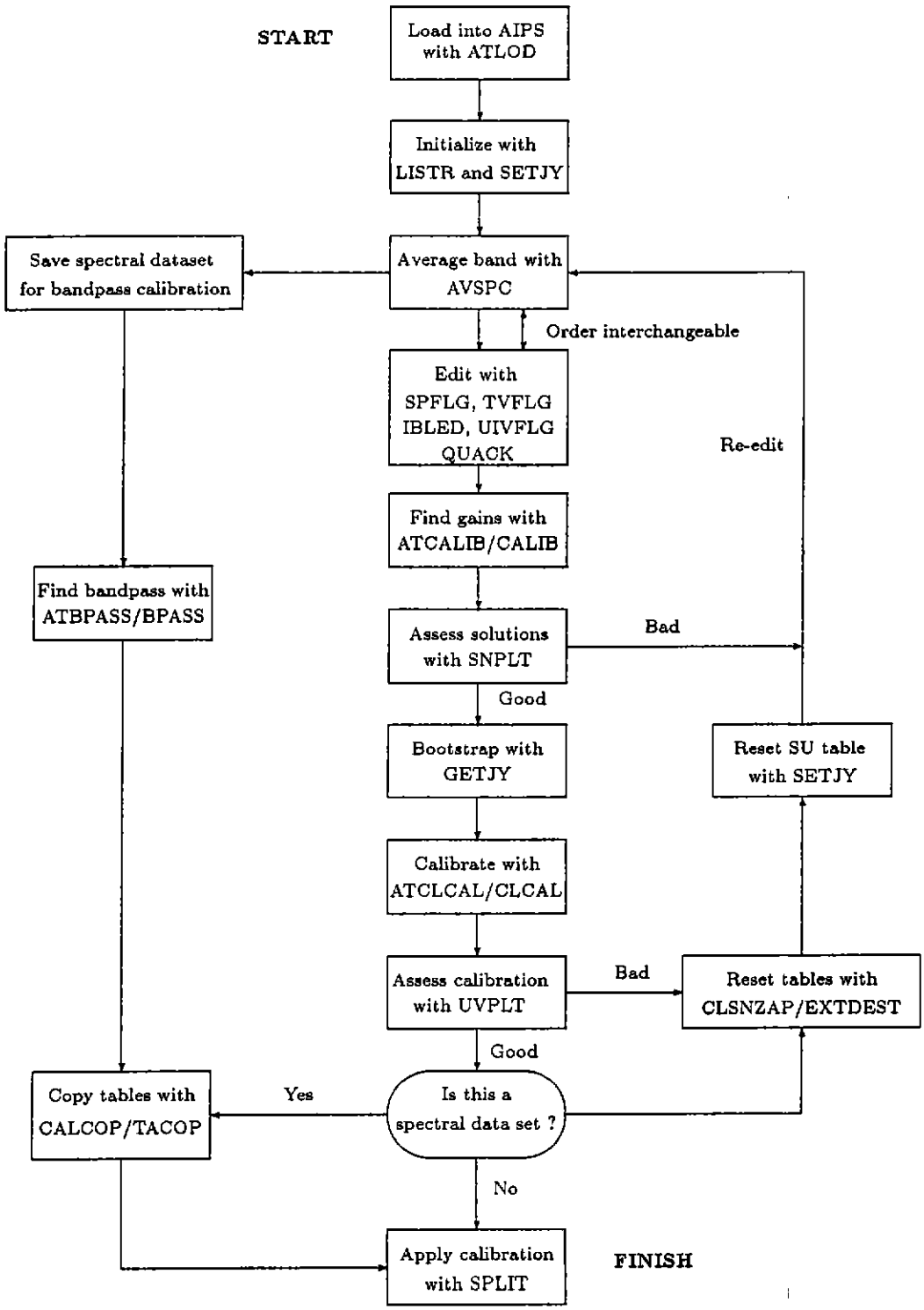
All the disks attached to all the different workstations can be accessed from any workstation or the Convex via the automounter. For example, if you are logged into PUPPIS, you can see the file systems on APUS by typing, say,

```
ls /DATA/APUS_2
```

However, there is a right and a wrong way to use this cross mounting. There is a large network penalty if you attempt to *write to* a remotely mounted disk. For example, if you try to write to a disk on APUS while logged into PUPPIS. However, the penalty is smaller if you attempt to *read from* a remotely mounted disk. In general, you should try to maintain your files on the disks of one workstation, and remain logged into that workstation. If you are forced to access files from a remotely mounted disk, *always* make sure you *read from* that disk, do not *write to* it.

Remember also that when you access a remotely mounted file system, the workstation on which those file systems belong will be impacted also. Several daemons will start up in order to provide the service which transfers the data to or from your recipient file system. Thus, consider users on the other workstations when operating across the network.

Finally, note that *AIPS* often generates scratch files. In many tasks you have control over which disks these scratch files *do not* go to via the baddisk adverb. Thus, if you have started *AIPS* such that you have remotely mounted disks available (which you should read from only), you can set baddisk to point at these disks so that *AIPS* will be dissuaded from writing scratch files to them. Scratch files are often very large and the penalty for not doing this can be enormous.



Flow chart for calibration

Last update: 27/11/93

3 BASIC CONCEPTS OF CALIBRATION FOR ATCA DATA

In this section I give some background on polarimetry and antenna based calibration. Ideally, this section would come after you have loaded and edited your data. However, some of the concepts discussed here are required background for some of the discussion of the task that loads your data into *AIPS* (ATLOD).

More extensive reading on the topics of this section can be found in the NRAO synthesis workshop books and in "Interferometry and Synthesis in Radio Astronomy" (Thompson, Moran and Swenson). Note however that the NRAO books tend to focus on telescopes with circularly polarized feeds (such as the VLA) when they discuss polarimetry. The ATCA has linearly polarized feeds, which has some thorny ramifications for users as discussed in the last two subsections below.

A synthesis telescope such as the ATCA records the Fourier transform of the intensity distribution of the sky. The ensemble of complex numbers recorded by a synthesis array is referred to as the *observed visibilities*. The definition of a single visibility is a bit murky and is really software-implementation specific. In its minimalist form, a visibility is one complex number, representing one sample of the Fourier transform of the sky measured by one baseline in one polarization in one integration in one spectral channel. In its broadest sense, and one often used in *AIPS* jargon, it is a collection of complex numbers representing all the channels and polarizations in one integration for one baseline.

Because the cosmic signals must traverse the ionosphere, the atmosphere, and a complicated train of electronics, the observed visibilities often bear little resemblance to the desired Fourier transform of the source. The purpose of calibration is to determine the corrections necessary to produce a good estimate of the true visibilities, which can then be Fourier transformed to the image domain.

3.1 Antenna-based gain calibration

The most basic form of calibration that the *user* undertakes (there are many forms of calibration taken care of by the on-line system) is aimed at making corrections for changes in the visibilities induced by the atmosphere and electronics that the on-line system cannot deal with. Such changes may be on time scales from minutes to hours.

We determine the corrections by observing known sources. It is advantageous that their structure be as simple as possible. Ideally, we would like a strong unresolved source with a well known position. If such a source is observed at the phase centre, then regardless of the baseline, the visibility amplitude should be equal to the source flux density, and the visibility phase should be zero. The *observed* visibilities of such a source thus yield direct estimates of the desired corrections. The corrections that we determine are referred to as the complex (amplitude and phase) gains, and the calibration formula can be written simply

$$V'_{ij}(t,\nu) = G_{ij}(t,\nu)V_{ij}(t,\nu),$$

where t is the time of the observation, ν is the frequency, i and j refer to a measurement associated with a pair of antennas (i,j), V is the true visibility, V' is the observed visibility, and G is the gain.

The visibilities and gains are complex quantities; they have an amplitude and a phase (or a real and imaginary part if you prefer to think that way). Each instantaneous visibility measures a Fourier component of the sky intensity distribution. The combination of these two-dimensional 'corrugations' with the correct (as they should be after calibration) relative phases and amplitudes yields the sky intensity distribution convolved by the point response function of the synthesis array.

Since most of the data corruption occurs before correlation, we can factor the baseline-based gain, G_{ij} into antenna based gains, G_i and G_j so that the calibration equation becomes

$$V'_{ij} = G_i G_j^* V_{ij}.$$

If there are N antennas, then there are N complex antenna gains, G_i. However, there are $N(N-1)/2$ baseline based gains, G_{ij} from which we must extract these N antenna gains. That is, we must solve the set of equations

$$G_{ij} = \frac{V'_{ij}}{V_{ij}} = G_i G_j^*.$$

Because this set of equations remains true even if we subtract an arbitrary phase from every G_i, we can subtract the phase of one G_i from all the G_j, so that the phase of G_i becomes zero. This is known as the reference

antenna. The number of real valued unknowns is therefore $2N-1$. A non-linear least squares technique is usually used to extract these from the $N(N-1)/2$ complex equations. In practice, we deal with the amplitude and phase of the gains, rather than the real and imaginary parts. This is because the primary effects of the wave propagation are to rotate the phase and attenuate the amplitude.

3.2 Stokes Parameters and Polarization

The (Sir George) Stokes parameters provide a very useful description of the polarization state of an electromagnetic wave. Because of the importance of polarimetry in radio astronomy, I will give a brief description of Stokes parameters, their relation to the quantities measured by the ATCA and the ensuing ramifications for calibration purposes. See also the *MIRIAD* manual for good (we think) descriptions of polarimetric issues involving the ATCA and the ATNF scientific memorandum by Sault, Killeen, and Kesteven on ATCA polarization calibration.

The Stokes parameters are related to the amplitudes of the components of the electric field, E_x and E_y, resolved in two perpendicular directions normal to the direction of propogation. If E_x and E_y are represented by $e_x(t)\cos[\omega t+\delta_x]$ and $e_y(t)\cos[\omega t+\delta_y]$, respectively, then the Stokes parameters are defined as:

$$\begin{aligned} I &= <e_x^2(t)> + <e_y^2(t)> \\ Q &= <e_x^2(t)> - <e_y^2(t)> \\ U &= 2<e_x(t)e_y(t)\cos[\delta_x-\delta_y]> \\ V &= 2<e_x(t)e_y(t)\sin[\delta_x-\delta_y]>, \end{aligned}$$

where the angle brackets denote a time average. The parameter I is a measure of the total power in the wave, Q and U represent the linearly polarized component, and V represents the circularly polarized component. These values can be converted to a measure of polarization as follows:

$$\begin{aligned} m_l &= \sqrt{Q^2+U^2}/I \\ m_c &= V/I \\ m_t &= \sqrt{Q^2+U^2+V^2}/I \\ \chi &= \frac{1}{2}\tan^{-1}(Q/U), \end{aligned}$$

where m_l, m_c, and m_t are the degrees of linear, circular and total polarization, and χ is the position angle of the plane of polarization. The Stokes parameters have the dimensions of flux density, and they combine additively for independent waves.

The next question is how to relate the Stokes parameters to the correlations that an interferometer produces. In general, the response of an interferometer is a linear combination of two Stokes parameters, the combination determined by the polarization of each antenna (e.g. linear feeds for the ATCA or circular feeds for the VLA). By observing with different combinations of polarizations, all the Stokes parameters can be determined, and the complete state of the polarization of the wave found. For the interferometer, each Stokes parameter has a corresponding complex visibility. Therefore, the variation of each of the Stokes parameters over the source can be imaged individually, and the polarization of the radiation emitted from the source at any location determined. Note that in the *image* plane, I is always positive (ignoring noise, errors and beam sidelobes), whereas Q, U, and V may be positive or negative depending on the polarization position angle, or sense of rotation.

The correlations from an interferometer with perfectly linearly polarized feeds and identical gains are related to the Stokes parameters as follows:

$$\begin{aligned} XX &= I+Q \\ YY &= I-Q \\ XY &= U+iV \\ YX &= U-iV, \end{aligned}$$

where the X and Y feeds are orthogonal and in the equatorial reference frame of the source. It is instructive to look at the equations for telescopes (such as the VLA) with circularly polarized feeds as well. In this case,

$$RR = I+V$$

$$
\begin{aligned}
LL &= I - V \\
RL &= Q + iU \\
LR &= Q - iU,
\end{aligned}
$$

In the discussion of § 3.1 there is no polarization information explicitly included in the calibration equation; the model must match the data. For circular feeds, the V_ij generally represents the RR and LL visibilities and the calibrator model source is assumed to be not circularly polarized so that RR and LL respond purely to Stokes I. However, it is not so simple for linear feeds. Here we associate the V_ij with the XX and YY visibilities. But now, the model visibilities should really include the fact that XX and YY respond to Stokes Q and U as well as Stokes I. However, generally, we do not know what the polarization characteristics of a calibrator are, so that this is not straightforward. This is why it is more complex to calibrate linearly polarized feeds compared to circularly polarized feeds.

The above equations are for ideal antennas. In reality, the feeds are not perfectly linearly polarized, so that an observation of an unpolarized sources yields a source which appears polarized. Furthermore, the feeds rotate with respect to the equatorial reference frame of the source, so that the above equations are not directly useable. A polarization calibration attempts to remove both these effects leaving just the source polarization in the data.

3.3 Should I calibrate in *AIPS* or *MIRIAD* ?

When you calibrate in *AIPS*, it is simply assumed that the calibrator is unpolarized. Calibrator sources are typically linearly polarized at a few percent, and generally not more than 10 percent. When you calibrate ATCA data with *AIPS*, this error goes directly into the gains that you determine. However, if your program source is strong so that you are going to be able to self-calibrate it (see § 17) then this error can be largely removed. In addition, because *AIPS* is poorly attuned to linear feeds, you will be unable to extract any Stokes parameter other than I from your ATCA data. Finally, you will not be able to account for the antenna polarizations either (although this is a second order effect in total intensity).

When you calibrate in *MIRIAD*, it is possible to account for both the calibrators polarization and the antenna polarizations so that an optimum calibration is possible. However, despite the calibration virtues of *MIRIAD*, you should be aware that most spectral-line observations record only the XX and YY polarizations. Without the XY and YX correlations we are unable to make a full and correct polarimetric calibration, and the calibration in *MIRIAD* would be the same as the one in *AIPS*.

In general, to get the best possible calibration of any XX, YY, XY and YX observation, whether you are interested in total intensity only or all Stokes parameters, you should calibrate in *MIRIAD* as described in the *MIRIAD* User's Guide. If you intend to calibrate in *MIRIAD*, then you need to concern yourself with the XY phase differences which are discussed below in § 3.5. If you have only XX and YY polarizations, both *AIPS* and *MIRIAD* make the same calibration.

3.4 Do I really an offline secondary calibration at all ?

Heresy you say. Of course I must calibrate. I love to calibrate. Slowly but surely, we are moving in the direction of producing fully calibrated data from the ATCA. We are not there yet, but we are part of the way. This section gives some hints on how useful the on-line calibration is, but does really assume you have more knowledge than has so far been presented in this manual.

When you began your observation, you did some set up chores which involved a program called CACAL (formerly you would have used DELCOR). These chores were really to make a basic on-line calibration of antenna delay and complex gain. These are the very same gains that the usual offline secondary calibration process wants to redetermine. Because CACAL knows the spectrum of 1934 – 638 (if you set up with it) your data should already be on a flux density scale that is essentially correct (if 1934 – 638 was at a very low elevation it might be a bit wrong). That is, the antenna complex gain amplitudes should be pretty much right.

The usual idea behind secondary calibration is that the atmospheric phase changes on a scale of size of roughly a few degrees on the sky (at cm wavelengths), so that the phase you determine from the primary calibrator (either on line or off line) is probably not going to transfer unless you happen to be observing very close to it. In addition, the atmosphere is dynamic and the atmospheric phase changes with time; sometimes very rapidly and sometimes not so rapidly. The secondary calibrator is used to determine the phase close to your program source and to track it with time. The main component of the change of gain phase with time is the atmosphere. The receivers are very phase (and amplitude) stable and are unlikely to drift significantly during your observation. We

generally also use the secondary calibration to redetermine the gain amplitudes with time because the corrections made on line to account for attenuation effects with elevation (T_{sys} correction) are usually not perfect.

Now although the phase may wander with time, it very unlikely that the differential phase between the XX and YY feeds will change. Since the basic on-line calibration results in XX and YY being in phase, they will remain so, even if their phase changes with time. This means that in *AIPS*, rather than examining the XX and YY data separately, you can combine it straight away and examine Stokes I. For example, when (if necessary) you are editing your data with `TVFLG`, say (see § 7), you could request Stokes I and pass through the data once, rather than requesting XX and then YY and passing through the data twice.

Now all this means that if the phase stability of your observation was good, it is possible that the basic on-line calibration that you did will be useful at some level. Unfortunately, we do not yet produce an on-line band-pass calibration, so if you are doing spectral-line work, then you need to work your way though the off-line calibration procedure. If you plan to do an optimum calibration in *MIRIAD*, you should do just that. But let us say you have decided to remain within *AIPS*. Let us also say that you do not need a band-pass calibration and you have averaged all of your channels together (see § 6 for details). You could then make an image of Stokes I (no other Stokes parameters are available in *AIPS* for ATCA data) directly from the on-line calibrated data. Then, if your source was strong enough, you could rely upon self-calibration (see § 17) to improve the complex gains.

3.5 All about XY phase differences

If you have XX and YY polarization data only, you can ignore (gratefully) this section.

A thorny part of calibrating ATCA data concerns the XY phase differences (Φ_{XY}). This is the phase difference between the X and the Y feeds on each antenna; in principle they should be steady in time. If we did not know Φ_{XY}, the phase relationship between the XX, YY, XY and YX correlations would be arbitrary and we could not combine them to make the Stokes parameters. Thus, Φ_{XY} is used to rotate the phase of the Y gain to that of the X gain, so that the XX and YY visibilities are in phase (this gives the XY and the YX visibilities the correct phase relationship as well).

Φ_{XY} is measured for each antenna every integration via a noise source in each feed horn which is switching on and off continuously throughout your observations. Before November 1992, this measurement was quite poor because it detected the on plus the off signal. Any ambient signal (and there is plenty, including your source and ground pickup) would contribute to the measured value of Φ_{XY}. Since this time, the difference between the on and off signals is now detected and the quality of the Φ_{XY} values has improved significantly.

However, despite this improvement, they are still not of sufficiently good quality to use blindly. This is especially true in the 13 and 20 cm bands where deviations which do not appear to represent actual deviations occur.

If you are to be concerned with Φ_{XY} values, you will be calibrating your data in *MIRIAD*. The calibration program solves for Φ_{XY} for each antenna relative to the reference antenna. Therefore, it is only important that you get a good measurement for one antenna. Usually, the on-line reference antenna has the best Φ_{XY} values, but this does not have to be the case. It also turns out (see Sault, Killeen, and Kesteven 1992) that knowing the absolute Φ_{XY} on the reference antenna is unimportant for many observations so that these relative solutions are usually sufficient.

Currently, our preferred path with regards Φ_{XY} values is to deal with them entirely in *MIRIAD*. It is still possible to apply them with *AIPS* `ATLOD` as current users will be familiar with, but since *MIRIAD* offers greater flexibility we would like to isolate application of Φ_{XY} to that part of the process. Please refer to the *MIRIAD* manual for details. You should be aware of a convention difference between *AIPS* and *MIRIAD* for Φ_{XY}. A formula to convert is given in the *MIRIAD* User's Guide.

Last update: 27/11/93

4 LOADING YOUR ATCA DATA INTO *AIPS*

Armed with a complete comprehension of tables and *AIPS*, you can now proceed with the mechanics of your data analysis and begin by loading your data into *AIPS*. There are two routes for loading your data, depending on how you wrote your tape, and what sort of tape it is. Very early ATCA data was written with the VMS 'BACKUP' command on 9-track magnetic tape. These data must be read onto a Convex data disk, and then loaded into *AIPS*. Current ATCA data are written with the VMS 'COPY' command onto Exabyte tapes (thus they are ANSI standard tapes) and can be read directly into *AIPS*.

If at any time you need to use the Convex rather than the workstation (e.g. to access a 9-track tape drive), then remotely log into the Convex from the workstation with the UNIX command

```
rlogin ateles
```

Note that UNIX is case sensitive, so you should be careful to follow the instructions with regards case as well as alphabet.

4.1 Data on Exabyte tape written with VMS COPY

This section is for ATCA data (RPFITS files) written on an Exabyte tape with the VMS operating system command COPY. This is the usual way in which ATCA data is archived. The tape structure is that of the ANSI standard. This means that for each RPFITS file, there is first a header file (with file names etc. encoded into it), the data file, and last a trailer file. You need to be aware of this if you position the tape drive with the *AIPS* command AVFILE as for every RPFITS file you wish to move, you must actually advance 3 tape files. Similarly if you manipulate the tape device from the Unix operating system with the mt command.

You must log into the workstation that has the desired tape drive associated with it. This is usually the workstation that you have booked. As discussed in § 2.3.2, it is also highly desirable that you write the data to a *local* disk associated with that workstation. Writing to a remote disk mounted on another workstation incurs a high network penalty and should be avoided (it penalizes users on the other workstation as well as yourself.

AIPS does have a remote tape mechanism by which you can access tape drives on other machines. However, for the purpose of loading ATCA data, this mechanism is **NOT** available. This is because the task that reads RPFITS files (ATLOD) does not use the standard *AIPS* tape handling software.

4.1.1 *Loading the data directly from Exabyte tape into AIPS*

Here I describe how to read your RPFITS files on an Exabyte tape directly into *AIPS*.

1. At the UNIX prompt, start up *AIPS* on the workstation with the *lower case* command

   ```
   aips
   ```

   and type in your *AIPS* user number when prompted.

2. Insert the Exabyte tape into the drive and prepare to select the tape drive in *AIPS* by setting `intape=n`, where n is the tape device number. The *AIPS* tape drive number is written on a label on each Exabyte tape drive. It is usually 1 or 2. Allocate the tape with the *AIPS* command MOUNT.

3. You can now read the data from tape with the task ATLOD. I will discuss ATLOD in detail in § 4.4. Refer to that section for information on its many inputs. But note here that you should make sure the adverb `infile` is blank, otherwise ATLOD will look for its input from disk rather than tape.

4. Once ATLOD has run to completion, dismount the tape device with the *AIPS* command DISMOUNT and collect your Exabyte tape from the tape drive.

4.1.2 *Copying your data from Exabyte tape to disk*

In this section I describe how to copy your RPFITS files from tape to a Unix file system outside of *AIPS*. The output file is thus still an RPFITS file.

1. First set your working directory to the appropriate data area (see § 2.3) with a command such as

   ```
   cd /DATA/PUPPIS_3/other/ehubble
   ```

2. You can now copy your RPFITS files from Exabyte to RPFITS files on the disk in this area. Use the Unix command **ansiread** for this purpose. You can do **man ansiread** to learn all about it. Here is an example of how to read the second and third RPFITS files from your Exabyte tape.

   ```
   ansiread  -f /dev/nrst0  -s 1  -r 2
   ```

   where `/dev/nrst0` is the tape device name (it is written on the Exabyte tape drive cabinet), the `-s 1` directs it to skip one file and the `-r 2` directs it to read two files. The file will be created on disk with the name given to it by the on-line observing computers. You can then read it into *AIPS* (or *MIRIAD*) from here as described in § 4.3 if you wish.

3. Once you are finished, eject the tape by pushing the button on the tape drive cabinet, or by issuing the Unix command

   ```
   mt  -f /dev/nrst0  offline
   ```

4.2 Data on 9-track tape written with VMS backup

This is relevant only at Epping, and use of 9-track tapes has been phased out so this is applicable only to very early ATCA data. You cannot read directly into *AIPS* from this format, so first you must extract the RPFITS files onto disk outside of *AIPS*.

1. You must remotely log into the Convex from the workstation with

   ```
   rlogin  ateles
   ```

2. Physically mount your tape (after removing the 'write enable' ring) on either of the Convex tape drives, and allocate it (in 'norewind' mode) by typing

   ```
   tpmount  -a /dev/rmt21  -b
   ```

   for the left-hand drive or

   ```
   tpmount  -a /dev/rmt20  -b
   ```

   for the right-hand drive.

3. Now change your working directory to the desired data area (see § 2.3) with a command such as

   ```
   cd  /DATA/FITS
   ```

   for the generic Convex data area, or

   ```
   cd  /DATA/ATELES_1/other/ggalilei
   ```

   for a user-specific area.

   You could set your working directory to a workstation data area but you (and everyone else) will pay the network penalty of writing to a remote disk (see § 2.3.2). More effective (except for small files) is to load your data to the Convex disk, and then copy it from the Convex to the workstation using the workstation (i.e., after you log out of the Convex) or read it straight from the Convex's disk into *AIPS*.

4. Read the RPFITS data off the tape with the command

   ```
   vmsbu  -d21  -f0
   ```

   for the left-hand drive or

   ```
   vmsbu  -d20  -f0
   ```

   for the right-hand drive (the `-f0` specifies that all files from the save-set be extracted) and answer each of the questions with a carriage return. Each file name will be printed on the terminal as it is extracted.

5. If you need to reach a save set other than the first, position the tape with a command such as

```
mt  -f /dev/rmt21  fsf 2
```

which moves the tape forward two files. I believe it is necessary to move this many to get to the second save-set on the tape from the beginning.

6. Deallocate the tape drive by typing

```
tpunmount
```

for either drive, and remember to go and collect your tape.

7. The files will be created in the Unix file system and you can list them with the Unix command `ls`. You can now read the files into *AIPS* (or *MIRIAD*) from the disk files as described in § 4.3 if you wish.

4.3 Reading your data from disk into *AIPS*

You may have decided to first load your data onto disk outside of *AIPS* and now you want to convert it into *AIPS* format. You may have done this because you wish to experiment with the many options in ATLOD and it will be faster to read from disk than tape. This is the section for you. You can load the data from disk into *AIPS* with either the Convex or a workstation depending upon where you have your disk booking (it could be the Convex) and what your needs are. Generally, you will be doing this with a workstation. We encourage you to restrict your use of the Convex to large computational jobs.

1. If you wish to be able to read into *AIPS* all the disk RPFITS files that you have created with one application of ATLOD, then you may find it convenient to concatenate all the files into one with a Unix command such as

```
cat  91-04-23_*  >  BIGFILE
```

where your files are all prefixed by `91-04-23_`, say, and suffixed by the the UT (wild-carded with the *). It will save you some trouble later on in *AIPS* if the concatenated file is in time order. The current file naming convention should ensure this (i.e., the files list in time order when you examine a directory of them with, say, the UNIX command `ls`), but data from the first half of 1990 may not. If not, then put them all in time order on the command line in the cat operation. For example,

```
cat 31jun90_2301  01jul90_0044  01jul90_0413  >  BIGFILE
```

Note that you must make the name of the concatenated file in upper case (here I have called it 'BIGFILE') because of the disinterest *AIPS* shows in lower case. Remember to delete the multiple files, leaving only the concatenated file. Do this with a command such as

```
rm  31jun90*  01jul90*
```

If there was insufficient space to concatenate the files, then just rename them to upper case with the command

```
RENAM .
```

The '.' signifies that RENAM should do its work in the current directory. You will do the concatenation within *AIPS* with the task ATLOD in this case.

2. Before you start *AIPS*, you must set an *upper-case* environment variable to point at the location where you have stored your RPFITS files. Do this with a command such as

```
setenv MYAREA 'PWD'
```

Note that this is not the standard Unix lower case command `pwd` but an ATNF specific upper case one which does not get confused by the automounters. Note also the backwards quotes; you must type these as shown. You can examine the value of 'MYAREA' with

```
printenv MYAREA
```

3. Start *AIPS* on whatever machine you choose to use, with the *lower case* command

```
aips
```

and enter your *AIPS* user number when prompted.

4. You can now load the data into *AIPS* with ATLOD. All of the inputs to ATLOD are described in § 4.4. Note here that you must set the adverb `infile` to point to the file you wish to read from disk. For example, set `infile='myarea:bigfile'` (you must include the forwards quotes; all strings in *AIPS* are quoted). If you do not include the environment variable, ATLOD will look, by default, in the /DATA/FITS area for your file.

4.4 How to use ATLOD

ATLOD has a long list of adverbs, and I will discuss them all and in the order in which they list in *AIPS*. Note that if you have a large number of sources to load in one go (i.e. more than 400), please alert a myself or Henrietta May so that we can ensure that there is adequate internal storage in ATLOD for your needs.

1. *Input file specification adverbs*

   - `intape` should indicate which tape drive you mounted and plan to read your RPFITS files from (`intape` is irrelevant if `infile` is set).
   - You can skip files at the beginning of the tape with the `nskip` adverb. It is a little tricky to skip files on a VMS 'COPY' tape with the *AIPS* verb AVFILE because the ANSI format actually has three files (a header, the data, and a trailer) per RPFITS file. You have to remember to skip three for every on If you get lost you can get really lost, because the verb TPHEAD, which generally lists the header of the next file on a tape, does not recognize ANSI file headers. Note however, that ATLOD now skips files as fast as AVFILE and you don't have to be able to multiply by three (`nskip` is irrelevant if `infile` is set).
   - Select the number of files that you wish to load with `nfiles=10` (say, for 10 files). All the input files selected on the tape will be concatenated into the one output file, so it helps if they are in time order on the tape – if not, you will need to use UVSRT and INDXR as described in § 5). The default is that ATLOD loads one file (`nfiles` is irrelevant if `infile` is set).
   - `bcount` and `ncount` give you control over which scans you would like to load from the file. Leave them at zero for all scans, or specify the first scan to load with `bcount`, and the number of scans to load with `ncount`. These apply to *all* files loaded, and are mainly useful for disk file input (when `infile` is set).
   - If `infile` is blank, ATLOD will try to load data from the specified tape device (`intape`). Otherwise it will look for disk RPFITS file input. You should set this adverb to something like `infile='MYAREA:BIGFILE'` where 'MYAREA' is an upper-case environment variable pointing to the directory where the upper-case file 'BIGFILE' resides (see § 4.3). If you don't include the environment variable, ATLOD will look for the specified file in the /DATA/FITS area.

2. *Output file specification adverbs*

   - Fill in the output file name with commands such as `outname='multi'`, `outclass= 'uv'`, `outseq=0`, and `outdisk=1`. Select the local disk with the most free space (use the *AIPS* command FREE to see which). Note that if you completely specify these four adverbs, ATLOD will always try and put the data it is reading into that file. If you leave, say, `outseq` at zero, every application of ATLOD will try to make a new file with an increasing version number. One application of ATLOD reading multiple input files will always put the output in one file (even if `outseq=0`). This is the usual mode of use.

     DBCON can be used to combined multi-source files from the same observation. However, it has some problems with multi-source files, so please try to do this concatenation with ATLOD.

     Note that data from different observing runs with different configurations should *not* be combined into one multi-source file with ATLOD. This is largely because baselines with the same number but from different configurations would become confused. In principle, the `subarray` capability of *AIPS* could be used to deal with this, but ATLOD does not currently take advantage of this. There are other practical problems too. Multi-configuration data can be combined after calibration with DBCON. This is discussed in § 14.

3. *Action selection adverb*

- ATLOD has the capacity to just list summaries of the contents of the file rather than actually load it into *AIPS*. If you would like to see summaries of each scan, set `optype='list'`. For very brief summaries of each file, set `optype='summ'`. Otherwise, to load the data, put `optype='load'` or leave it blank.

  A further option, `optype='sysc'`, is available to write three text files containing the on-line measurements of ancillary system calibration information. These text files can contain (see also `cparm(9)` the antenna-based XY phase differences (Φ_{XY}), the system temperatures and sampler statistics for the X and Y feeds of each antenna. These files are written into the /DATA/FITS data area. The only other selection adverbs that are active in this mode are `freqsel`, `ifsel`, `sources` and `timer`. You can use the program `pltsys` to plot the contents of these files. Just enter the command `pltsys` at the Unix prompt and answer the questions. This program has frequency discrimination, so you can use this option of ATLOD and select all frequencies when generating the text files rather than running ATLOD once for each frequency.

  A further option is `optype='losy'`. This option is a combination of `optype='sysc'` and `optype='load'`. It loads your data and generates the system calibration text files simultaneously. At the time of writing, it has not been coded, but should be available soon.

4. *Data selection adverbs*

   - You can select a subset of the available sources in your RPFITS file to load by specifying a list to load with the `sources` adverb. If you prefix *any* of the sources in the list with a minus sign, such as `sources='-0032+11',' '`, then all sources except the named sources are loaded. The default (blank) means load all sources.
   - Load sources from a specified time range with the `timerange` adverb. The first day in the data is day zero, so, to select data from 12 hours and 33 minutes into the first day (day zero), until 2 hours, 12 minutes and 53 seconds into the second day (day one), use `timer=0,12,33,0, 1,2,12,53`. To load all data, leave it at zero.
   - Select data at the desired frequencies (this refers to the frequency at the band centre) with a command such as `freqsel=4740,8500`, where the frequency must be in MHz. Load data at all available frequencies by leaving `freqsel` at zero. The tolerance for matching frequencies is 1 KHz.
   - Select data from the desired IFs with a command such as `ifsel=2,3,4`. For example, consider an observation where IF 1 was set to the 2300 MHz band and IF 2 was set to the 1400 MHz band and you cycled in time through different pairs of frequencies in these bands. If you wanted to load just the 1400 MHz frequencies, you could list them all with `freqsel` but easier would be to just set `ifsel=2`.
   - You can load a subset of the available channels in each spectrum with the `chansel` adverb. Leaving it at zero ensures all channels are loaded. You can specify up to 10 groups (start, end and increment) of channels; all others will be loaded, but flagged. In the special case of one group of channels (such as `chansel=7,25,2`) ATLOD will actually drop the unselected channels and adjust the channel spacing in the file header to reflect this.

     For example, to load (unflagged) channels 1 to 320, and 322 to 513, use `chansel=1,320,1, 322,513,1`.

     This option would be useful if you are forced to use data compression (see below) and there is a channel with strong interference in it. The unselected channel is not used when scaling the visibilities into integers.

     You may also find it useful to discard a group of channels at the beginning and end of the spectrum where the filter response is poor. In this way also, should you have chosen data compression, you can increase the potential dynamic range of the data, as the range of visibility amplitudes from the central portion of the band is likely to be smaller than the range from the entire band. This also helps with any disk space problems.

     You should also consider discarding every second channel if you have invoked Hanning smoothing (see `dparm(2)`).

5. *Output file control adverbs*

   - If you have data with 2 or more simultaneous frequencies, the IF axis of the AIPS visibility file structure can be used for the contemporaneous data. It is a regular axis, so the IFs are numbered 1 to N. The actual frequency of the IF axis location is tracked by the use of offsets in the FQ table (see also Appendix C). Each entry (FREQID) in the FQ table is in fact an array of length equal to the length of the IF axis. Thus, each FREQID delineates a different collection of simultaneous frequencies.

For example, you might observe at 4400/4600 for a while, and then at 8400/8600 for a while. Each pair of simultaneous frequencies would be described by a separate FREQID.

Setting `ifmap=1` will cause `ATLOD` to map the on-line IF number to the AIPS IF axis location. For example, if you observed at 4400/4600, selected only frequency 4600, and set `ifmap=-1` (default) then `ATLOD` would put this in IF axis location one. However, if you set `ifmap=1`, then `ATLOD` would put it on IF axis location 2, and zero fill location 1 (this would be wasteful of disk space of course).

⇒ Generally, I suggest that you set `ifmap=-1`.

- `NIF` allows you to tell `ATLOD` how long the IF axis should be. Normally, `ATLOD` will work out the length of the IF axis based upon the information present in the first scan header that is consistent with the user's selection crteria. However, it may be that you know more than ATLOD, and you need to tell it the length of the IF axis.

  Consider the case where you have an observation such as

	IF chain 1	IF chain 2
Scan 1:	4700	8300
Scan 2:	4600	8000

  Let us say you set, for some reason known only to yourself, `freqsel=4700.,4600,8000`, thus omitting the 8300 frequency on IF chain 2. Now, you MUST set `nif=2` to get all these frequencies loaded. `ATLOD` notes that you want one frequency from the first scan, and that it would go to IF axis location 1. But this is the only information it has when it reads the header for the first scan, which is when it must build the attributes for the output file. It doesn't know that the other specified frequencies are is succeeding scans and that they will need an IF axis of length 2. Thus, you must set `nif=2`. IF axis location 2 for the first scan will get filled with dummy visibilities of zeros.

  Note, that if you set `freqsel=8300,4600,8000` instead, and `ifmap=-1`, then the 8300 frequency would be put onto IF axis location 1. However, if `ifmap=1` then it would go to location 2, and location 1 would be zero filled instead.

  ⇒ Generally, I suggest you set `nif=0`.

- If you plan on loading a very large visibility data base, then it may be advisable to set the data compression adverb, `douvcomp=1`. However, note that you sacrifice some dynamic range (each real and imaginary part of the correlation is stored as a 16 bit integer rather than a 32 bit floating point value) in doing this, so it's a trade off. If your data suffer from severe interference, then loading them straightforwardly with data compression would compromise the good data, as all the dynamic range would be expended in coping with the interference. However, each visibility is compressed independently, so if a visibility has strong intereference and you compress it, you only sacrifice dynamic range in that visibility, not all the others as well.

  ⇒ For continuum work you should set `douvcomp=-1`. For spectral-line work you must make your own decision.

6. *The APARM array*

- `aparm(1)` controls an important part of the procedure which interacts with a number of the other `ATLOD` inputs. This is what form, if any, the correlations are to be converted to. That is, are they to be converted to Stokes parameters (see § 3.2), or should they be left in their original form.

  Note that a full polarimetric calibration of ATCA data cannot be performed with *AIPS*; you must do this with *MIRIAD* (see the *MIRIAD* User's Guide).

  Recall that raw ATCA correlations (visibilities) are in terms of linear polarizations, and that our ultimate goal is to convert them to the Stokes parameters, I, Q, U, and V which we can then image. A nominal conversion to Stokes parameters is achieved if you put `aparm(1)=1`.

  This nominal Stokes conversion makes three assumptions (the need for such assumptions is obviated by a correct calibration in *MIRIAD*).

  (a) It assumes that the X and Y gain amplitudes were equal when observing. Such an equalization is generally a part of the observing set-up chores (ATCA users will be familiar with the program `CACAL` and earler `DELCOR` that does this) so that this should be a good assumption. ATCA data from the first year or two of operation may not have equalized gains. Any mismatch causes Stokes I, Q, and U to be corrupted by a term approximately equal to

$$I \times (\frac{\delta a_X}{a_X})_i + (\frac{\delta a_Y}{a_Y})_i + (\frac{\delta a_X}{a_X})_j + (\frac{\delta a_Y}{a_Y})_j \cdot$$

where δa is the error in the amplitude of the gain a (for the X and Y feeds, and i and j are the antennas involved in the interferometer). This is a first-order effect.

(b) It assumes that the feeds are perfectly linearly polarized. Errors in this assumption cause an unpolarized source to appear polarized by an amount given by the instrumental polarization. This is about 3%. It affects I only as a second order effect.

(c) It also assumes that the Φ_{XY} measurements are good (see § 3.5). The quality of these measurements is quite good now, although you should still inspect them first (with the `optype='sysc'` and `pltsys` combination) rather than use them blindly. They are generally worse at 13 and 20 cm than at 3 and 6 cm.

If you don't convert to Stokes parameters, then you must calibrate the XX and YY (remember $2I = XX + YY$) polarizations separately under the assumption that your calibrators are not linearly polarized (see § 3.3).

If you set `(aparm(1)=-1)`, then no Stokes conversion is performed, but *AIPS* is tricked into believing that the linear polarizations are circular (you could also use `PUTHEAD` on the header to effect this trickery if you don't do it with `ATLOD`). It does not affect the actual calibration in any deleterious way. Leaving `aparm(1)=0` means load the polarizations just as they are and label them as linear; I do not recommend doing this.

⇒ I suggest you set `aparm(1)=-1`, thus loading the correlations in their natural form, whilst labelling them as circular. You should use *MIRIAD* to make the best possible calibration, whether you are interested in Stokes I only or all Stokes parameters.

- If you set `aparm(2)=1` then the data that the on-line system flagged as bad are loaded to your *AIPS* file. You should have a good reason for doing this.

  ⇒ Generally I suggest you set `aparm(2)=0`.

- Auto-correlations are dropped by default. Put `aparm(3)=1` to retain these. It is sometimes useful to retain the auto-correlations, as Φ_{XY} is the phase of the auto-correlation between the X and the Y feeds. The Φ_{XY} that is used for the Stokes conversion is an average across the central part of the band, and is stored in a different place. If you were particularly conscientious, you might care to examine Φ_{XY} as a function of frequency by examining the relevant auto-correlation.

  ATCA data with 128 MHz bandwidth and one IF taken after February 1991 contain a full set of meaningful auto-correlations. 128 MHz bandwidth data from earlier periods do contain the XY correlations, but they are not labelled in an obvious fashion. Please consult with me should you wish to extract them from the older style data. Spectral line data do not contain auto-correlations, so `aparm(3)` is unimportant in this case. 128 MHz data with 2 IFs do not contain auto-correlations as there is not enough correlator to form them.

  ⇒ Generally I suggest you set `aparm(3)=0`.

- For a compact array like the ATCA, the problem of shadowing may arise. That is, one antenna may be blocked by another when the source is low and of declination close to the latitude. `ATLOD` will drop shadowed antennas by default. If you want to change the shadowing criterion, then set `aparm(4)` in metres. An ATCA dish is 22 m in diameter. To relax the shadowing criterion, make `aparm (4)` smaller. Note that the shadowing calculation assumes that the source is at the pointing centre. For wide-field imaging, you might keep in mind that a source far from the pointing centre will be shadowed slightly differently.

  ⇒ Initial tests indicate that at 20 cm, cross-talk begins before geometric shadowing. At 6 cm, it appears to occur soon after the onset of geometric shadowing. I suggest you set `aparm(4)=0` at 3 and 6 cm, and perhaps `aparm(4)=25` at 13 and 20 cm if you are feeling conservative.

- If you load your data in time order, `ATLOD` will create the index (NX) table (see Appendix C). You have control over when index records are written. Basically, you want to start a new index record when there are no data for some period, and after data have been acquired for some contiguous time. The defaults of `aparm(5)=10` and `aparm(6)=60` minutes should be adequate.

  ⇒ Generally I suggest you set `aparm(5)=0` and `aparm(6)=0`.

- ATLOD will also build the pristine calibration (CL) table if you load the data in time order. There is a CL table entry as often as you want interpolated gain solutions (see Appendix C). The default of 3 minutes (aparm(7)) should be acceptable for most situations. Note, however, that the CL table entry time interval sets the time scale on which you can self-calibrate a multi-source data base. You can however, split a source into a single-source data base, whereupon the fundamental integration time of the observation (generally 15 seconds for ATCA data) sets the minimum possible time scale for self-calibration.

  $\Rightarrow$ Generally I suggest you set aparm(7)=0.

- An on-line correction for the system temperature is generally made (although you can turn it off when observing). This is used to scale the visibilities as the correlation coefficient gives the ratio of correlated power to uncorrelated (noise) power. However, the hardware that measures the system temperature is not ideal, so that this measurement is noisy, which contributes to the noise in your final image. aparm(8) allows you to average the T_{sys} measurements for a few integrations so that you apply less noisy T_{sys} measurements. If the on-line correction was already applied, this is first undone before application of the averaged correction.

  Set aparm(8) to the number of measurements that you wish to average (I would suggest 10 or so). If you leave aparm(8) at zero (or unity), then no additional T_{sys} scaling is done. If you set aparm(8)=−1 then the on-line T_{sys} correction is undone, and redone with an assumed T_{sys} of 50 K. This is equivalent to not turning on the on-line correction.

  $\Rightarrow$ Because the improvement is small, and it slows ATLOD down, I suggest that in general you leave aparm(8)=0.

- If you require the on-line Φ_{XY} values (see discussion of aparm(1), cparm(4) and § 3.5) for Stokes conversion or for application to the Y gains (neither of which is recommended) then you may consider the use of aparm(9). Even if the current hardware allowed us to make a perfectly accurate measurement of Φ_{XY}, it would still be a measurement of a noisy quantity. aparm(9) specifies how many measurements should be averaged together to form the applied Φ_{XY}. If using this option, I would suggest averaging over 5 or so points. Leave it at zero to use the on-line values without any averaging.

  When this averaging option is active, wildly discrepant Φ_{XY} values are replaced by the running mean of the current stack of aparm(9) measurements. A point is considered discrepant if it is more than a few times the width of the current stack from the stack mean.

  $\Rightarrow$ Because we do not recommend general application of the on-line Φ_{XY} values, and because we now prefer all application of Φ_{XY} to be in *MIRIAD*, I do not recommend that you use this option. Set aparm(9)=0.

- Dropouts in the visibility data generally occur in both the cross- and auto-correlations simultaneously. Thus, we can use the Φ_{XY} measurements (an auto-correlation) as a monitor and discard cross-correlations when the auto-correlation is obviously discrepant. If aparm(10)=0 then no flagging based on the quality of Φ_{XY} values is done.

  We can detect discrepant Φ_{XY} values in two ways. First, if aparm(9)>1 then the current Φ_{XY} can be compared to the mean of the previous aparm(9) measurements. Second, if you are not using the on-line measurements of Φ_{XY} (as recommended) directly but have input mean values into the xyphase array then a Φ_{XY} can be noted as bad if it is discrepant from the value in the xyphase array by more than some value.

  In the first case, you should have set aparm(9)>1 so that a stack of aparm(9) Φ_{XY} values can be accumulated for comparison with the current value. If aparm(10) >0 then visibilities associated with the antennas with the discrepant Φ_{XY} are dropped. If aparm(10)<0, then in addition to ATLOD's own internal criteria a point is not treated as discrepant unless it is abs(aparm(10)) degrees different from the running mean of the current stack of aparm(9) measurements. You need to inspect the Φ_{XY} values via the optype='sysc' and pltsys combination to be able to set this value meaningfully.

  In the second case, you should have set aparm(9)=0. Then, if you set aparm(10) <0, a Φ_{XY} will be deemed bad if it is more than abs(aparm(10)) degrees from the relevant value in the xyphase array. Visibilities associated with the bad antenna will then be dropped.

  $\Rightarrow$ This *is* a useful editing option. However, because we do not usually recommend direct use of the on-line Φ_{XY} values, I suggest you only use it only if you input the mean Φ_{XY} values into the xyphase array and set aparm(10)<0 (and set cparm(4)=1 indicating use of the xyphase array).

7. *The BPARM array*

   - The `bparm` array allows you to tell `ATLOD` not to load specific baselines or antennas. You might have *a priori* knowledge that specific ones are bad and not flagged by the on-line system. You would want to be very certain of this before doing it, as every baseline is precious. See the `HELP` file for details.

     ⇒ Generally I suggest you set `bparm=0`.

8. *The CPARM array*

   - `cparm(1)` allows you to print out some useful information on the workstation screen.

     If `cparm(1)=1` then some of the system calibration group information is listed. You get the system temperatures, the Φ_{XY} values and the parallactic angle. You get the chance to stop the listing after a while (answer 'Q' when prompted) and continue loading the data. Control is then returned to *AIPS*. This is meant for a quick inspection rather than an exhaustive listing. Use option `optype= 'sysc'` and examine the output text files for more extensive examination.

     If `cparm(1)=2`, then you are informed when `ATLOD` detects that the Φ_{XY} values or the system temperatures have jumped. By this, I mean jumped to a new value and stayed there, not just a dropout. `ATLOD` accumulates its averages for the Φ_{XY} for each frequency, and the system temperatures for each frequency and source. It resets its accumulations if one of these accumulations jumps unexpectedly. This option is active only if `aparm(8)` or `aparm(9)` is greater than 1.

     If `cparm(1)=3`, both the previous two options above are active.

     ⇒ Generally I suggest you set `cparm(1)=0`.

   - `cparm(2)` should be left at zero.

   - If `cparm(3)=1` then the visibility phases are negated. This can be useful for some data taken in 1990. Refer to the communiqué issued on the Epping bulletin board to see if any old data that you have needs this treatement.

     ⇒ Generally I suggest you set `cparm(3)=0`.

   - If `cparm(4)=1`, then the Φ_{XY} values in the `xyphase` array are used rather than the on-line measurements.

     ⇒ I suggest you only do this if you wish to use the Φ_{XY} editing option (see `aparm(10)`).

   - Φ_{XY} can be thought of as the phase part of the gain for either the X or Y feeds. Our convention is that they are allocated to the Y gains. If `cparm(5)=1` and you have *not* asked for Stokes conversion, then the Φ_{XY} values are allocated to the Y gains which are then applied to the visibilities. There are two reasons for doing this.

     First, it causes the XX and YY visibilities to be in phase. Thus, you could ask an *AIPS* task to form Stokes I by summing XX and YY before calibration. This would allow you to edit on only one quantity, instead of doing it for XX and YY separately. However, generally the correct observing setup done on-line will ensure that XX and YY are already roughly in phase anyway.

     The second reason is now deprecated. We used to recommend that for those of you who were going to calibrate in *MIRIAD* that you apply the Φ_{XY} values in `ATLOD`. However, our preferred path now is to do this in *MIRIAD*.

     ⇒ I suggest you set `cparm(5)=0`.

   - `cparm(6)` offers the prospect of either listing (`cparm(6)=1`), deleting (`cparm(6)=2`), listing and deleting (`cparm(6)=3`), and correcting (`cparm(6)=4`) data for which the sampler statistics are not at their optimal values. When these statistics are bad, it is likely that either something is wrong and the data are useless (such as strong interference for which there was insufficient range in the attenuators to bring the sampler statistics into the servo loop) or that the attenuators were slightly wrong as the servo loop bought them back into range for a source of significantly different flux density from the previous one; basically, it means that the visibilities have been scaled with the wrong factor.

     If you choose the option to correct the visibilities when the sampler statistics are wrong, then this works only for small errors ($< 10\%$), and data with larger errors in the statistics are dropped.

     Note that since the sampler statistics are antenna based, all visibilities involving the bad antenna are dropped.

It has been noted in some data prior to March 1992 that the sampler statistics were not getting updated every integration. This means that the off-line correction would make the wrong correction for some visibilities. To assess whether you have this problem or not, you should use the `optype='sysc'` option with `cparm(9)=2` or 3 (see below) to create text files containing the sampler statistics which you can then plot with utility program `pltsys`. Look for successively duplicated entries.

⇒ For data prior to March 1992, I suggest you set `cparm(8)=1` (just providing you with warnings). For more recent data I suggest you put `cparm(6)=4` and correct the data.

- If `cparm(6)` > 0 but you want to use a tolerance different from that of `ATLOD` for the tolerable error in the sampler statistics, then set `cparm(7)` to your desired value. `ATLOD` marks an antenna as bad if the sampler statistics are more than 5% from the correct value (`cparm(6)=1,2` or 3), and uncorrectable if they are more than 10% from the correct value (`cparm(6)=4`).

  ⇒ Generally I suggest you set `cparm(7)=0`.

- If `cparm(8)=1` then the file specified by the `infile` adverb is converted to lower case before being used. This is intended for use at Narrabri where VAX disks are mounted on the SUNs via multinet.

  ⇒ Generally I suggest you set `cparm(8)=0`.

- When you set `optype='sysc'`, `ATLOD` writes useful sytem calibration information into text files in the FITS area. By default, you get the Φ_{XY}s, and the X and Y system temperatures. If you set `cparm(9)=2` then the text files contain the X sampler statistics (one file for each of the low, mean and high sampler measurements). Similarly, `cparm(9)=3` gets you the Y sampler statistics.

  ⇒ Generally I suggest you set `cparm(9)=0` unless you are interested in the samplers.

- If you set `cparm(10)>0`, then the system temperature for IF 1 will be applied to IF 2. This is a fudge for data taken in a period of time when IF 2 did not have its own measurement of the system temperature.

  ⇒ Generally, I suggest you set `cparm(10)=0`.

9. *The XYPHASE array*

   This adverb is offered for you to enter one number per antenna and frequency for the Φ_{XY} values that you choose from the `pltsys` plots if you require them. We no longer recommend application of Φ_{XY} in `ATLOD`. However, the Φ_{XY} editing option (see `aparm(10)`) is still useful, and this is the only case now for which you should enter values into this array.

   You must enter one phase difference (in degrees) for each of the *six* antennas in the array, regardless of how many antennas were in your observation. Naturally, the values that you insert for antennas that you do not have are arbitrary. If you have observed at more than one frequency (e.g., 3 and 6 cm), then you must enter these values for *all* frequencies. First, you enter the six numbers for the first frequency observed, and then the six numbers for the second frequency observed, and so on. They *must* be in the correct frequency order as encountered and loaded in the observation. You could check this by listing a bit of your data (`optype='list'`) and seeing what order the frequencies arrive in.

   ⇒ Except for Φ_{XY} editing, I suggest you set `xyphase=0`.

10. *The DPARM array*

    - There was a problem with some data for a short period of time where the sources were mislabelled on-line. If you set `dparm(1)=1` then all sources in the scan will be labelled as the first source in the scan based RPFITS source table. This is appropriate only for data containing one source per scan (i.e., not mosaicing data).

      ⇒ Generally, I suggest you set `dparm(1)=0`.

    - If `dparm(2)=1` then each spectrum is three-point Hanning smoothed. The usual reason for Hanning smoothing spectral-line data is to reduce ringing around narrow spectral features. The ringing occurs because the weighting function in the lag domain is a top-hat function so that the frequency spectrum is convolved by a narrow sinC function. If you do this then you should probably consider dropping every other channel (with adverb `chansel`) as adjacent channels will no longer be independent.

      Of course, at 128 MHz bandwidth, adjacent channels are already correlated so this option is only meant for spectral-line data.

      ⇒ For 128 MHz bandwidth data you should set `dparm(2)= 0`. For narrower bandwidths, this option should be considered.

4.5 Some Examples of using ATLOD

In the following examples, any adverb that is not listed should be set to zero (for numeric adverbs) or an obvious default (such as bcount = 1 for first scan to load) or blank (for string adverbs). ATLOD is of course activated with the GO ATLOD command.

1. Extract from 10 files on tape drive number 2 the Φ_{XY} and T_{sys} values for all frequencies and sources to text files in the DATA/FITS areas. These files will be called XYPHS_UID, TSYSX_UID, and TSYSY_UID where 'UID' is your *AIPS* user number.

ATLOD	
intape=2	Load data from tape drive 2
nfiles=10	Load 10 files
optype='sysc'	Extract SYSCAL infomration

   You can then plot the contents of these text files with the utility program pltsys. Just enter pltsys from the Unix command level and answer the questions. It can select frequencies so you can just load them all with ATLOD. It plots on a PGPLOT device.

2. Load 3 files from tape drive number 1 starting at the second file with sampler corrections activated. The three RPFITS files are all put into the same output *AIPS* file by ATLOD.

ATLOD	
intape=1	Load data from tape drive 1
nskip=1	Begin loading at the second file
nfiles=3	Load 3 files
outname='zeeman'	Specify output file or default
outclass='uv'	is source name. When appending
outseq=1	data, must specify fully
outdisk=3	Choose *local* disk with most space
optype='load'	Load the data
ifmap=−1	Allow ATLOD to arrange IF axis
douvcomp=−1	No data compression
aparm(1)=−1	Leave as linear polarizations
	but call them circulars
cparm(6)=4	Correct data with bad sampler statistics

3. Load a file from disk, appending to an existing *AIPS* file, and activate sampler corrections and Φ_{XY} flagging. The values in the xyphase array should be found from plotting up the text files generated with the optype='sysc' option (see example 1 above). In addition, by examining these plots, it was determined that any point more than 10 degrees from the mean is bad and these data are dropped. Finally, it is assumed that there are two IFs in these data.

ATLOD	
infile='myarea:bigfile'	Specify disk file
outname='1934-638'	Specify existing
outclass='uv'	file for appending
outseq=1	
outdisk=4	
optype='load'	Load the data
ifmap=−1	Allow ATLOD to arrange IF axis
douvcomp=−1	No data compression
aparm(1)=−1	Label linear polarizations as circular
aparm(10)=−10	Discard data when Φ_{XY} discrepant
	from values in xyphase by 10 degrees
cparm(4)=1	Tell ATLOD to look in xyphase array
cparm(6)=4	Correct data with bad sampler statistics
xyphase=−5.0,10.2,3.4,−0.2,1.2,12.3,	Φ_{XY} values for IF 1
−2.1,0.3,−10.2,8.9,5.4,−4.7	Φ_{XY} values for IF 2

4. Consider an observation where the first IF has been set to observe in 4-MHz, 513-channel, 2-polarization mode and IF 2 has been set to observe in 128-MHz, 33-channel, 4-polarization mode. These two IFs cannot go into the same *AIPS* file because they have different numbers of polarizations and different numbers of channels. Thus, you must make two passes with ATLOD, selecting the two IFs separately. In the following example, we select just the 8 MHz data (IF 1), reading 5 files from tape drive number 1, activating sampler corrections, data compression, Hanning smoothing and dropping every other channel. ATLOD is allowed to choose the name of the output file.

ATLOD	
intape=1	Read from tape drive number 1
nfiles=5	Read five files
outdisk=4	Specify *local* output disk
optype='load'	Load the data
ifmap=−1	Allow ATLOD to arrange IF axis
ifsel=1	Select IF 1 only
chansel=1,513,2	Select every other channel after Hanning
douvcomp=1	Activate data compression
aparm(1)=−1	Leave as linear polarizations
	but call them circulars
cparm(6)=4	Correct data with bad sampler statistics
dparm(2)=1	Hanning smooth spectra

5. The *AIPS* verb UCAT will show the new file in the *AIPS* directory (termed a catalogue). Now, use the verb GETNAME to select the new file (e.g. GETNAME 1 to get the file located in catalogue slot 1 on the current indisk), and examine its header with the command IMHEAD. After the header information, you should see that there are five extension files associated with the multi-source file (see Appendix C for descriptions of the types of extension files you will encounter). There *must* be an antenna (AN) file, an SU (source) table, and an FQ (frequency) table. Never delete these three tables, as there is no simple way to regenerate them short of re-running ATLOD. You should also see the index (NX) and initial calibration (CL) tables. If the NX and CL tables are not present, probably your data were not loaded in time order and need to be sorted. ATLOD should have informed you if this was the case. The NX and CL tables can then be created with INDXR (see § 5).

6. Most of the messages that ATLOD writes to the terminal also get put into the history file for inspection at your leisure.

4.6 Show and Tell

AIPS has a facility by which you can change the adverbs of some tasks while they are running. To see which ones you can change, for ATLOD for example, enter the command SHOW ATLOD. You can then change their values and convey the new values to ATLOD with the command TELL ATLOD.

Perhaps the most useful thing is you can tell ATLOD to terminate gracefully at some point rather than being forced to abort it via ABORT ATLOD. If you aborted it, the file it is writing would be compromised if the I/O buffers have not been flushed to disk. As an example, you may have told ATLOD to load 20 files from tape, but you realize part way through that the 20th file is something you don't want. You could change nfiles to 19 while ATLOD is running.

Last update : 27/11/93

5 INITIALIZATION OF THE MULTI-SOURCE FILE

Before you become buried in an avalanche of adverbs, I remind you of Appendix B, which explains some common adverbs that occur in the calibration tasks and procedures.

5.1 Sorting and Indexing

1. Hopefully, you used ATLOD so that it loaded your data in time (and possibly baseline order). The time order is essential, but not the baseline order. Use IMHEAD and see if the visibility file header shows TB order. If not, then you must sort the data. Use the task UVSRT and set the adverb `sort='tb'`. The input and output file names can be the same in UVSRT if you are running short of disk space. But if the machine crashes part way through, you will have to reload your data with ATLOD. Note that UVSRT creates two scratch files as well as the output file, so that you need plenty of space anyway.

   Make sure that you don't allow UVSRT to access any network mounted disks (see § 2.3.2) by setting baddisk (control of scratch files) and outdisk (control of disk for sorted data base).

UVSRT	
sort='tb'	Sort into time-baseline order
outdisk	Use these to avoid putting output
baddisk	on NFS mounted disk

2. If you needed to run UVSRT you must also run INDXR to create the initial index (NX) and calibration (CL) tables (ATLOD usually creates these for you). The defaults for INDXR with regard to the NX table should be adequate, but make sure of this. For the CL table, you need to specify the interval at which entries are to be written. The default of 5 minutes is generally adequate (i.e., significant gain variations don't occur much faster than this). More detail on CL tables will be provided later. This is all you need to know for now.

INDXR	
cparm=0	Most of the defaults OK
cparm(3)=3	Gain table entries every 3 mins

5.2 Summarizing the Observation

Make a hardcopy (`docrt=-1`) of the summary of the observation with the task LISTR by setting the adverb `optype='scan'`. This is a very useful reference sheet and contains details of the times of each scan, the frequencies observed, the number of visibilities, and the source information in the SU table. LISTR can list all sorts of things about your data (including the data) in a variety of ways.

LISTR	
inname	Select multi-source file
inclass	for the initial
inseq	summary listing
indisk	
optype= 'scan'	Scan summary
docrt=-1	Print on line printer

5.3 Setting the Flux Density Scale

When you set up your observation you should have used CACAL (or DELCOR in olden times) to make the initial on-line calibration. This generally involves setting the flux density scale via 1934 – 638 In the off-line calibration procedure, we essentially redo that step, although you could proceed with out it if you are confident you got it right on line.

We continue by establishing the flux density scale of the primary calibrator. Generally, this calibrator will be 1934 – 638, although possibly it might be one of the northern sources 3*C*286 and 3*C*48. You need to enter its flux density at all observed frequencies into the source (SU) table with SETJY(you can print the contents of the SU table with PRTAB). Later it will be used to determine the secondary calibrator flux densities, which will in turn be

used to calibrate the program sources. There are very few sources with constant flux density and this is the reason for the leap-frogging technique.

You can either enter the correct flux density by hand, or allow SETJY to work it out for any of the above sources. First, I will discuss manual setting of the primary flux density.

1.
   - Make sure optype is blank.
   - Fill in the source name of the primary calibrator.
   - Assign the flux density of the primary to the first element of the zerosp adverb with something like zerosp=10.0,0 (this is also the correct thing to if you didn't convert to Stokes and have tricked *AIPS* into thinking you have circular polarizations). zerosp has room for only one source at a time, so don't give SETJY a list of sources. zerosp allows you to specify a model of the polarized flux density in a source as well, but we won't use this capability.
   - Here also is your chance to assign a calcode adverb for the primary calibrator source (see Appendix B). You can make this anything you like for easy selection of your calibrators. For example, you might use calcode='P' for primary calibrators, calcode='S' for secondary calibrators (although you will leave their flux density at zero for the moment), and calcode='BP' for bandpass calibrators (you can choose any character code you like, they fon't have to be these ones). Later on, you can select all calibrator sources which have non-blank calcodes by putting calcode='*'. Don't set this for your program sources (i.e., any non-calibrator sources). Similarly, you can select all the claibrators that have a common calcode by setting, say, calcode='S' for the secondary calibrators. It is just a way to avoid typing in the a long list of source names over and over. See HELP calcode for more informatio I recommend making use of this adverb in this way.
   - You will see that freqid is an input to SETJY. However, it does not behave as you might expect. freqid *does not* have any effect on source selection in SETJY, and you should ignore it for this particular application of SETJY (I will describe what it does do later). There is no space in the SU table for the freqid. Thus, when a source has the same name, but a different freqid, you would be unable to set the source flux differently for the two freqids. This is a design limitation.
   - To deal with the above problem, ATLOD uses the qual source selection adverb, and gives it the freqid as its value. As double insurance, ATLOD also appends the freqid to the source name if it detects any name/frequency ambiguity. The value of qual for different sources can be seen with the LISTR scan summary output discussed above, or by printing the SU table with PRTAB. If you only have one qualifier, or no source name ambiguity, set qual=−1, which means any qualifier. In addition, you may have appended strings such as '_A' and '_B' to the source names for different frequencies to deal with this ambiguity.
   - Somewhat different to freqid is the IF axis. If you have more than one simultaneous frequency (i.e., the IF axis is of length greater than 1), you should select the IF with bif and eif, as the flux density of the primary is likely to vary significantly between the frequencies designated by the IF axis. The combination of freqid (which specifies a group of simultaneous frequencies) and the location on the IF axis (which specifies which of the group you are interested in) fully specifies the frequency.

SETJY	
sources='1934 − 638',' '	Select primary calibrator
zerosp=6.33,0	Set flux density
optype=' '	Use specified source parameters
calcode='p'	Make 'p' the primary calibrator code
qual=−1	Means any qualifier or
qual=2	do each qualifier separately
bif=1	Select range of IFs to set
eif=1	
freqid=0	Not relevant here

2. If you would like SETJY to enter the correct flux density into the SU table for you automatically, put optype='calc'. Fill in the source name as before (note that as long as 1934 is the first 4 characters of the name SETJY will recognize it as 1934 − 638). You *must* specify the freqid correctly for your source because this is the only way SETJY can know what frequency to use when calculating the flux density. As before, freqid *does not* interact with the source selection criteria.

   Recently, the spectrum of our primary calibrator has been more accurately determined (J. Reynolds, private communication). Previously, we relied on a very old Parkes spectrum augmented by a MOST point. The

new spectrum differs from the old by some 5% at cm wavelengths. If you would like to use the most recent determination of 1934 − 638 then leave `aparm(2)=0`. If you are combining new data with older previously calibrated data (using a flux density scale set with the old spectrum), then set `aparm(2)=1`.

SETJY	
sources='1934 − 638',' '	Select primary calibrator
zerosp=0	Allow program to work
optype='calc'	out flux density
calcode='p'	Make 'p' the primary calibrator code
qual=−1	Means any qualifier or
qual=2	do each qualifier separately
bif=1	Select range of IFs to set
eif=1	
freqid=2	Must specify
aparm(3)=0	New polynomial coefficients

3. For secondary calibrators, you should not enter any flux densities, but you might like to give them calibrator codes such as `calc='s'` and so on (see above).

4. `SETJY` can also reset information in the SU table. For example, if you put `optype='rese'` then the flux densities for the selected sources are zeroed. You might want to do this if the boot-strapping procedure that evaluates the secondary calibrator fluxes goes wrong. See the `HELP` file for more details.

5. After finishing with `SETJY` you may wish to re-summarize the observation (§ 5.2) to check the new source information. Alternatively, you could print the SU table directly with the task `PRTAB`.

5.4 Setting the Velocity Scale

If you are doing spectral-line work, you will, at some stage, need to set the velocity scale. Refer to § 13 for some discussion on velocity types and velocity reference frames. You will need to know whether you want the 'radio' of 'optical' velocity definition, and whether you want the LSR or heliocentric (barycentric really) reference frame. The relevant velocity information is entered into the SU (source) table with `SETJY`. You can specify different values for different sources. I will dicuss here only those adverbs specific to velocities.

Many spectral-line users will end up running a task called `CVEL` (see § 13 which makes Doppler corrections (the ATCA does not use on-line Doppler tracking) to the spectra. Optionally, you can enter ther velocity information at that time, rather than now.

- Specify the source name with `sources`. Naturally you are only interested in your program source(s), not any of your calibrators.
- Specify the velocity (km s^{-1}) of the source with `sysvel` and the channel that this velocity refers to with `aparm(1)` (see discussion in § 13.5). You should know this from when you computed your central observing frequency when you observed. If you look in the SU table with `PRTAB`, you will see that the velocity has been recomputed to refer to the reference pixel. Make sure it did it correctly.
- Indicate which velocity reference frame the above velocity is measured in with `veltyp`. *AIPS* offers LSR (for Galactic work usually) and Heliocentric (for extra-galactic work usually) only.
- Indicate whether the above velocity was measured in the 'radio' or 'optical' definition with `veldef`.
- For `SETJY` to be able to work out the width of one channel in the appropriate velocity type, you must specify the rest-frequency (Hz) of the spectral-line that you are observing with `restfreq`, and the `freqid` so that the correct observing frequency can be selected. If somehow you have managed to create a spectral-line data base with an IF axis of length other than unity, you must also specify the correct IF with `bif` and `eif`. Otherwise leave these at 0. Note that you should use the sum of `restfreq(1)` and `restfreq(2)` to enter the rest-frequency; see the example below.

SETJY	
sources='NGC9434',' '	Select program source
optype=' '	Use specified source parameters
freqid=1	Select observed frequency
bif=1	with this combination
eif=1	
sysvel=5353	Specify velocity at
aparm(1)=257	this channel in
veltyp='helio' or 'lsr'	this reference frame and
veldef='radio' or 'optical'	this velocity definition
restfreq(1)=1667.0E6	Rest frequency of line
restfreq(2)=0.359E6	Rest frequency of line

Use **PRTAB** after running **SETJY** to ensure the parameters you have entered have been correctly transferred to the SU table.

Last update : 27/11/93

6 AVERAGING DATA IN FREQUENCY AND TIME

The order of this section, and the one on editing (§ 7) are interchangeable. If you decide that you need to average your data (be it in frequency or time), you may need to edit it first. For example, there is no point to averaging in frequency if there is channel specific interference that would result in a corrupted averaged channel (such interference is not uncommon in the 13 and 20 cm bands). You would need to edit out the bad data first. Therefore, I suggest you read this section, and the section on editing first, and then proceed with whatever suits your needs.

6.1 Should I average my continuum data in frequency ?

Now you should make a fairly fundamental decision. Do you want to retain the spectral information across each band for each frequency or would you prefer to average all the channels in each band to one channel. The answer to this question is that you will *always* need to average the data in frequency. Why bother asking the question then ? Well, you must always average in frequency for the initial determination of the antenna gains with time from your calibrators (§ 8), but you may want to retain the spectral information on your program sources for multi-frequency synthesis. I will address this latter issue here.

If you are only interested in continuum information, then you may want to do this averaging. However, you must estimate the advantage in terms of the (u, v) coverage that you gain by retaining the spectral information and gridding all channels together in the imaging task (this is multi-frequency synthesis) against the reduction in the amount of data you need to handle and degradation of the image that occurs when you average channels together. The latter effect is known as bandwidth smearing and is essentially a chromatic aberration. The effect is a radial smearing of the source with a reduced amplitude and increases with distance from the phase centre. It is discussed in detail in Appendix D. Note that calibrators are not affected by bandwidth smearing because they are at the phase centre.

The improvement in the radial (u, v) coverage by using the full bandwidth is easily estimated from an expression such as

$$\Delta u = u_0 \times \frac{\Delta\nu}{\nu_0},$$

where $u_0 = D$ is the baseline in wavelengths at the band centre (frequency ν_0), and $\Delta\nu$ is the bandwidth. Note that for a given observation, the most additional coverage is gained by the longest baselines. With the compact array, it is possible to fill in some of the inter-spacing gaps in this fashion (and also by rapid frequency switching) and this is a significant advantage for complicated sources. At 13 and 20 cm, there is often significant benefit to be gained by retaining all the channels, especially in the 6 km array. At 3 cm it is very unlikely you would retain all channels, and at 6 cm the benefit is marginal.

If you conclude that there is no benefit to be gained with multi-frequency synthesis for your data, there is an additional point to consider before you average some channels together. This concerns interference, which is often troublesome in the 13 and 20 cm bands. You may want to edit the data first (see § 7) and remove channel-specific interference from the data, and then average the good channels. If you need to consider this option, read and apply the section on editing first, and then come back to averaging of the data later.

6.2 How do I average my data in frequency ?

1. The task that averages channels together is called `AVSPC` and it will create a new data base. Thus, you can have two data bases, one with all the spectral information (should you conclude there is benefit in retaining it) and one without. The latter is always used for gain determination with time (see § 8).

   The new channel-averaged data base is often referred to as the channel 0 data base, and I will refer to the `AVSPC` output in this way henceforth. `AVSPC` will copy across the relevant extension files that are attached to the input file. It is wise not to include all channels in this average. The first and last few are likely to contain rubbish as the band response falls off fairly fast at the band edges. There is not much point to averaging in noise with your signal. The default for `AVSPC` is to take the central 75% of the band, and this should be adequate; experience with ATCA data indicates that a good channel range for 128 MHz data is 7 to 27. For other bandwidths, you must make your own assessment (see below), as the band edges roll off at different rates from bandwidth to bandwidth.

2. A good way to examine the spectrum to determine the useful channel range is with the task POSSM. This task enables all sorts of plots of your data as a function of channel, something most of the other plot routines cannot do. For this purpose, select your strongest calibrator source; probably the primary would be best, even if you just have a short observation. POSSM has a lot of inputs, but you need only change a couple from their default values. Select the desired sources with the combination of the source, calc and freqid adverbs (and maybe the other frequency selection parameters).

   Setting POSSM in motion at this point will produce a scalar averaged plot from all the visibilities. This means working out the amplitudes for each visibility and then averaging them, rather than averaging the real and imaginary parts of all the visibilities which is called a vector average. The phases are always computed with a vector average because of the circular nature of phase. The scalar averaged spectrum contains contributions from emission in the whole primary beam. Any interference will be obvious in this plot.

   On the other hand, the vector average aparm(1)=1 can be very useful for supressing interference. If the phase stability was good and as is usual, you did the basic on-line calibration with a strong calibrator (on-line programs CACAL or DELCOR), then the phases on all baselines would have been set to zero and the vector sum would be meaningful. Any interference generally has semi-random phase so it tends to average out. The vector average will make a spectrum at one location on the sky (it is basically a spectral-line cube with one spatial pixel) so make sure you set offset to point at the piece of sky you are interested in. Alternatively you could restrict the average to the short baselines so that the exact values of offset are less critical.

   You can display this plot directly on the TV (set dotv=1 in POSSM), or display the resultant plot file (created if dotv=−1) with TKPL (plot on Tektronix or SUN Tektronix emulator), MIXPL (display on terminals or laser printers), or TVPL (display on a TV), and then you should have a fairly reasonable idea of which channels to keep and which channels to throw away. Note also that displaying on the SUN's Tektronix emulator with TKPL is *much* faster than displaying on the SUN's TV with TVPL or directly on the TV with dotv = 1. However, for small plots, such as those produced by POSSM, it makes little difference, most of the time going into starting the task up rather than executing it.

   The 'divide by channel 0' (POSSM will just use the central 75% of the band for this) option (bparm(1)=1) might also prove handy. This option divides the frequency dependent visibilities by the channel 0 visibilities. This removes most of the calibration errors as well as most of the source structure and turns the source into a pseudo point source (you could then use something other than a point source for these plots). The remaining response across the band is because of the filters and electronics. Experiment with this option, and see how much it affects the outcome. Make sure there is no severely corrupted data in the channel 0 data channels if you use this option.

   Note that channel 1 in ATCA spectra contains the zero frequency or DC term. This is why there are 33 channels in 128 MHz bandwidth spectra, rather than 32 as you would expect. This is very useful in determining if your data contain DC offsets, a problem discussed more below.

POSSM	
source='1934 − 638',' '	Select source by name or perhaps
calcode='p'	select the primary with its calcode if you set it in SETJY
freqid=1	select desired frequency
antennas=0	average over all antennas
docalib=−1	no calibration to apply
aparm=0	scalar average
aparm(1)=1	vector average
bparm=0	
dotv=1	plot directly on TV or
dotv=−1	make a plot file for display later

MIXPL	
outfile=' '	Don't write plot to disk file
invers=0	Plot highest version plot file
device=0	Laser printer
device=10	Visual 603 terminal

3. Before setting you loose on AVSPC, it is worth making a diversion here to discuss a problem (and its cure) that arose largely in 128 MHz bandwidth ATCA data taken in 1990. This is the occurrence of DC offsets in the correlation. The correlator is a Fourier transform device. This means that it measures a time-lag spectrum, which is then Fourier transformed to the frequency domain. The natural weighting function of the 128 MHz bandwidth lag spectrum is a triangular function (there are less long than short lags for a given time sequence), the Fourier transform of which is a SinC squared function. Thus, the lag spectrum is multipled by the weighting function, and by the convolution theorem, the frequency spectrum is the convolution of the Fourier transforms of the lag spectrum and the weighting function. This indicates that adjacent channels are not independent and that the sidelobes of the SinC squared function propagate into the spectrum.

   If the lag spectrum is corrupted by a DC offset, its Fourier transform is modifed by the addition of a zero frequency delta function. The convolution of this delta function (which might be very strong) and the SinC squared function can thus produce a strong ringing (i.e., propogation of the SinC squared sidelobes) throughout the spectrum. You can see any such DC offset in the spectrum that POSSM produced (see above) by examining channel 1 of the spectrum (see above). If the offset just occurs in one baseline you would need to plot individual baselines with POSSM to see this clearly.

   However, there is an easy cure. The SinC squared function has zeros every other *odd* channel. Therefore, you need only select the odd numbered channels in the good channel range that you determined above with POSSM. In addition, because adjacent channels are not independent, you sacrifice very little with regards to the signal-to-noise ratio in doing this. AVSPC has an easy option for doing just this. It may be worth doing regardless of whether you can see any DC term or not. However, note that this discussion is for the *128 MHz bandwidth only*. The situation is more complex at other bandwidths and this simple procedure is not a cure for them.

4. Now set up AVSPC by specifying which channels you wish to average. You can specify up to 10 groups of channels to average with the chansel adverb. In this way you can exclude channels with strong interference in them. Alternatively you could edit the data first and then use AVSPC specifying a channel range which included bad but flagged data - this is the better way to exclude corrupt data. Note that the FG table is not copied to the output. All flagged channels are excluded from the average so that by definition, all data in the output file are good. Editing of the data is described in the § 7.

AVSPC	
flagver=0	Apply highest FG table if any
chansel=7,27,2	Select odd channels in range 7 to 27
chansel=7,27,1	Select all channels in range 7 to 27

5. After you have run AVSPC, examine the header of the new file with the verb IMHEAD. The new bandwidth should be reflected in both the header and in the FQ table (use PRTAB to see this). In addition, the frequency reference pixel should have changed to reflect this new increment (note that an integer pixel value refers to the centre of a channel). However, the reference frequency will be unchanged. The reason for this is that the (u, v, w) coordinates assigned to each visibility are referenced to this frequency. If you change it, and then for some reason need to recompute (u, v, w) from the antenna (AN) file, they will come out all wrong. If you select a group of channels symmetrically placed about the reference frequency, then it will remain unchanged in the output data base header.

6.3 Spectral binning

AVSPC also has the capability to bin up channels in a visibility file. For example, you may wish to take a spectral line file with 1024 channels, and bin it up by a factor of 4 because you stupidly observed with way too high spectral resolution.

AVSPC	
flagver=0	Apply highest FG table if any
avoption='subs'	Spectral binning option
chansel=1,1024,1	Select channels
channel=4	Bin up by factor of 4

6.4 Averaging in time

The main advantage to averaging in time is to reduce the size of the data base. If you are working with large spectral-line data bases, this may be essential to reduce the data quantity to something manageable. Another reason you might like to average is so that any signal in very noisy visibilities can be seen more clearly when you are editing the data.

If you wish to average, use the task UVAVG, but make sure you do not incur significant loss of intensity owing to time smearing. This occurs when the source visibility structure changes significantly on the time scale over which you are averaging. It won't affect your calibrators much (as the visibility amplitudes should be roughly constant and the phases only slowly winding [assuming it is close to the phase centre]), if at all, but your program source might be badly affected. Do the sums correctly (see Appendix D for a more complete discussion) or don't average in time.

Another potential danger with time averaging is that you can't correct the antenna gains on a time scale shorter than the integration time with the self-calibration procedure. Atmospheric fluctuations that affect the antenna based gains may occur on time scales as short as a few seconds. However, you may not have sufficient signal-to-noise ratios in your data to benefit from this. Another way to put this is to say that if the atmospheric phase is winding systematically and significantly on a time scale of the order of 10 seconds, then averaging the data will degrade its quality irretrievably. You should inspect a piece of your observation first, and then average it if it is still desirable to do so. You could use UVPLT or IBLED for this inspection (see § 7).

As an example, consider a 6 cm observation with the 3 km array. If we demand no more than a 5% loss of intensity of a point source owing to time smearing at a distance of 6 arcminutes from the phase centre, then the integration time could be as long as almost 2 minutes (see Figure 2 in Appendix D). The acceptable averaging time would be longer for 20 cm, by 20/6.

UVAVG will not average across source or frequency changes. Note however, that it cannot read FG (flagging) tables (see § 7 below). Thus, if your data suffer from, say, intermittent but severe corruption (perhaps from interference), then averaging in time will corrupt what was previously uncorrupted data.

UVAVG	
xinc = 1	Write all records
yinc = 30	Averaging time in seconds
opcode = ' '	Average data

Last update : 27/11/93

7 EDITING THE DATA

It may be that you will want to average your visibilities in frequency and/or time. This is discussed in § 6. However, if you are going to do so, you may need to edit your data first to that you do not average in corrupt correlations. On the other hand, you may be confident in the quality of your data and wish to average first and edit later. I suggest you read this section and the section on averaging before embarking upon either.

A few words of caution about editing in general. First, there is generally no need to get too carried away with editing unless you are looking for extremely high dynamic range. The reason is that each pixel in an image of the sky's intensity distribution is formed from a linear combination of *all* the visibilities. Thus, when you have thousands of visibilities, one visibility has to be very bad to make an appreciable dent in the quality of the image.

Second, with an instrument such as the VLA, one can be very cavalier with editing; there are so many baselines that obliterating large swathes of data does not hurt much. However, the more modest number of baselines provided by the ATCA dictates that one edit with a little less abandon. Of course, this also means that in general, an equally bad visibility will cause more harm in an ATCA image than in a VLA image (all other things being equal) because of the much smaller number of baselines.

A third caution is to do with flagging (marking data as bad) sources with complicated structure. If a source (such as a calibrator) is a point source, then all the visibilities should have a roughly constant amplitude, even before calibration. Thus, poor data are very easily spotted. However, if a program source contains a lot of structure, then the visibility amplitudes will vary with time, baseline and frequency. In 128 MHz bandwidth mode in the 13 and 20 cm bands, you may also find some significant structure across the bandwidth (this is why we do multi-frequency synthesis). In these sorts of cases, true structure (quasi sinusoids with time) may appear to the inexperienced eye as bad data. Often, apart from really obvious garbage, it is best to leave the program sources largely alone until after the initial calibration and imaging. Bad data in a program source does not affect the calibration, as that is determined from the point calibrator sources. Therefore, one can flag the program source either before or after calibration, and often the best way to discover that there are bad data in the program source is to make an image after calibration, and then go hunting for the rubbish.

The quality of data now coming from the ATCA is sufficiently good that editing should be kept to a minimum except for data corrupted by externally generated interference (such as that produced by the Russion GLONASS satellites).

Finally, I recommend you *always* examine the spectral data at 13 and 20 cm, even if you are not interested in retaining the spectral information in the long run. This is because there is often channel-specific interference in these bands and you do not want to include it in your channel 0 data. A good tool for just inspecting your spectral data, even if you don't edit it, is SPFLG, as described below in § 7.2.

7.1 Averaged continuum (channel 0) data

Consider now the editing of a continuum data set in which you have averaged each band to one channel (channel 0) as discussed in the previous section. Inspection of the data is now fairly straightforward.

1. First, it is good to get an overall feel for the data; do they appear to be of high quality, or are there sections of bad data etc. The best way to do this is to make some plots. There are two main plotting tasks that you might use: UVPLT and VBPLT. Unlike all other *AIPS* tasks, UVPLT can plot all `freqids` together. VBPLT has the capability to put multiple plots on the one page, one for each baseline.

   With UVPLT, the sorts of plots that are initially useful are plots of all the visibility amplitudes and/or phases from all baselines as a function of both time and uv-distance ($\sqrt{u^2+v^2}$). Use the `bparm` adverbs to control these quantities. When you plot amplitude versus time, include all sources, as the time axis distinguishes sources. When you plot amplitude or phase versus uv-distance, do them one source at a time, or you will get all the sources jumbled up on top of each other. When plotting time with UVPLT, I suggest you use time in hours (`bparm(1)` = 11) rather than the aesthetically displeasing decimal days. The adverb `xinc` is also very useful; it lets you plot every `xinc`th point. This is very handy when you have large data bases. If the data are sorted in time order, make sure `xinc` does not divide exactly into the number of baselines. Otherwise, you will see the same baselines over and over. This is not a problem with data sorted in spacing order.

UVPLT	
sources=' '	Plot all sources
calcode=' '	Plot all sources
qual=−1	Plot all sources
stokes='I'	Plot Stokes I or plot
stokes='RR'	RR or LL if polarization
stokes='LL'	trickery (really XX,YY) invoked
freqid=2	Select frequency id
uvrange=0	Select full uv range
docalib=−1	No calibration
flagver=0	Read highest version
xinc=1	Plot every xinc'th visibility (if you have a lot you may want to make this bigger)
bparm=11,1,0	Plot time (x-axis) v amplitude (y-axis) and self scale plot
bparm=0	Plot *uv*-distance v amplitude and self scale remember to select one source only in this case

For calibrators, you should see a roughly constant amplitude with time or *uv*-distance and some scatter determined by the noise. You might see a gain jump, where suddenly a source gets brighter or fainter by tens of per cent. However, this is not necessarily a problem and should not be marked as ready for the grim reaper just yet, because you can probably calibrate it out. It is really no different from a slow gain change – except that the time scale is a bit shorter. You will have to be a little careful in the calibration procedure though, to deal with this correctly, and I will discuss what to do in the appropriate section. Such a gain jump is most likely to occur because you reran the on-line calibration program (`CACAL` or `DELCOR`) for some reason (not that this is generally reccommended). It is extremely unlikely that the receivers themselves underwent a gain jump.

2. You may notice from your plots that data are often corrupt at the beginnings of scans on all baselines (this problem is most common in ATCA data taken in the first couple of years of operation). These problems usually follow a change of source or frequency, or both, as sometimes the system takes some (short) time to settle down when reconfigured. There is a specialized task to deal with this. It has the unlikely name of `QUACK`; I presume you are 'doctoring' the data. It enables you to flag a specified amount of time at the beginning of all scans in a specified time range in a simple fashion. Look out for the confusing definition of `opcode='end'`.

QUACK	
sources='2512 − 92',' '	Select source to flag
timerang=1,23,10,0, 2,0,10,0	Select desired time range to `QUACK` over In this case, examine data from day 1 23 hours, 10 min and 0 sec to day 2, 0 hour 10 min and 0 sec
flagver=3	0 means 1 here. Specify if you want something other than one.
opcode='beg'	Flag first aparm(2) minutes of scan
opcode='end'	Flag up to aparm(1) minutes from end of scan
reason='bad AGC'	Write reason into flag table too. May be used by `UVFLG` to easily unflag data
aparm=0.5,0.5	Specify times to flag in minutes as specified by opcode

3. Now, having plotted and `QUACK`ed the data, it is time to clean it up in earnest, if required. The task best suited to this is `TVFLG`, which displays the data on a TV device as baseline in the x-direction and time in the y-direction. The intensity of the image displayed might be amplitude, or phase, or amplitude difference from the running mean and so on. Thus *all* your data at *one* frequency can be clearly displayed at one time. You can read the `EXPLAIN` file to `TVFLG` to find out exactly what it can do. However, `TVFLG` is a little daunting at first, so feel free to seek help straight away with it. It is easier to learn how to drive it by demonstration than by reading. But it is worth learning. Note also that `TVFLG` has good on-line help for every command. Usually, you can start it up in default mode, taking care to specify the averaging time equal to the integration time of your observation (15 seconds or the like). `TVFLG` will now display the visibility amplitudes, and it may do some averaging in time to fit it all on the TV.

Now, with the menu and the mouse select (two steps – first click on desired box with the mouse and then hit button A at the top of the workstation keyboard) to display amplitude differences from a running mean. Then select the number of integrations over which to form the running mean (say 20 or so) and select to reload the image. TVFLG will then display the new quantity. Now comes the real power of such a task. Sporadic bad data will stand out very obviously as a large amplitude difference. You can then flag these bad data out with the cursor, selecting by pixel, by area, by time or by baseline and so on. This is a very fast way to clean up obvious garbage.

Experiment with the various options of TVFLG to become comfortable with it. For example, the clip option is also very useful. This option can be used to clip say, on an amplitude or an amplitude difference image. You specify the range outside of which data are clipped and flagged as bad. In this way, TVFLG replaces the functionality of the CLIP task which is not useful for multi-source tasks. Make sure you do not iteratively clip your data, at each iteration apparently reducing the rms. Eventually, after you have done this enough times you will have clipped away *all* of your data. Just do it once to get rid of the really bad outliers.

When examining your program source, you might find the option to order the baselines by increasing length helpful, because for sources that are extended, you expect the visibility amplitude to increase with decreasing spacing (the unmeasured zero-spacing is the total flux of the source), and this helps distinguish genuine source structure from errors. However, again, I suggest that you concentrate largely on the calibrators at this time.

TVFLG doesn't actually write to the FG table until you exit, and then you have the option to write to the FG table. In addition, it writes (and optionally catalogues) a master image where the data have been gridded (TVFLG uses this image to keep track of source names, frequencies, times, baseline numbers, and if samples have been flagged already). You can restart TVFLG and have it read this file in, rather than have it recompute it, which may take a significant amount of time. This option is useful only if you are starting up TVFLG to do some more editing with exactly the same inputs. Otherwise, TVFLG will need to make a new grid.

Note that it is meaningless to display Stokes I from a data base that you have loaded as polarizations (i.e., no Stokes conversion), and which has not had an on-line (done with observing program `delcor`) or off-line calibration made or has not had the XY phase difference applied to the Y gains (see § 4.1). This is because XX and YY (called RR and LL with the polarization kludge) must be summed to make I, and they must be added up in phase.

If your source is polarized, it is important that when you flag one polarization as bad (e.g., $RR = XX$) you flag them all. This is because you will form Stokes $I = XX + YY$, and it is not true that $XX = I$ or that $YY = I$ when the source is polarized. You can interactively set the Stokes flagging mask with TVFLG by selecting the 'SET STOKES FLAG' option and typing in the mask in the *AIPS* window. Each bit in the mask corresponds to a polarization type. Thus, mask= 1011 means flag XX, XY and YX, but not YY. Mask= 1111 means flag all polarizations. This is what I recommend you use if the source is polarized. You might use this mask even if the source is unpolarized, working on the assumption that data which are bad in XX are likely to be bad in YY too. This is often a good assumption and may save you some time in the editing procedure.

TVFLG	
docat=1	Catalogue master work file
in2seq=0	Create new master work file or
in2seq=2	Select old master work file with sequence number 2
sources=' '	Select all sources
timerang=0	Select all times
stokes='I'	Display Stokes I or
stokes='RR'	RR or LL if polarization
stokes= 'LL'	trickery (really XX,YY) invoked
freqid=1	Select desired frequency id
antennas=0	Select all baselines
baseline=0	
docalib=−1	No calibration
flagver=0	Write to highest version of FG table already present or make a new one
doband=−1	No bandpass to apply
dparm=0	
dparm(6)=15	Set minimum integration time in seconds

4. Either solely, or in conjunction with TVFLG, you should consider the use of IBLED (Interactive BaseLine Editor). This is a task that allows much finer control over the editing procedure than TVFLG. It provides interactive editing of graphical displays (on a TV device) of either amplitude or phase as a function of time, one baseline and polarization at a time. Since the ATCA has, at most, 15 baselines, most people do have the patience to edit 12 hours of data with IBLED. It allows you to do an extremely careful job of editing. For strong program sources, you can allow your eye and brain to do mean and r.m.s. calculations, so that you can edit them as well as you might edit your calibrators. But remember the warnings about being over zealous with your editing. A point that is just a bit wrong may be more useful retained than discarded.

   IBLED breaks up each full observation into a number of shorter time frames, and plots one at a time, allowing you to step through them sequentially. It also plots all the data points from the full time range at the top of the screen so that you can select a frame at random. This is useful if you have already cleaned a lot of the data up with TVFLG, and just need to do some filagree work. You can step through the baselines sequentially or at random also.

   The actual editing is achieved by placing a vertical cursor on the offending point or by embracing a collection of points with a cursor designated box. Like TVFLG, the editing commands are written into the FG table on exiting from the program.

   The comments about Stokes flagging masks in the discussion on TVFLG above applies equally to IBLED and any other editing task.

IBLED	
docat=1	Catalogue master work file
in2seq=0	Create new master work file or
in2seq=2	Select old master work file with sequence number 2
sources=' '	Select all sources
timerang=0	Select all times
stokes='I'	Display Stokes I or
stokes='RR'	RR or LL if polarization
stokes='LL' p	trickery (really XX,YY) invoked
freqid=1	Select desired frequency id
antennas=0	Select all baselines
baseline=0	
docalib=−1	No calibration
flagver=0	Write to highest version of FG table already present or make a new one
doband=−1	No bandpass to apply
dparm=0	
dparm(5)=15	Set minimum integration time in seconds

5. There are other more basic tools available for data editing. If you suspect that a particular section of data has some miscreant points in it, you might like to just plot it quickly, rather than wait for IBLED to do so. You can use UVPLT again for this; fine control is given by the **timerang** (specify time range), **antennas** (specify antennas only) and **baseline** (specify actual baselines with the combination of **antennas** and **baseline**), and **uvrange** (specify a *uv*-range of data) adverbs.

   Another very handy task is UVFND. It allows you to set clip levels, above and below which data are deemed bad and reported to the user. The example shows how to look for bad data on one source for which you expect the flux density to be about 2 Jy on an uncalibrated channel 0 data base. It searches for data outside the range 2.5 and 1.5 Jy. UVFND has a full complement of calibration adverbs.

UVFND	
channel=0	One channel only
niter=500	Print 500 lines maximum of output
uvrange=0	No *uv* range restriction
stokes='I'	Examine Stokes I or
stokes='RR'	RR or LL if polarization
stokes='LL'	trickery (really XX,YY) invoked
sources='1223-23',' '	Select desired source
timer=0	No time restrictions
freqid=3	Select frequency id.
docalib=−1	Optional calibration
gainuse=2	CL table to apply if docal true
flagver=0	Read highest version of FG table
opcode='CLIP'	Search for points outside of range
aparm=2.5,100,1.5,0	100 and 0 ensures no cross hand point will be found erroneously

You can also print out some of your data, just to confirm your suspicions about its quality, with PRTUV. Although PRTUV is not enormously well endowed with adverbs (and especially it does not read FG tables) you can select by source and time range, so it's sufficient. Note that the adverb ncount in PRTUV gives the maximum number of visibilities to print. Its default value of 0, just gives one page of listing; set it to 100 or 1000 or so. You can always quit when you have seen enough, as all these tasks, when listing to the terminal, give you the option to stop every page of output.

PRTUV	
sources='1333-33', ' '	Select source
channel=0	For continuum data base
channel=10	Print channel 10 from spectral data base
ncount=1000	Print at most 1000 visibilities (0 means 1 page)
xinc=1	Print every visibility
cparm=0,10,0,0, 0,10,20,0	Select time range to print 10 hours 0 min to 10 hours 20 min on first day

You might also use LISTR to list your data in a variety of ways. In particular, LISTR has a matrix listing option well known to VLA users. LISTR, however, can also apply any calibrations you desire, so that you can print calibrated data later on if you wish.

6. If you have tracked down some more bad data with any of UVPLT, UVFND, PRTUV, or LISTR, now you need to actually flag it. Do this with the task UVFLG. Note that the **freqid** adverb is not included in UVFLG, so that you must specify completely the bad data with the other adverbs, notably, **sources**, **timerang**, **antennas**, and **baseline**. UVFLG will write commands into the FG table, and you specify which version with the **flagver** adverb. You may wish to keep multiple FG tables when you're, say, testing the effect of flagging a certain section of data. A very useful adverb in UVFLG (and QUACK) is **reason**. This string, specifying why you flagged the data, is written into the FG table. It can then be used simply to unflag that data if you make a mistake. You do this by setting **opcode = 'reas'** and specifying the desired **reason**. Note that all other selection criteria are ignored when unflagging by **reason**. TVFLG and IBLED fill in the reason location in the FG table with their names and date. If you wish to unflag with all selection criteria (including **reason**) then put **opcode = 'uflg'**. These must be identical to the way the FG table entry was written to unflag an FG table entry. I suggest you read the UVFLG **EXPLAIN** file fairly carefully before getting stuck into it.

UVFLG	
sources='0233-27',' '	Select source
timerang=1,12,0,0,1,12,30,0	Select time range carefully
bchan=0	Flag all channels
echan=0	
antennas=2,0	Flag antenna 2 with all others
baseline=0	
stokes ' '	Flag all polarizations
flagver=0	Write to FG table 1
opcode='flag'	Flag data
opcode='uflg'	Unflag data by all selection criterial
opcode='reas'	Unflag data by reason
reason='antenna 2 died'	Reason so that it's easy to unflag

You may come across a task called CLIP in which you set a clipping range, above and below which the data are bad and are flagged. However, this task too has not been adapted for multi-source files, so that it is short on adverbs. CLIP works well for single-source files, but is best ignored for multi-source files.

7. If you need to look at the contents of the FG table, use the task PRTAB with inext='fg'. Note however, that the order of the table may not reflect the order in which you entered flagging commands. This is because a number of the calibration tasks will sort some of the tables into time order for speed of access, and rewrite them. You can copy the FG table (to a new file or the same one) with TACOP. If you get desperate, you can edit your FG table with TABED. You can also flag rows of it (flagging the FG table !!) with TAFLG and plot it with TAPLT.

8. Occasionally you may find you want to concatenate FG tables. This can also be done with TABED using the 'COPY' option. If the output table already exists, the input table is just joined to it. The most instance of needing to do this is as follows. Imagine that you do your initial editing on a spectral-line data base (see § 7.2 below). You then form a channel 0 data base with AVSPC and decide to edit further with it. You then wish to combine the FG tables from the channel 0 and spectral-line data bases. You can do it as in the example below. Make sure that when you edited the channel 0 data base that you set bchan=1 and echan=0 so that all the spectral-line data base will be flagged whenever a channel 0 FG table entry is found.

TABED	
inname='1934'	Input file
inclass='ch0'	
inseq=1	
indisk=4	
outname='1934'	Output file
outclass='uvtb'	
outseq=1	
outdisk=4	
inext='fg'	Append FG table
invers=1	Input FG version
outvers=1	Output FG version
optype='copy'	Copy mode

7.2 Spectral (line or continuum) data

If you decide to retain the spectral information, whether you are interested in continuum or spectral line data, then the editing procedure is more time consuming. This is obviously because of the additional dimension and no-one wants to fossick through every channel, one by one. There are two approaches to editing the spectral data base.

1. The first method is to form the channel averaged data base (channel 0, see § 6) anyway, and flag it with QUACK, IBLED, TVFLG, and UVFLG just as described in the previous section. You can copy the FG table back to the spectral data base, and apply the flags to all channels for the flagged time ranges and baselines. The rationale is that many problems occur at a certain time in all channels rather than in individual channels. This is probably a good way to proceed at 3 and 6 cm where channel-specific interference is rare. At 13 and 20 cm I would advise against this approach.

2. Similar to TVFLG is the task SPFLG. This task does the same job as TVFLG except that the spectral dimension is displayed in the x-direction instead of the baseline. In this way, you can edit fairly fast, but you must display each baseline separately. It also removes the useful feature of flagging all baselines simultaneously at a common time stamp. IBLED suffers from the same defects. SPFLG is *absolutely vital* in most spectral line experiments as, up until the time of writing, interference has been a severe problem, especially in the 13 and 20 cm band. It is the *only* way to edit such severely corrupted data properly. If you just worked on a channel 0 data base, you would probably find that you flagged everything.

   See the discussion on TVFLG in § 7.1 for some help on how to use SPFLG.

SPFLG	
docat=1	Catalogue master work file
in2seq=0	Create new master work file or
in2seq=2	Select old master work file with sequence number 2
sources=' '	Select all sources
timerang=0	Select all times
stokes='I'	Display Stokes I or Stokes
stokes='RR'	RR or LL if polarization
stokes='LL'	trickery (really XX,YY) invoked
freqid=1	Select desired frequency
bchan=1	Select all channels
echan=0	
antennas=0	Select all baselines for
baseline=0	sequential display
docalib=−1	No calibration
flagver=0	Write to highest version of FG table already present or make a new one
doband=−1	No bandpass to apply
dparm=0	
dparm(6)=10	Set minimum integration time in seconds

Last update : 27/11/93

8 DETERMINING THE ANTENNA GAINS

In this section I will discuss the task CALIB, which is the work-horse of the calibration package (apart from yourself). CALIB determines the antenna gains as a function of time from the calibrator source(s). It determines the gain solutions only at the times of the calibrator(s). These gains are then interpolated to the times of your program source(s).

I remind you that a full gain and polarimetric calibration can only be made with *MIRIAD*. Please refer to § 4 for advice on the compromise you are making by calibrating your ATCA data in *AIPS*.

8.1 What to do with spectral data

If you averaged all your data into one channel per IF frequency, skip to § 8.2.

You need to determine the gains from the calibrators as a function of frequency as well as time. We would hope that the gains are stable in time and frequency, but generally the variation from channel to channel is more stable in time than the gain variation of a single channel with time. This is because the latter depends to a large extent on the atmosphere, and the former doesn't.

CALIB can only work on one channel at a time (or channel 0), so that you can't determine the calibration as a function of frequency with CALIB alone. Thus, CALIB is used on one channel, and the task BPASS is used to work out the variation across the band (see § 11). Hence, you need to produce a channel 0 data set for CALIB to work on, regardless of whether you want to do spectral-line or multi-frequency continuum synthesis work. If you haven't already produced the channel 0 data base, do this with AVSPC as detailed in § 6.2. Include as many (good) channels as possible in this average to allow the highest signal-to-noise ratio solutions (see discussion on how to determine this in § 6.2).

8.2 Computation

1. The procedure is the same whether you have converted the correlations to Stokes parameters ($IQUV$; not recommended) or left them as linear polarizations (but called them circulars, RR, LL, RL, LR), as the relevant tasks account for this. In the Stokes case, solutions are only found for I (as we don't have polarization models). In the latter case, two sets of solutions are found, one for XX (called RR) and one for YY (called LL). In addition, CALIB will determine the gain solutions for all the IFs contained in your data. There is no IF selection.

2. If you look carefully at the CALIB inputs, you will see that averaging over channels can be done within CALIB (with the bchan and echan) adverbs. However, I encourage you to use a separate channel 0 data base, rather than use this option. The reason is that the averaging switches are fairly well buried in CALIB's inputs, and they are poorly documented and easy to get confused over. The safe route is via AVSPC.

3. In this sub-section you should be working with the channel 0 data, regardless of your final spectral goal. The task CALIB determines the antenna gains as a function of time from the calibrator sources, and writes the solutions into the solution (SN) table ready for interpolation to the program source(s). If you examine the inputs to CALIB, you will find that its adverbs are as generously allocated as in ATLOD. CALIB is a very general task, and many of the inputs are irrelevant to our purpose. However, rather than continually scroll through all these unwanted adverbs, I have provided some *AIPS* procedures which hide many of these adverbs.

   The CALIB procedure is called ATCALIB. See § 2.2 for some useful information on the procedures. Most of the inputs have the same name as in CALIB; a couple are changed because they are normally poked into those aparm, bparm, cparm or dparm arrays. Obviously, ATCALIB has lost some flexibility compared with CALIB, but it should do what you want for this first calibration (if not, use CALIB). For example, it assumes that the model for each calibrator source is a point source at the phase centre; this is the usual premise for initial calibration (see § 3.1). If there is no flux density in the SU table for a particular calibrator, ATCALIB will assume that 1 Jy is the correct flux density; that is, application of the gains to that source would yield a mean visibility amplitude of 1 Jy. The only source for which you should have a correct flux density in the SU table is the primary calibrator, 1934 – 638 (see § 5). Later on, you will account for the 1 Jy assumption by comparison with the primary calibrator. This is called boot strapping the flux density scale (see § 9).

4. Because of ATCALIB's importance, I will discuss all of its adverbs.

- Fill in **inname, inclass, inseq** and **indisk** with the **GETNAME** verb. Make sure you get the channel 0 data base.
- Select the desired calibrator sources with **calsour** and **calcode**. If you have set **calcode** values as discussed in § 5, then it is often most convenient to leave **calsour** blank, and put **calcode='*'** to select all the primary and secondary calibrators. Otherwise, fill in the *calibrator* source names in **calsour**.
- Set the **freqid** (see **LISTR** summary sheet for these). Remember each **freqid** must be processed separately so that one run of **ATCALIB** for each frequency is required.
- Initially, you will probably want to solve for all the data so leave **timerang** as all zeros. Otherwise, **timerang** has space for a start time (day, hour, minute and second) and, similarly, an end time.
- Select the antennas to solve for with **antennas**. Generally, you will want them all, so leave this at zero too.
- Calibrators are not always perfect point sources, depending exactly on the size of the baselines that you observed them with. As you go to longer baselines, fine structure may become apparent. As you go to shorter baselines, extended structure may become apparent. **uvrange** allows you to select those baselines for which the source is strictly a point source. To assess this, use some **UVPLT** amplitude versus *uv*-distance plots. Be careful in setting the **uvrange** though, because you must have enough baselines to get good solutions (the more baselines the more the problem is overdetermined). However, because the compact array is just that, compact, the point-source model should be fairly good on its longest baselines and you may need to worry about excluding only the shortest baselines if the visibility amplitudes show signs of short baseline structure.
- Set **docalib=-1** as you don't have any calibration to apply yet and ignore **gainuse** which specifies which calibration table to apply if **docalib** is true. Generally, you would only set **docalib=1** while doing self-calibration iterations; it is used to apply a previously determined calibration. For basic calibration as you are doing here, leave it negative.
- Set **flagver** to zero and the highest numbered FG table will be applied.
- **refant** should be set to the number of the reference antenna, for which the phase will be set to zero (see § 3.1). It should be present throughout the observation, have a good signal-to-noise ratio, and be free of gain jumps. You might as well use the antenna chosen as the reference during the observations, but it is not vital. If you leave **refant** at zero, the antenna chosen will be arbitrary.
- Usually, we solve the calibration equations with a least-squares algorithm; this is obtained with **soltype=' '**. However, sometimes AT data processed like this has been prone to giving large numbers of failed solutions. The L1 solutions, invoked with **soltype='L1'**, in which just the modulus of the difference of the model and data is minimized instead of the square, appears more robust. This is probably because sometimes the real and imaginary parts of the visibilities are not drawn from good Gaussian distributions. The L1 solution is less affected by outliers and fails less often. You should try the least squares solution first, and if that does not prove efficacious, try the L1 solutions.
- **solint** controls the solution integration time (in minutes) over which the data are averaged. Leave this at zero, and all the data in a scan will be averaged. This is the usual approach, as calibrator sca. are generally just a few minutes long (howver, see discussion in § 8.3).
- **doscalar** controls whether the data are scalar or vector averaged over the solution interval. In a vector average, the real and imaginary parts of the visibilities for each baseline are averaged, and then the amplitude and phase can be formed from the averaged visibilities. In a scalar average, the amplitudes for each visibility are formed first and then averaged. However, the averaged phase is still made with a vector average, as a scalar average for a circular quantity like phase is not very meaningful. If **doscalar** $>$ 0 then a scalar average will be done, otherwise a vector average of the data is taken. Generally, you should set doscalar=-1, but see § 8.3 for a discussion of when this may not be true (bad phase stability).
- **minant** controls the minimum number of antennas that must be present in order to get a solution. Since the ATCA has only 6 antennas in the compact array (5 in the 3 km array and one 3 km further away), the loss of one or two antennas because of failure becomes important in determining the solutions. Should this occur for some period of time, you can exclude these scans by demanding that **minant** be, say, at least 5. This means you would then have to interpolate the solutions over a longer time baseline. It's a trade off. Experience suggests **minant=3** is a good choice. But really, the bigger the better (but not more than 6 !). The default of zero means three antennas.
- If **minamper** and **minpherr** are non zero, then residual errors are listed to the screen. For individual baselines, these are amplitude percentages from the model amplitude (the flux density for a point

source) and degrees from the model phase (zero for a point source at the phase centre), respectively. These are about the only diagnostics there are from CALIB to let you know when bad solutions are occurring (except outright failure). Set these to something like 10% and 10 degrees to start with. Any baseline that returns larger residuals than these will be reported to you. The actual errors you should expect because of noise only, clearly depend on the scan time and calibrator strength (see Cornwell 1981), but the occurrence of no residual errors above a few degrees and a few per cent is typically what we aim for. These residual errors are often called closure errors.

- docrt, if negative, allows you to list the messages from ATCALIB on the line printer automatically . These messages (e.g., closure errors) are normally written to the terminal as well as to the message file, and turning this switch on just invokes the *AIPS* verb PRTMSG to print the message file. Note, however, that it clears the message file of all old ATCALIB messages before printing the new ones. If docrt is non-negative, the messages just go to the terminal and the message file as usual.

5. Now it is time to set ATCALIB in motion with the GO ATCALIB command. Remember that you must run ATCALIB once for each source requiring a different combination of freqid, uvrange, and antennas and write a new SN table for each run. As it computes, ATCALIB reports the closure errors to you, and at the end, it will tell you how many good and how many failed solutions it found. A failed solution is when the algorithm is unable to find a global minimum for the function that it is minimizing within its internally allocated number of iterations.

ATCALIB	
inname,inclass,inseq,indisk	Fill in for averaged data base
calsour=' '	Select sources explicitly or leave blank
calcode='*'	and use calcodes ('*', 'p' or 's' say)
freqid=1	One run of CALIB per freqid is necessary
timerang=0	Solve for all time ranges
antennas=0	Solve for all antennas
uvrange=0	Solve with no uv restrictions or
	specify range in $k\lambda$
docalib=−1	No calibration to apply
flagver=0	Apply highest version FG table
refant=3	Specify the reference antenna
soltyp=' '	Least squares algorithm. If too many
	failed solutions, try 'L1'
solint=0	Averaging time in minutes. 0 means scan aver
doscalar=−1	Vector average
minant=3	Min. no. of antennas to work out a solution
minamper=5	Report amplitude errors bigger than 5%
minpherr=5	Report phase errors bigger than 5 degrees
docrt=1	List results on terminal
docrt=−1	List results on line printer

8.3 Should I scalar or vector average in the solution interval ?

If the phase stability is bad during the observation (the atmospheric conditions are poor), the phase of the antenna gains can vary rapidly. A change of tens of degrees or more during the calibrator scan is not uncommon. However the software that determines the antenna gains assumes that the gains are constant during a solution interval. So during periods of poor phase stability, it is often desirable to make the solution interval of the calibration software quite short. While the resultant gains probably track the phase during the calibrator scan, we are more interested in antenna gains for the program source. The best guess at the antenna gains for the program source is an interpolation between the average gain during the calibrator scan. Thus, after determining the gains at a fine time step in the calibrator scan, you would ideally average these gains together to get some representative gain for the whole calibrator scan. This would probably be the best guess you can make (at least as far as correcting the program source is concerned), although in times of truly awful phase stability, its a pretty poor guess (self-calibration will be needed in this case).

There are two ways in which you would want to average your gains, either the vector or scalar average. The general rule is that if you are going to self-calibrate later, then scalar averages are probably the most appropriate. Assuming the variation in gain is purely due to poor phase stability, a scalar average will give you a good estimate of the amplitude of the gain – which is advantageous for self-calibration.

On the other hand, if you are not going to self-calibrate, you are more interested in getting a good estimate of your image at this stage, and less concerned about partially correct gains (the two do not necessarily go hand in hand). If we were to use some average antenna gain, the poor phase stability will cause partial decorrelation in both the program source and calibrator. This will result in apparent reduced flux densities of both of them. Assuming that the decorrelation is approximately the same for both, then we could scale up the program source by the decorrelation that we note in the calibrator. This could be achieved by vector averaging the antenna gains.

You may have guessed from the frequent use of the word "would" in the discussion above that *AIPS* cannot do all that we would want here. It does not offer a useful gain averaging task. The task **SNSMO** can smooth gains, but it can only do a scalar average. The alternative in *AIPS* is to average the data rather than the gains. This is not ideal because the averaged visibilities may have non-closing errors in them (i.e., the mathematical model upon which the calibration procedure is based may be invalidated). However, in practice, it seems to work reasonably well. Thus, **ATCALIB** offers you vector or scalar averaging of data rather than vector or scalar averaging of the gains.

In summary, you should keep the solution interval at a scan average **solint=0**, and use vector averaging for data with good phase stability or if you not going to be able to self-calibrate your data (see § 17). Use scalar averaging if the phase stability fis poor and you are going to be able to self-calibrate.

8.4 Assessment and inspection of the gain solutions

1. If you incur substantial closure errors, then you should look carefully at the data for those times and baselines reported to try and find out what is wrong with it and flag it out; or just flag it out regardless. Note that the times reported by **ATCALIB** are the average times from the solution interval. **ATCALIB** has just written its solutions into the highest numbered solution (SN) table attached to the input file. The easiest way to examine these solutions is to plot the gains as a function of time with the task **SNPLT**. You should plot both amplitude and phase. You can plot the solutions directly on the TV, or plot the (PL) extension file created by **SNPLT** with **MIXPL**, **TKPL**, or **TVPL**.

   In amplitude, you should expect to see slow drifts in time of the order of up to $\sim$ 10%. In addition, at high frequencies and low elevations, there may be some atmospheric attenuation of the amplitudes, so that the inferred amplitude gain rises as the elevation decreases. The phase may wind more rapidly, being more subject to the effects of atmosphere and ionosphere. However, as a fiducial result, perhaps phase winds of tens of degrees over hours at 6 cm for the same source is not uncommon (you may do much better or worse).

   These plots should be examined for solutions that are obviously discrepant from the general trend. They may or may not have already made themselves known from the closure listings (but remember the times reported by **ATCALIB** or **CALIB** are the average times of the scan). Use **UVPLT** or **VBPLT** to plot all the data from the bad scan and hopefully the trouble maker(s) will be self-evident.

   It may happen that you find gain jumps. That is, an antenna suddenly jumps in gain (amplitude or phase) and then remains at the new level. Although these are not a good sign, they can be dealt with by careful calibration. Flagging them out is no good because the program source will have the same jumps, and you don't want to throw away too much program data. Note also that if a gain jump occurs in your reference antenna (all the antennas will then jump at one time instead of just one), then you should redo **ATCALIB** with a different reference antenna. Calibration with gain jumps will be dealt with in § 10.

SNPLT	
inext='sn'	Plot gains in solution table
invers=0	Plot highest table version
sources=' '	Plot all sources
stokes='I'	Plot Stokes I gains or
stokes='RR'	RR or LL if polarization
stokes='LL'	trickery invoked
freqid=2	Specify one freqid per run of SNPLT
pixrange=0	Self scale plot
ncount=6	6 plots per page (if 6 antennas say)
xinc=1	Plot all solutions
optype='amp' or 'phas'	Plot gain amplitude or phase
dotv=1	Plot directly on TV or
dotv=−1	make plot extension file

2. **ATCALIB** will tell you if it was unable to find a solution for some time ranges. It will list the number of failed solutions. The ones that failed will not appear on your **SNPLT** plots. Rather than just writing them off

as useless, it may be advantageous to find out why they failed. To do this, you to poke a bit harder into the system.

You can print the SN table on the line printer with PRTAB (put `inext='sn'` and see Appendix C. Under the columns listing the real and imaginary parts of the gain solutions, look for solutions called 'INDE'. This means indefinite and is the code for failed solutions. You can then inspect the data at the relevant times (also listed in the SN table) to see if you can work out what failed. Possibilities are too much flagging and insufficient signal-to-noise ratios for CALIB to get its teeth into; if the S/N is too poor, CALIB will set the results to 'INDE' (this cutoff level is set at 5 by ATCALIB but is an adverb in CALIB). Also, if `minant` is set to the number of antennas in the array, and one has dropped out for some period of time, failed solutions will arise.

3. Some people prefer to list their gains as numbers and look at those rather than plots. If you prefer this, use LISTR with `optype='gain'`, `dparm(1)=3` (for amplitude and rms) or `dparm(1)=4` for phase and rms. Don't get confused by the scale factors that LISTR prints its results with (they are printed at the top of each page). I think that the plots provide a better qualitative feel as well as providing adequate quantitative information.

4. If after all this inspection you decide that some more flagging is necessary, go and do it; UVFLG is best for this fine detail work. Then redo the gain determination. Keep on iterating until you are happy. Usually, one or two passes are sufficient.

5. If you find some bad solutions, but cannot find any reason for them being wrong (i.e., the visibilities look fine) then you have two options for those miscreant scans. You could flag the calibrator scans, or you could flag the gain solutions in the SN table. The former is the safe way to do it, but then if you want to have another go at them later, you would need to unflag them. The task SNCOR can be used to flag actual SN table entries by setting clip levels. This may prove the quickest way to ensure bad solutions do not get applied. However, it is a less robust approach, since should you want to redo the calibration, you have to keep remembering to flag these bad solutions before applying interpolating them to the CL table. SNCOR can also be used to zero the phase part of selected solutions.

   After finishing with ATCALIB (or CALIB if you prefer), you should have produced one or more SN tables.

8.5 Deleting SN tables

Each application of CALIB or ATCALIB generates a new SN table. You can delete SN tables in a couple of ways. First, the procedure SNZAP deletes deletes all SN tables. Alternatively, you can use the verb EXTDEST to delete a specified SN table version.

Last update : 27/11/93

9 CORRECTING THE FLUX DENSITY SCALE OF SECONDARY CALIBRATORS

Before applying the antenna solutions to the program sources, you must set the flux density scale correctly. So far, via SETJY, you have set the scale for the primary calibrator only. When ATCALIB worked out solutions for the secondaries, it assumed a flux density of 1 Jy for each of them (unless you had entered a value into the SU table or already done a round of ATCALIB and GETJY). Now the solution amplitudes for the primary and secondary calibrators stored in the SN table can be ratioed to work out the secondary calibrator flux densities. The task GETJY is used for this purpose.

- Select the list of secondary calibrators with the **sources** and **soucode** adverbs. If you have a lot of secondaries, it is quicker to leave **soucode** blank and select them only with **soucode** (provided you used SETJY to give the secondaries a **calcode**).
- Select the primary calibrator with **calsour** and **calcode**.
- You may need to set the **uvrange**, **antennas**, **freqid**, **bif**, and **eif** combinations in the same way that you did when running ATCALIB and run GETJY several times.
- You may also find it necessary to set a time range with **timerang**. For example, if you have observed at low elevations and at a high frequency, the calibrator will probably have been attenuated by the atmosphere and the gains will come out appropriately bigger. Ideally, you should use secondary calibrator scans at the same elevation and close in time to the observation of the primary calibrator. In practice, you can only approximate this requirement.
- You can select a specific SN table with **snver**, or select *all* SN tables by leaving it at zero. If you select a specific SN table, then gain solutions for the primary and secondary calibrators must be contained in it. It may be necessary for you to delete specific SN tables before running GETJY if you want most of the available tables.

GETJY	
sources='0123 – 23','1322 – 45',' '	Select secondary calibrators by name
soucode='s'	and/or with source code
calsour='1934 – 638',' '	Select primary calibrator by name
calcode='p'	and/or select with calibrator code
qual=−1	Select all qualifiers
bif=1	Select IF range
eif=2	of interest
timerang=0	Select all times
antennas=0	Select all antennas
freqid=1	One run per freqid
snver=0	Select all SN tables or specify

- Following computation, GETJY enters the new secondary calibrator flux densities into the source (SU) table and reports them to the user as well. In addition, GETJY corrects the solution (SN) table(s) to account for the newly determined flux densities of the secondaries.

Last update : 27/11/93

10 INTERPOLATION OF THE GAIN SOLUTIONS

Now it is time to take the antenna gain solutions and interpolate them to the times of the program sources. In addition, it is a *very* good idea to apply the solutions to the calibrators themselves (self-calibration), because this provides an excellent check on the success of the procedure – the phases for the calibrators should all come out to zero (plus noise) and the amplitudes should come out to the calibrator flux density (plus noise).

10.1 General information

The task CLCAL is used for the interpolation procedure. This task has rather a lot of inputs too, and I have provided a reduced set in the procedure ATCLCAL. Again, if you find insufficient flexibility for your purpose in ATCLCAL, go back to CLCAL. It is probably a good idea to look at both sets of inputs to start, so that you can see what ATCLCAL doesn't let you do. ATCLCAL writes the interpolated gain solutions into the calibration (CL) table(s) which usually have entries every few minutes or so (you specified this with either ATLOD or INDXR). The CL table contains one (interpolated) solution per time entry, and there can only be one source/frequency combination per entry. CLCAL also 'manages' the SN tables. This means that it merges *all* the SN tables (so make sure you have deleted any you don't want with EXTDEST) into one and deletes the old ones before beginning the interpolation.

Note that a different run of ATCLCAL is required for each source and its calibrator (unless you have one calibrator for more than one source). I will describe here a fairly simple case of one primary calibrator, one secondary calibrator and one program source. Extension to additional complexity is fairly obvious. Although you have the option to write as many CL tables as you like, it is generally best at this stage to write all the interpolated solutions to table 2. This makes the book-keeping as simple as possible. However, if you are prone to making mistakes, then writing a new CL table for each run of ATCLCAL may prove useful, since you need only delete the CL table that you messed up, rather than deleting table 2 which might contain the results of many runs from ATCLCAL.

Beware of trying to ask ATCLCAL to do too much. Don't try to calibrate all sources with the relevant calibrators in one go and expect ATCLCAL to do the right thing. It does its interpolation in time without regard to *where* sources are. Thus you would be very likely to get solutions interpolated through sources which may be in very different parts of the sky (depending on exactly how your program was set up). Do it in the most fool-proof way, a bit at a time, and, in the long run, you will save yourself a lot of trouble.

It is important to realise that CL tables are designed to be used in a 'cumulative' fashion. That is, one always generates a new CL table by applying the current gain solutions to a previous CL table. Thus, gain solutions can be incrementally improved. This is the procedure used in the iterative self-calibration technique (see § 17). In the current case, you are performing the first calibration, so you apply the gain solutions to CL table 1, which is all ones and zeros for amplitude and phase, respectively, and write CL table 2. Never delete the first CL table, as it is used to go back to a pristine calibration (if you do, then you must rebuild it [and the NX table] with INDXR).

You should still be working with the frequency averaged data base (channel 0) at this point, again regardless of what the final spectral goal is.

10.2 Computation

1. After the first run of CLCAL on a group of SN tables, it will issue a message reminding you that you have already applied the SN table(s) once. This is normal and nothing to worry about.
2. You should now self-calibrate the calibrators as a check. Recall that you have generated one solution per antenna averaged over some time interval (probably a scan average). You are now going to apply that solution to the individual visibilities that made up that scan. First self-calibrate the primary calibrator, and then the secondaries.
   - Select the calibrator by selecting it as *both* the source and the calibrator to be calibrated with the adverbs `source`, `soucode`, `calsour`, and `calcode`.
   - Select the appropriate `freqid` to be calibrated (always one at a time).
   - Leave `timerang` to calibrate all scans.
   - Select `interpol='self'`. This causes the nearest solution in time (for the specified source) to be selected (this nearest solution is for the calibrator(s) you are trying to self-calibate as well, of course). Thus, each calibrator scan is calibated with the solution from that scan, rather than with an interpolated solution.

- `intparm` is not relevant at this stage.
- Select the desired SN table(s) with `snver` or leave it at zero, to select *all* SN tables. SN tables which contain no solutions for the selected sources are safely ignored, so generally leaving this at zero is safe. If you are testing different calibrations, which were written into different SN tables, then this is how you select them.
- Put `gainver=0` (apply SN table solutions to CL table 1) and set `gainuse=0` (write solutions into CL table 2).
- Leave `refant` at whatever you used in `ATCALIB` or zero which induces `ATCLCAL` to work one out for itself.

Type `GO ATCLCAL` and the calibrator (do both primary and secondaries) has been self-calibrated. You can write the solutions of as many sources as you like to the same CL table (usually tables 2).

ATCLCAL	
inname,inclass,inseq,indisk	Fill in for averaged data base
sources='0121 – 43',' '	Select sources to calibrate by name
soucode='S'	and/or select by source code
calsour='0121 – 43'	Select desired calibrators by name
calcode='S'	and/or calibrator code
freqid=3	One freqid per run
timerang=0	Select time range
interpol='self'	Select self-calibration
snver=0	All SN tables or select
gainver=0	Apply solutions to CL table 1
gainuse=0	Write to CL table 2
refant=3	Select ref. antenna same as in CALIB

3. Now you must calibrate the program sources.
   - Select the source(s) to be calibrated with the `sources` and `soucode` adverbs, and set `calsour` and `calcode` to select the desired calibrator. You may want to calibrate more than one source with each secondary in this step.
   - Put `interpol='2pt'` to select two-point interpolation of the calibrator solutions.

   Leave all the other adverbs as they were for the self-calibration of the calibrators, and run `ATCLCAL` again. This writes the interpolated solutions into the selected CL table (usually version 2).

ATCLCAL	
inname,inclass,inseq,indisk	Fill in for averaged data base
sources='0155 – 48','0101-25',' '	Select sources to calibrate by name
soucode=' '	and/or select by source code
calsour='0121 – 43'	Select desired calibrators by name
calcode='S'	and/or calibrator code
freqid=3	One freqid per run
timerang=0	Select time range
interpol='2pt'	Select interpolated calibration
snver=0	All SN tables or select
gainver=0	Apply solutions to CL table 1
gainuse=0	Write to CL table 2
refant=3	Select ref. antenna same as in CALIB

4. Run `ATCLCAL` as many times are as needed to get through all the different selection criteria you have. For example, with ATCA data you will need to run it for each of the `freqids` separately (but always writing CL table 2). If you have more secondaries and program sources, run it for each combination of these.

10.3 What to do with gain jumps

Gain-jump calibration follows the same procedure as above, with one modification. You must set the `timerang` and calibrate the pre- and post-jump sections of data separately. It is possible that the jump occurred in a

program source between calibrators. If you can determine this from inspection of the data, then fine, set the time range boundaries here. However, often the program source is too complicated or weak to show this clearly and the best solution is to flag out all the program source between the two calibrators which show the jump. You then calibrate before and after these scans separately.

The 'slap-dash' approach in which you just ignore the gain jump and interpolate through it, may, in fact, not be too bad. As the two-point interpolation approaches the time of the gain jump, the calibration errors will increase, and then decrease as they pass through the jump towards the next calibrator. Therefore, quite a lot of the data will be reasonably well calibrated. You can then take the approach that, if your source is sufficiently strong, you will correct these errors later with self-calibration. Since the best solution is really not very difficult to do, I don't recommend this second approach unless 'your plane back to Iceland leaves in 1 hour and you still haven't written your backup tape. Do it correctly if possible.

10.4 Assessment of calibration

This is a very important step which you should always undertake with the calibrated calibrators. It simply involves amplitude and phase plots versus time which you make with UVPLT in the usual way (see § 7) remembering this time to set docalib=1 and gainuse=2 (say) to turn on the calibration and select a CL table. You should examine the plots to ensure that the phases are distributed about zero and that the amplitudes are distributed about the flux density that is now in the SU table (i.e., that which you set with SETJY or GETJY). If you had gain jumps, you can now see whether you did a good job of calibrating them out.

An alternative is to print out lots of numbers with LISTR in a matrix format. Set optyp='matx', select the desired sources with sources and freqid, turn on the calibration with docalib=1 and gainuse =2 to select CL table 2, select amplitude and phase for display with dparm(1)=6, and send the results to the line printer with docrt=-1 and GO LISTR. I think this is an awful way to look at your data, but those entrenched in old VLA DEC-10 habits will feel a warm glow.

10.5 Resetting the calibration

It may occur that you get in a terrible mess; the calibration is in chaos and you have no idea what table has what in it. It is time to start over again at this point. This requires that you simply delete the appropriate tables. The *AIPS* procedure CLSNZAP deletes all SN tables and all CL tables except the first. Similarly, SNZAP deletes all SN tables, and CLZAP deletes all but the first CL tables. If you want to remove the flagging tables too, use EXTZAP (delete all tables of specified type) or EXTDEST (delete the specified table). To avoid confusion, it's probably a good idea to reset the SU table too. Use SETJY to do this with sources set to your secondaries, and optype='rese'. You don't want to reset the primary flux density to zero as well, just the boot-strapped secondaries. Don't delete the SU table, as there is no way to make a new one without reloading the data. Just do this resetting on the channel 0 data base.

Last update : 27/11/93

11 BANDPASS CALIBRATION FOR SPECTRAL DATA

This section explains how to derive the variation of the complex antenna gains as a function of frequency for spectral data. These responses are stored in the bandpass (BP) table. You must undertake this step if you have spectral-line data, or if you want to do multi-frequency synthesis.

11.1 General

It is necessary to calibrate the frequency response of each telescope, in both amplitude and phase. In principle, one could apply bandpass corrections in a baseline- or an antenna-based scheme. The former is the more conceptually accessible since the visibilities are baseline dependent. However, for large arrays such as the VLA, this would mean the bandpass table could become very large as there are $N(N-1)/2$ interferometers for N antennas. This is the main reason why the *AIPS* bandpass generating task, BPASS, decomposes the baseline-based responses into antenna-based responses and writes these to the BP table. This is done in a manner very similar to the general calibration discussed previously.

To make a good bandpass calibration, it is common to observe a strong source for some period of time. This is especially important for high dynamic range spectral-line applications. Usually, the bandpass calibrator is observed before and after the rest of the program, so that a check on the temporal stability of the bandpass can also be made. If you don't have a special bandpass calibrator, then you will probably need to use your secondary calibrator(s), or perhaps the primary, if you observed it long enough. So, you must decide whether to try and work out the bandpass response as a function of time, or whether you really only have sufficient signal-to-noise ratio if you average all your calibrator data together (usually it is the latter).

A useful technique in determining bandpasses is that which divides the spectral data by the channel 0 data first. This essentially removes any source structure and atmospheric gain (phase and amplitude) errors and reduces the source to a pseudo point source. Therefore the standard calibration equation (see § 3) can be used to work out the gains for each channel. This means that the bandpass calibrator does not necessarily have to be a point source, just that it is bright. However, this technique relies on their being minimal source structure across the bandpass. If you are doing multi-frequency synthesis work, then it is quite possible that there will be source structure across the 128 MHz bandwidth. In these cases, you should make sure that your bandpass calibrator is indeed a point source (such as 1934 – 638).

If this techniques is applicable, there is no need to calibrate the data, because the division removes the gain errors. In any case, even if you could calibrate with this option, it would also just divide out.

It is also important to realize that this type of bandpass calibration implicitly assumes that the bandpass calibrator's flux density does not vary significantly over the band. Remember this when and if you attempt multi-frequency synthesis, as it affects the validity of the whole procedure. Note that *MIRIAD* can do a better job of this because you can input the spectrum of the source if you know it. *MIRIAD* has built in knowledge about some calibrators, in particular, 1934 – 638.

11.2 Transfer of the calibration to the spectral data

You might have edited your data before you created the channel 0 data base with AVSPC, or you might have edited after. If you edited first, then the FG table is in place attached to the spectral data base. If you edited after averaging in frequency, then the flagging table must be copied back to the spectral data base. This will cause the whole band of channels to be flagged as bad when the channel 0 data were flagged as bad.

As well as the FG table, the calibration tables must also be copied to the spectral data. Recall that the interpolated gains as a function of time were written into a CL table(s) attached to your channel 0 data base. This must be copied to the spectral data base.

The procedure CALCOP copies all the relevant tables (BL, CL, FG, and NX), if they exist, from the specified input file (use GETNAME) to the specified output (use GETONAME). Note that CALCOP removes all CL and NX tables attached to the output file before copying.

If you want finer control, do them one at a time with the task TACOP. Select the type of extension table to copy with a command like `inext='cl'`. Copy all the tables of that type by setting `ncount=0` and set `invers=0` and `outvers=0`. Make sure the `key*` adverbs are set at their null values. Run TACOP and make sure the table(s) has arrived at its destination by looking at the header of the recipient file with IMHEAD. Also, remember that if you go back and do some more flagging on the channel 0 data, then copy the FG table across again (you can delete the

old one if you like, but TACOP will just make a new one and the calibration routines look at the highest version of FG tables if flagver=0). There is no need to copy the first CL table, as it is already in place in the spectral data base.

Although, as mentioned above, the bandpass response can be determined without the calibration, you will need it when imaging, and now is a convenient time in the procedure to copy it across.

TACOP	
inname,inclass,inseq,indisk	Set up for file with table to be copied
outname,outclass,outseq,outdisk	Set up for file to copy table to
inext='FG'	Copy FG table(s)
inext='CL'	Copy CL table(s)
invers=2	Copy tables starting at, say, input number 2
ncount=0	Copy all tables
outvers=0	Output version number of first copied table is next highest

11.3 Computation

1. BPASS also comes replete with a vast set of adverbs, so I have also provided a subset of them, at the cost of some flexibility, in the procedure ATBPASS. Once again, it is recommended that you look briefly at BPASS to ascertain the differences. The main change is that in BPASS, you can divide the visibilities by a model, but in ATBPASS you can't. In addition, division by channel 0 is optional in BPASS, in ATBPASS it is enforced. Here follows a description of *some* of the ATBPASS adverbs.

   - inname, inclass, inseq, indisk. Fill these in as usual with GETNAME on the slot containing the spectral data base.
   - calsour selects the sources to use for the bandpass calibration. You may include more than one source if so desired.
   - calcode is the calibrator code for easy selection (for example, perhaps you set calcode='bp' for the bandpass calibrators with SETJY).
   - timerang allows you to select some subset of the bandpass calibrators in time.
   - bchan, echan allow you to determine the response for some sub-portion of the band. The procedure can be very time consuming for big data bases and this may prove useful.
   - solint allows you to specify the solution time interval. −1 means average over all times (you should probably not use more than one source if you use this option, although you might get away with it because of the 'divide by channel 0' option) and 0 means do a scan average. Otherwise, it is the time in minutes.
   - refant should be set to the same one that you used in ATCALIB or CALIB, but it is not vital.
   - in3name, in3class, in3seq, in3disk refer to an external channel 0 data base such as that produced by AVSPC. If in3name etc. is not filled in, then the channel 0 visibilities will be formed from the central 75% of the band. The channel 0 division is *always* performed in ATBPASS, one way or the other.

2. Set ATBPASS in motion with GO ATBPASS and let it grind away. You should run ATBPASS once for each freqid and write a new BP table for each case.

ATBPASS	
inname,inclass,inseq,indisk	Fill in for spectral data base
calsour=' '	Select bandpass calibrator(s) explicitly
calcode='bp'	or with calcode
uvrange=0	Select full uvrange or set in klambda
timerang=0	Select all times
freqid=1	One freqid per run
bchan=0	Select all channels or specify subset
echan=0	
solint=-1	Average all times or
solint=0	Scan average or average
solint=30	for 30 minutes
refant=3	Select the same ref. antenna as in CALIB
bpver=0	BP table version to write
in3name,in3class,in3seq,in3disk	Fill in for channel 0 data base or leave
	blank to use central 75% of band

3. After **ATBPASS** completes, you should examine the bandpasses. This can be done with **POSSM**. Almost all the inputs to **POSSM** can be ignored for this purpose. Just make sure that **aparm(8)=2** (plot the BP table) and **bpver** points at the desired BP table. Depending upon the **solint** used when you ran **ATBPASS**, there may be one or more time entries in the BP table. Use the **timerang** adverb to select which entry you would like to see (for example, to see the temporal stability of the bandpasses), or leave it at 0 to average them all. Make one plot for each of the N antennas by running **POSSM** N times, selecting the desired antenna with **antennas**. If you select all antennas in one go, you get a bandpass averaged over them all. You can display the plot directly on the TV with **POSSM** by putting **dotv=1**, or display the plot file created (**dotv=-1**) with **MIXPL**, **TKPL**, or **TVPL**. The antenna that you picked as the reference antenna should have zero phase across the whole band. Make sure also that the amplitude scale is sensible and that the response peaks at a value close to unity.

POSSM	
bpver=0	Highest BP table version
aparm(8)=2	Plot BP table.
antennas=3	Plot response for antenna 3 say,
antennas=0	or all antennas averaged together

4. You can also check that the application of the band-passes to the data is working correctly. Do this with **POSSM** as well. However, instead of asking it to plot the BP table, you ask it to apply the BP table to the data and plot a spectrum.

5. You can also check that the application of the band-passes to the data is working correctly. Do this with **POSSM** as well. However, instead of asking it to plot the BP table, you ask it to apply the BP table to the data and plot a spectrum. In the example we plot a single baseline of the primary calibrator with the calibration and bandpass corrections applied. We vector average and since the source is a point source in the phase centre, we get a strong signal.

POSSM	
inname,inclass,inseq,indisk	Specifiy multi-source file
source='1934-638',"	Specify source of interest
antennas=1,2	Select baselines of interest
docalib=1	Calibrate data with
gainuse=2	CL table 2
doband=1	Calibrate data with
bpver=1	BP table 1
aparm(1)=1	Vector average
solint=-1	Scan average
dotv=1	Plot on TV

Last update : 27/11/93

12 APPLYING THE CALIBRATION TO THE VISIBILITIES

After you are satisfied with the calibration, you may not want to worry about remembering about all these calibration tables, but would rather simply apply them directly to the data and write a corrected file. This step is also necessary should you want to combine multi-configuration data (see § 14). This is done with the task SPLIT. In addition to applying the calibration, SPLIT extracts individual sources from the multi-source file and writes them to single-source files. The main reason for this is that the most important imaging routines, MX and UVMAP function only on single-source files. Note that there is an imaging task called HORUS (for those who understand NRAO, HORUS was the son of ISIS) which functions on multi-source files (see § 15) as well as single-source files, so that you may not necessarily need to run SPLIT.

- You can apply SPLIT either to the channel 0 data base or to the data base containing all channels if you are interested in the spectral information.
- If you give SPLIT a list of **sources**, then you get one output file per source (the file name is given by the source name).
- The **stokes** adverb allows you to convert to Stokes parameters after applying the calibration. However, the only Stokes parameter that you can get from ATCA data with *AIPS* is that of total intensity (I) (see § 3.2).

  If you are working with a large spectral-line visibility file now is not an unreasonable time to set **stokes='i'** as this converts XX and YY into I thus producing an output file half as big as the input file.

  If you are working with continuum data it doesn't matter much in terms of disk space. You can always ask for Stokes I when you image the data. If the source is unpolarized, and you intend to self calibrate, you are best to leave this adverb blank and pass the correlations in their raw polarization form. However, if the source is polarized and you intend to self calibrate, then you should convert them to Stokes I (this can be done during the self calibration stage but the switch is hidden in one of those **aparm** arrays).
- As usual, you can only get one **freqid** at a time.
- If you are desperate conserve disk space, you might consider dropping channels at the beginning and end of the band with **bchan** and **echan** where the filter response has rolled off. However, note that if you are going to do visibility-based continuum subtraction of spectral-line data (see § 13) it is important to have as many line-free channels as possible.
- It is generally safer to split one IF at a time, as from time to time, there have been problems imaging other than the first IF. This has especially been true if you concatenate multi-IF, multi-channel files with DBCON and then try to image the second IF (see § 14; is fixed in ATNF *AIPS* only so caution is advised). So use **bif** and **eif** to select a single IF at a time.
- Now, to apply the calibration, set **docalib=1**, and set **gainuse** to the version of the CL table that you wish to apply. If you have several CL tables, then make sure that you apply the correct CL table with the correct source selection criteria.
- Leave **dopol** and **blver** at zero as you have not done any polarization or baseline calibration.
- Set **flagver** to point at the desired flagging table.
- If you wish to apply a bandpass calibration, then put **doband** to 1 (average all bandpass solutions in time for each antenna) or 2 (take the nearest bandpass solution in time). The option to set **doband=3** does not nor will it ever work. Set **bpver** to point at the desired bandpass table version. You should do this if you are planning to make a multi-frequency synthesis image from all your channels, or if you are doing spectral-line work.
- If you wish to either retain or write compressed data, set **douvcomp=1**.

Leave the rest of the adverbs at 0 or blank and GO SPLIT. Always check the integrity of the output file by examining the header with IMHEAD to make sure that the axes are labelled as expected (RA, DEC, frequency increments and sign and so on).

SPLIT	
inname,inclass,inseq,indisk	Fill in for spectral or channel 0 data base
sources='1123 − 77','1333 − 33',' '	Specify sources to split or select subset
qual=−1	All qualifiers or specify
calcode=' '	All calcodes or specify
timerang=0	All times
stokes=' '	Pass Stokes parameters as they are
stokes='i'	or ask for STokes *I* only
freqid=3	One freqid per run
bif=1	Select IF
eif=bif	
bchan=0	All channels
echan=0	
docalib=1	Calibrate data
gainuse=2	Specify CL table to apply
dopol=−1	No polarization calibration
flagver=0	Apply highest version FG table or specify
doband=1	Apply bandpass by averaging all times
doband=2	Apply bandpass by finding nearest time
bpver=0	Apply highest BP table version
outclass, outseq, outdisk	Specify out class, seq and disk of each output file, named by source
douvcomp=−1	Select un-compressed data
aparm=0	Defaults OK
chansel=0	Select all channels or specify
aparm(1)=1	channels to AVERAGE with chansel
chansel=8,28,1	average channels 8 through 28

Last update : 27/11/93

13 SPECTRAL-LINE SPECIFIC PROCESSING

13.1 General

This section discusses some of the more complex areas of spectral-line processing. Some of the steps described here are pre-imaging, but some include imaging, and in those cases you should also refer to the imaging section (15) which follows.

The main differences between spectral-line imaging and continuum imaging are a) the amount of data you produce is much greater, b) you must remove any putative continuum emission, and c) you must try to keep track of velocities and frequencies more precisely.

Much of the processing discussed below can be done on either multi- or single-source files, and I discuss both options. However, it gets rather confusing as to what order things should be done in when there are many paths through the maze. I suggest that the clearest route following calibration and bandpass computation is generally `SPLIT` (applying editing, calibration and bandpass), `UVLSF` or `UVLIN` (visibility based continuum subtraction), `CVEL` (velocity alignment), `DBCON` (combine data), and then imaging. Bear this in mind when reading what follows.

13.2 Disk management

Spectral-line data sets can be very large, and you may find, even with a Gbyte all to yourself, that you are short of disk space. There are a few things you can do to alleviate this problem.

Firstly, you should seriously consider averaging the data in time. For many purposes, the 15 s samples that the ATCA usually offers by default (although you may have set this to something longer when you observed) are much too short; see the discussion in § 6 to see how to do this with `UVAVG` and what penalty, if any, you pay. Secondly, you can also save space by compressing each complex visibility into half as many bits. See the discussion of the adverb **douvcomp** in `ATLOD` in § 4 for what you sacrifice in doing this. This compression is available in many *AIPS* tasks, and there is a specific task, `UVCMP`, to compress or uncompress data. Thirdly, you can discard channels that are of no further use to you. For example, when you run `SPLIT`, you might decide to drop channels at the beginning and end of the spectrum which are particularly noisy (because of the multiplication by the bandpass correction). However, if you are going to do visibility based continuum subtraction (see § 13.3.3), then you should not do this, as you need to retain as many line-free channels as possible.

Naturally, you would want to do these things as early in the processing chain as is practicable. For example, you might run `UVAVG` straight away after `ATLOD`, or, if there are large amounts of corrupt data, after you have edited the data and run `SPLIT`.

13.3 Continuum subtraction

Each visibility has spectral-line and continuum components. You must remove the continuum to be able to study the spectral-line accurately. In § 11 I described how to make a bandpass calibration; this *must* be applied before you attempt to subtract the continuum. There are several continuum subtraction methods now available in *AIPS*. I will discuss them below in quasi-historical order, but I suggest that you read about all the methods before having a go.

I also recommend that you consider the visibility-domain subtraction method as your first line of approach; this is the last method I discuss, and it is probably the least intuitively obvious. However, it is the easiest to use, the fastest, and the most accurate.

13.3.1 *Subtraction in the image domain by subtracting dirty images*

The simplest conceptual approach is to make the spectral-line cube, and then subtract from all planes (channels) in that cube an image made from the channels containing continuum only. You can see which channels have line and which have continuum by making a spectrum either with `ISPEC` on the cube, or `POSSM` on the visibilities (but you only get the spectrum at one location on the sky).

This method has two main problems.

1. The beam changes with frequency. Thus, the continuum image made from the line-free channels has a beam that does not match any of the line-only channels exactly. This does not matter if the ratio of the total

bandwidth to the central frequency is small, viz., $\Delta\nu/\nu_0 << 0.01$, and the continuum emission is not too strong, as the differential beam structure will be negligible.

2. This method does not, in general, account for any non-zero spectral index in the continuum emission.

If neither of these problems affects your data, you could proceed by first making the spectral-line cube containing all channels (both continuum and line) as described in §15.6. Next, compute the averaged continuum image with SQASH.

SQASH	
inname,inclass	Input cube
inseq,indisk	
outname,outclass	Output averaged
outseq,outdisk	continuum image
blc=0,0,N	Specify channels N
trc=0,0,M	to M to average
bdrop=3	Squash axis 3
dparm=1,1	Average and ignore blanks

It may be that you have continuum at several places in the spectrum. In that case, run SQASH successively on each region, and then average the SQASHed images with COMB. Finally, subtract the continuum image from the cube with COMB.

COMB	
inname,inclass	Input cube
inseq,indisk	
in2name,in2class	Input continuum
in2seq,in2disk	image
outname,outclass	Output subtracted
outseq,outdisk	cube
blc=0	Full region
trc=0	
opcode='sum'	Summation operation
aparm=1,−1	Negate continuum

Note that this method requires you to have two copies of your cube; this can be a severe problem if the cubes are large and disk space is tight. You should examine the image for obvious signs of differential beam effects. One way to check this empirically is to make beams from channels as widely separated as possible, and then subtract them with COMB. Examine the output image and find the peak value. See if this is large enough to cause you difficulty for the science you are trying to extract.

An alternative way to achieve the same result, but which requires less disk space is as follows. Instead of making a dirty cube from all the channels, first use AVSPC to average the visibility channels containing continuum only signal into a separate visibility data base (see § 6). Then make the dirty continuum image from this. You then make the dirty spectral-line cube (see § 15.6) from only those channels containing the spectral-line. Then you proceed with COMB as described above.

In the special case that your continuum channels are symmetrically placed about the spectral-line, and the line is sufficiently narrow, this method will cope with a non-zero spectral index, to first order. But this condition is not always the case.

13.3.2 *Subtraction in the image and visibility domains combined*

To overcome the problem of the beam changing with frequency, van Gorkom and Ekers (1983) suggested a method which models the continuum emission, Fourier transforms the model correctly for each channel (thus dealing with the frequency-dependent beam problem), and subtracts the Fourier transform from the visibilities for each channel. You then re-image the subtracted visibility data base. This procedure is quite CPU intensive, but, provided you can make a good model of the continuum emission, produces a better result than the simple image subtraction method described above, especially if the line is weak and the continuum is very strong. If the continuum emission is very simple, you may be able to model it by a collection of discrete point sources. More typically, we use CLEAN (see § 16 describing deconvolution of images) to produce this model. It is sometimes

necessary to estimate a residual continuum image after this procedure and subtract it as well. This method suffers from the following difficulties.

1. It assumes that the spectral index of the continuum emission is zero.
2. The model may not fully represent all the continuum structure. For example, CLEAN does not perform well on large regions of extended emission. You might also try to produce the model with a maximum entropy technique such as VTESS in *AIPS*.
3. The computational cost is large (deconvolution plus lots of Fourier transforms).

Let us proceed with the implementation of this technique. First, you must produce a dirty image of the continuum emission with the methods presented in §15. It may be necessary for you to image all the channels so as to ascertain which channels are line free and suitable for continuum (use ISPEC to plot spectra averaged over some spatial window or POSSM directly on the visibilities).

Next you should deconvolve the continuum with one of the CLEAN programs as described in §16 and produce a model of the continuum emission. The CLEAN model is the CLEAN component list. You must decide whether all of the CLEAN components that CLEAN found are suitable to be included in the model, or whether you should truncate the list at some point (for example, you may have CLEANed into the noise and there is not much point to Fourier transforming and subtracting noise).

The Fourier transformation of the model and subtraction from the visibilities is done with the task UVSUB. You should run this only on a single-source visibility data base. If you don't have one, go and make it with SPLIT, as described in § 12.

UVSUB	
inname,inclass	Input visibility
inseq,indisk	file
nmaps=1	One input model file
channel=0	Subtract from all channels
in2name,in2class	Image associated with CLEAN
in2seq,in2disk	component list
outname,outclass	Output subtracted
outseq,outdisk	visibility file
bcomp=1	Start subtracting CLEAN components at this one
ncomp=0	Subtract all components ?
cmethod= ' '	Allow program to choose subtraction method
cmodel='COMP'	Model is CLEAN component list
factor=1	Subtract components
opcode=' '	Subtract components
smodel=0	Used for point source models
baddisk	Keep scratch off NFS mounted disks

You can now image the subtracted visibility data base with the methods described in § 15.6 to see the line-only signal.

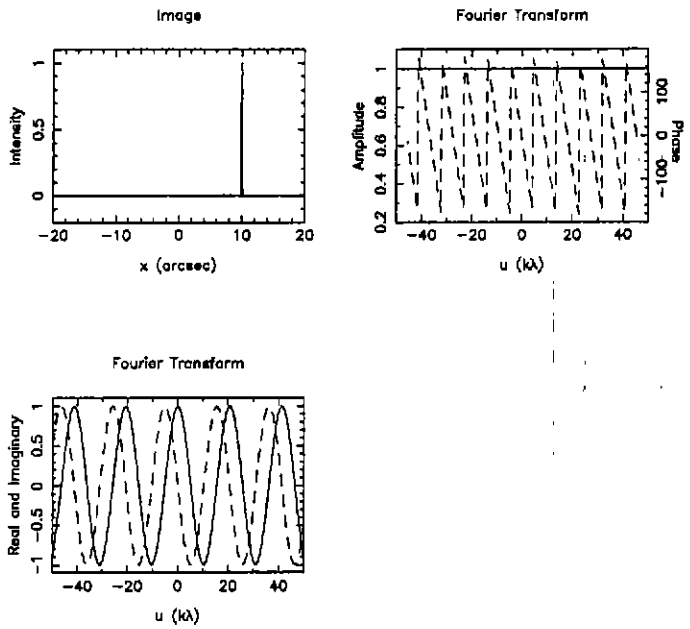
13.3.3 *Subtraction in the visibility domain*

This technique is a recent development (see van Langenvelde and Cotton 1990, Cornwell *et al.* 1991, and Sault 1993). In spectral-line work, the bandwidth is generally fairly narrow, such that an acceptable approximation to the continuum emission is a linear function of frequency at each point in the image.

Each visibility consists of measurements in a band of frequency channels. The value of the visibility at each channel is one sample of the Fourier transform of the image. At the centre of the band (frequency ν_0) we sample the Fourier transform at the location (u_0, v_0); recall in one dimension, $u = D/\lambda = D\nu/c$. However, as the frequency changes up and down the band, we move radially away from (u_0, v_0) in the (u, v) plane.

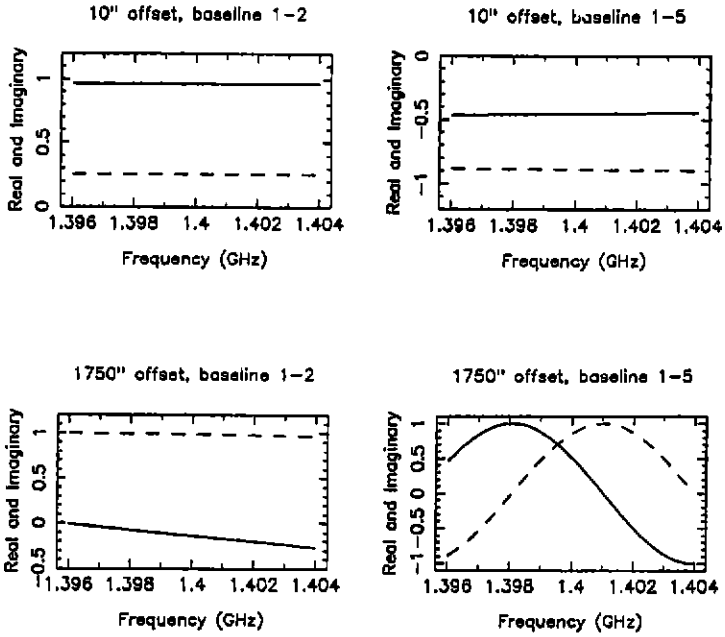
Because Fourier Transforms are linear operators, the visibility can also be approximated by a linear function with frequency (the continuum) with the spectral-line superposed on top, provided the visibility function changes only slowly across the bandwidth. We can then fit (by least squares) the continuum-only channels with a linear function. We fit the real and imaginary parts separately, and each visibility is fitted and subtracted independently. This procedure is very fast and requires very little user intervention, apart from working out which channels contain continuum. In general, it is better to fit to the real and imaginary parts than to fit to the amplitude and phase as the whole process remains linear. The non-linear fit to amplitude and phase couples the errors on all sources and also produces an amplitude bias at low signal-to-noise ratios. Also, the more line-free channels you have, the better the fit tracks the continuum.

As an example, consider, in one dimension, a point source offset from the phase centre. The visibility, as a function of spacing, is a constant amplitude and a linearly varying phase. Alternatively, the real and imaginary parts are sinusoids with spacing. This is demonstrated in the following figures which show the image, and the Fourier Transform. Because u varies linearly with frequency, you can think of the abcissa as reflecting a change in frequency or a change in telescope spacing (I have assumed that the spectral index is zero). Clearly, if the change in frequency across the band is sufficiently large, you will sample a significant fraction of a period of the real and imaginary sinusoids, such that the linearity assumption is invalidated.



To clarify this, consider some pseudo ATCA data for two offset point sources. I will use a typical spectral-line setup of five antennas in a 3-km configuration, $\nu_0 = 1.4$ GHz, $\Delta\nu = 8$ MHz, and 256 channels. In this case, the sysnthesized beam is $\theta_s \approx 10$ arcsec.

The following figure shows spectra for two point sources (upper panel offset 10 arcsec from phase centre, lower panel offset 1750 arcsec from phase centre) from one 10 s integration and for the shortest (1–2; 183 m) and longest (1–5; 2983 m) baseline.



For the source that is only 10 arcsec from the phase centre, the assumption that the real and imaginary parts of the visibility are linear functions of frequency is a good one. However, for the distant point source, the longest baseline clearly exhibits non-linear visibilities. I leave it as an exercise for the reader to work out why the short baseline is less affected than the long baseline.

Like all methods, this one cannot be used blindly, but the restrictions are simple. Clearly, as the figures show, the approximation that the visibility is a linear function with frequency will break down if the visibility function varies too quickly as a function of radial distance in the (u, v) plane (i.e. with frequency). This means a restriction on the field of view that you can image well. Sault (1993) shows that for a point source of brightness S Jy at location (l_0, m_0) (with respect to the phase centre), the residual continuum following a first order fit is given by

$$\frac{S\pi^2}{36}\left(\frac{\Delta\nu}{\nu_0}\right)^2 \frac{l_0^2 + m_0^2}{\theta_s^2}$$

where $\Delta\nu$ is the total bandwidth, and θ_s is the FWHM of the synthesised beam.

You might like to be aware that the *MIRIAD* task that implements these ideas can in fact fit higher order polynomials than first (removing the need for the linearity of the continuum with frequency assumption). This allows you to subtract the continuum more accurately at larger distance from the phase centre than the first order fit. There is also a *MIRIAD* task that produces an error image as well to help you assess the reliability of the fit. See the *MIRIAD* manual and the paper by Bob Sault for more information.

The fact that the residual continuum is a function of the distance of the point source from the phase centre should not be a surprise. As (u, v) coordinate is proportional to frequency, the channels in a visibility spectrum are sampling slightly different locations in the (u, v) plane. For a point source (which we assume to have a spectral index of zero, for simplicity) will have a visibility function whose real and imaginary parts are sinusoids. The fraction of a period of this sinusoid contained within the (u, v) coordinates spanned by a visibility spectrum will be directly proportional to the distance of the point source from the phase center. The further the point source is from the phase centre, the greater proportion of a sinusoid period is present in the visibility spectrum, and the poorer the approximation of the visibility spectrum by a low order polynomial.

This method of subtraction has the advantage over other approaches that it is generally more robust to a large variety of systematic errors (such as antenna gain errors). It also copes with continuum emission that has a non-zero spectral index, is much cheaper computationally and much easier to use than the other methods.

Now let us continue with the implentation of this method in *AIPS*. There are three choices, `UVBAS`, `UVLIN`, and `UVLSF`. In standard *AIPS*, `UVBAS` should be avoided, as it fits to the visibility amplitude and phase (see the discussion above). The ATNF version has been changed to fit real and imaginary, but to avoid confusion, I suggest you keep clear of `UVBAS`. The tasks `UVLIN` and `UVLSF` have basically the same functionality, but differ slightly in their packaging. `UVLSF` has the advantage that it can output the fit into a separate data base, which you can then image. This gives you an excellent indication of how the method has performed, as well as providing you with a good continuum image. On the other hand, `UVLIN` offers more flexibility when selecting the continuum

channels, plus a useful shifting option (see below). I will show example adverb boxes for them both.

Both UVLIN and UVLSF will work on multi- or single-source files. However, there are no source selection adverbs, and more importantly, you can't apply the FG table so that bad channels within the range you designate as line free cannot be excluded easily from the fit. Thus, you should probably SPLIT the desired source into a single-source file (see § 12) first, applying all calibrations and flags.

Note however, that you may want to run CVEL (offline Doppler tracking; see § 13.5) and there are some advantages to doing this on a multi-source file. Countering this is the fact that you are best subtracting the continuum before running CVEL if the continuum is strong and the continuum subtraction programs are really aimed at single-source files !! Good design isn't it ? See the extra discussion in §'13.5 for more help.

UVLSF is fairly straightforward to use.

- Use bchan and echan to designate the range of input channels that will be written to the output file.
- chansel is used to specify which channels are continuum channels and should be fitted. You can specify up to 10 groups of three numbers; each group is the start, end, and increment. Thus, chansel=5,9,2, 20,30,1, 400,410,2 would select channels 5, 7, 9, 20–30, 400, 402, 404, 408 and 410 as continuum. If you leave chansel=0 all channels are selected. It is unlikely that you would want to do this.
- Set dooutput=1 if you would like to write a file containing the continuum visibilities (i.e. the fit). UVLSF will choose the name of this file as the outname you have set and an outclass = 'BASFIT'. Select the channel at which the fit is to be evaluated for this continuum file with channel.

UVLSF	
inname,inclass	Input visibility
inseq,indisk	file
outname,outclass	Output subtracted
outseq,outdisk	visibility file
bchan=1	Write all channels to the
echan=0	output subtracted file
chansel=5,30,1	Select groups of continuum channels to fit (start, end, increment)
dooutput=1	Write the fit into an output file
channel=128	Select a channel at which to compute the continuum fit

UVLIN is not quite so straightforward, as it offers more flexibility with regards continuum channel selection. It can also be used to flag channels as bad, based upon their discrepancy from the fit rather than subtracting the fit.

- If the channels that you wish to designate as continuum are not easily delineated by a series of groups of contiguous channels, you can specify a weight (either 0 or 1) for each channel. To do this you create a tex file of two columns (channel and weight), which is read by UVLIN. If you leave out a channel, it is given zero weight (excluded from the fit), so all you really need do is list the channels you wish to fit and put a '1' in the next column (see the EXPLAIN file for more details). As usual with *AIPS*, the text file must have an upper-case name, and should be pointed at with an upper-case environment variable.
- If you leave file blank, you can specify the desired continuum channels with the nboxes (number of channel ranges) and boxes (start and end channels for each range) adverbs. You can have up to 20 channel ranges. All channels will be used for the fit if both infile and nboxes=0; this would probaby not be very useful.
- From the discussion earlier, it is clear that UVLIN will work best for a point source at the phase centre. Thus, if you have a dominant point source in the field, and it is significantly distant from the phase centre, shift offers the chance to shift it to the phase centre, to do the fit and subtraction, and then to shift it back again (of course, the whole field is shifted, not just the point source). You specify the shift in the RA and DEC directions in arcseconds (the sense is the same as in UVFIX). The amount to shift in RA is not the distance on the tangent plane. It is the amount of polar rotation. Thus, for small shifts, you can measure the shift from an image in pixels, multiply by the pixel increment to get arcseconds on the tangent plane, and then divide by the cosine of the declination. Note that the *MIRIAD* version of this program also has this shifting facility.
- UVLIN has the useful facility of flagging based on the discrepancy of the data from the fit. flux specifies the maximum error allowed in Jy. This can be useful for removing narrow-band interference. The number

specified is the limit per visibility weight (w) (nothing to do with the 0 or 1 weights specified in the text file). `flux` is then adjusted by $1/\sqrt{w}$ to correct for integration time differences that may be present in the data. Note that the flagging is only performed on the channels for which the fitting is done. You should be very conservative in setting this value; something of the order of 10 times the theoretical noise per visibility would be reasonable.

Obviously, there is not much point to flagging data after the fit and subtraction; the fit would have been biased already. The flagging option should be used in conjunction with `docont=1`. This tells `UVLIN` to retain the continuum rather than subtract it, so that the output data-base contains all the data with the bad channels flagged (the weights will be set negative).

UVLIN	
inname,inclass	Input visibility
inseq,indisk	file
outname,outclass	Output subtracted
outseq,outdisk	visibility file
infile='AREA:WEIGHTS'	Specify a file with the the weights for each channel
nboxes	Specify number of pairs of channels if **infile** blank
boxes	Specify **nboxes** start and end channels
shift	Specify RA & DEC offsets of dominant point source from phase centre
docont=−1	Subtract the fit or
docont=1	Retain the continuum and
flux	Flag channels discrepant from the fit by this amount

13.4 Velocities and reference frames

There is confusion over the velocity definitions in spectral-line astronomy. This is because we have two definitions of radial velocity and neither is particularly meaningful, physically.

The *radio* velocity definition is

$$v_{\rm rad} = c\frac{\nu_{\rm rest} - \nu}{\nu_{\rm rest}},$$

where $\nu_{\rm rest}$ is the rest frequency of the line, ν is the observed sky frequency, and c is the speed of light. This expression is often used by Galactic radio astronomers (astronomers from our Galaxy ?) but is now deprecated by the IAU. For reference, this means that the radio velocity increment is

$$\Delta v_{\rm rad} = \frac{-c}{\nu_{\rm rest}}\Delta\nu,$$

and the frequency in terms of the radio velocity is

$$\nu = \nu_{\rm rest}(1 - \frac{v_{\rm rad}}{c}).$$

The *optical* velocity definition is

$$v_{\rm opt} = c\frac{\nu_{\rm rest} - \nu}{\nu} = cz,$$

where z is the redshift. This expression is often used in extragalactic astronomy. The optical velocity increment is

$$\Delta v_{\rm opt} = \frac{-c\nu_{\rm rest}}{\nu^2}\Delta\nu$$

and the frequency in terms of the optical velocity is

$$\nu = \nu_{\rm rest}(1 + \frac{v_{\rm opt}}{c})^{-1}.$$

Neither of these velocity definitions has any physical meaning unless the radial velocity is truly small compared with the speed of light. These expressions are not interchangeable; you must use the correct expression with the correct velocity definition to get a meaningful frequency. Note also that the optical velocity increment depends upon the observing frequency, whereas the radio velocity increment depends on the *rest* frequency.

We must also define our velocities in some reference frame; the velocity with respect to the observer is not usually of particular interest to anyone (except for solar system work). It is convenient to define the velocity with respect to some astronomical reference frame and the following table lists the standard ones (this table is from the spectral-line chapter of the NRAO Imaging Workshop book by Pjotr Roelfsma).

Velocity Rest Frames		
Correct for	Correction (km s^{-1})	Rest Frame Name
Nothing	0	topocentric
Earth-rotation	< 0.5	
Earth's motion around earth-moon barycentre	< 0.013	geocentric
Earth's motion around Sun	< 30	heliocentric
Sun's motion around solar system barycentre	< 0.012	barycentric
Solar motion	< 20	local standard of rest
Galactic rotation	220	galactocentric

In Galactic work, the local standard of rest (LSR) frame is generally used. In extragalactic work, the barycentric frame is generally adopted (although it's often misnamed the heliocentric frame).

13.5 Associating a velocity with a channel

We are ultimately interested in knowing the velocity of each spectral channel. This has two components; the velocity of the observatory in some reference frame in the direction of the source, and the velocity of the source with respect to that frame (the redshift).

The ATCA produces channels that are equally spaced in frequency, and when imaged, are generally stored along the third axis of a cube (3-D image). In the *AIPS* header, such an axis would be labelled 'FREQ'. If we then converted the frequencies to radio velocities, the axis would be labelled 'VELO'. Note that the frequency increment can be converted to a unique velocity increment in the radio definition. However, if we converted to optical velocities, the above expression for the optical velocity indicates that the velocity increment would change along the axis because the observed frequency (rather than the rest frequency) is the denominator (see equation above). In this case, the axis is labelled 'FELO', and the axis velocity increment worked out at the central observed frequency. Clearly this can give only an approximate representation of how the optical velocity changes along the axis. I leave it to the reader to work out how the error in the optical velocity varies with offset from the reference pixel (where the optical velocity is defined correctly); it can be quite substantial.

With the ATCA, on-line Doppler tracking is not used. This means that you have a well-defined sky-frequency axis with each observation, but not a well-defined velocity axis. Because of the earth's rotation, and the earth's motion about the Sun, the velocity that is correctly associated with a particular frequency channel will change with time.

If you have a single observation (12 hours say), then the velocity change at a given frequency owing to the earth's rotation is less than about 0.5 km s^{-1} (see the table above). For broad-line features generally encountered in extragalactic work, this velocity shift can be ignored (as you would have observed with a channel width much larger than this shift). However, for narrow-line Galactic sources such as masers, you may need to align the data so that each channel is correctly aligned with a specified velocity rather than frequency. Even if you are doing extragalactic work, you will definitely need to make this alignment if you are combining data taken on different dates (although I have never been on a date with a synthesis telescope) owing to the earth's orbital motion.

The procedure to align the data requires two steps as follows.

1. You should already have set the velocity scale, to first order, in the multi-source file via the SU (source) table and `SETJY` (see § 5). When you run `SPLIT`, and produce a single-source file, the velocity information

in the SU table is maintained in the header of the new output file as an 'alternative' definition. You can list it with INHEAD. However, the frequency axis will be retained as the primary regular spectral axis. If at any time you want to switch the labelling of the spectral axis from frequency to velocity, you can issue the command ALTSWTCH, which toggles between the two. Note that the imaging tasks require that the spectral axis be labelled as frequency, not velocity. I suggest you restrict the use of ALTSWTCH to your spectral-line cube(s), rather than your visibility file(s).

2. The task CVEL is used to shift each visibility spectrum so that a designated channel becomes associated with a specific velocity. By running CVEL in the same fashion on all the files that need to be combined, they can be aligned ready for DBCON. You may also use CVEL on a single file to remove the small misalignents owing to the earth's rotation. CVEL is extremely CPU intensive; it Fourier transforms each visibility from the frequency domain to the lag domain, applies the necessary 'phase' correction, and then Fourier transforms back again to effect the frequency shift (this is an application of the shift theorem of Fourier theory). This large computational load is another reason why you might like to time average your data before running CVEL. CVEL does a lot of complicated calculations to work out how much the shift should be, and these depend upon correct information being found in the antenna (AN) table (such as the antenna locations) as well as the source (SU) table. Try very hard to get it right the first time; you will tire quickly of running CVEL.

   In principle, CVEL should be able to work out for itself the velocity of the observatory at any given time in any given frame in the direction of the source of interest. By knowing the rest frequency of the spectral line, it is then possible to compute the velocity of the source in the relevant frame. However, *AIPS* and CVEL do not seem to be this clever. You have to tell CVEL which channel in the spectrum you would like to be associated with which source velocity. When you set up your ATCA observation, you will have worked this out because you had to set the central observing frequency to correctly deal with the source velocity (that is you applied the velocity definition equations given above). Note that *MIRIAD* correctly deals with velocities and the CVEL operation is done transparently by an interpolation scheme.

   CVEL works on single- and multi-source files. Unlike older versions, CVEL now (15JUK93 *AIPS*) reads FG tables, so that CVEL can be usefully run on multi-source files. This has two advantages. First, CVEL wants the apparent source coordinates which are stored in the SU table. These are lost in single-source files which contain only the mean coordinates. Second, you need less disk space if you stick with the multi-source file, assuming the file is dominated by the source(s) you plan to shift. For multi-source files, the FG table is applied to the data as it is read in and completely flagged visibilities are dropped from the output. Partially flagged visibilities are dealt with by setting the weight negative for the relevant channels. Therefore, there is no output FG table. Any source that is not selected for shifting is copied unshifted to the output multi-source file (but still the flags will have been applied).

   However, running CVEL on multi-source files has the following disadvantage. If you have a strong continuum, it is important that you subtract it before running CVELThis is because the cutoff in the spectrum at the edges from the large continuum value to zero will cause a ringing at the edges of the spectrum after CVEL. But, the visibility-based continuum subtraction programs are really designed for single-source files, as they have no calibration adverbs, including applying the FG table. Thus you cannot easily exclude bad channels from the designated continuum sections of the spectra.

   On balance, I would say that the error you make from having the source positions slightly wrong (J2000 coordinates as opposed to apparent coordinates) is sufficiently small that if you have strong continuum, you should generate a single-source file first (with SPLIT), do the continuum subtraction and then run CVEL on that subtracted single-source file (disk space allowing).

   Let us proceed with some of the adverbs to CVEL. Their use depends upon whether you are working on a single- or a multi-source file.

13.5.1 *Multi-source files*

It is assumed that the correct velocity information is in the SU table. If it is missing, then enter it with SETJY as described in § 5.

- Select the sources that you wish to shift (there is not much point to shifting calibrators) with the **sources** adverb. The unselected sources are copied unshifted to the output file.
- You may select by time range as well with **timerang**.
- Select the desired frequency with the usual adverbs **selband**, **selfreq**, and **freqid**.

- It is unlikely that you will have more than one subarray in the data at present, unless somehow you have already run DBCON (which you shouldn't have; run CVEL first and then DBCON). Select all subarrays for shifting by leaving subarray at zero.
- flagver If −1 no flagging is applied, otherwise specifies the FG table to apply, and flagver=0 means the highest FG version available. There is no FG table associated with the output file.
- It is *very* important that the bandpass correction is applied before you shift the spectra. The bandpass correction applies to the data in their original fixed frequency frame. doband can take the value 1 (average all entries in the BP table for each antenna) or 2 (use nearest entry). I suggest doband=1. Use bpver to specify the version of the BP table to apply. You cannot apply the bandpass correction with any validity after running CVEL.
- In principle, there can be source-specific Doppler offsets hiding in a CL table. This might be the case with VLA or VLBI data, but it is not the case with ATCA data. Set gainuse=−1.
- Leave aparm(1) through to aparm(6) at zero for multi-source files.
- Set aparm(7)=0 for ATCA data (the ATCA and VLBI arrays have a similar geocentric coordinate system) and leave aparm(8)=0 so that you are warned of unduly large shifts.

CVEL	
inname,inclass	Input visibility file
inseq,indisk	
outname,outclass	Shifted visibility file
outseq,outdisk	
sources 'NGC9899',' '	Specify sources to shift
timerang=0	Select all times
freqid=1	Select frequency ID
subarray=0	Shift all subarrays
fgver=0	Highest FG table
doband=1	Apply band pass correction
bpver=1	Select BP table version
gainuse=−1	Nothing hiding in CL table
aparm=0	Defaults correct

13.5.2 *Single-source files*

You *must* have already applied the calibration and editing tables (CL, BP, FG) with SPLIT before you attempt to run CVEL on a single-source file. You can supply the velocity information in two ways. Either it is in the header as the alternative definition already (via SETJY and SPLIT) or you can supply it with the aparm adverb. Whatever you supply with aparm(1) to aparm(6) overides anything already in the header.

- sources, selband, selfreq, freqid, flagver, doband, bpver, and gainuse are irrelevant to single-source files.
- timerang, subarray, aparm(6), and aparm(8) act as described above for multi-source files.
- aparm(1) specifies the velocity (m s^{-1}) of the channel designated by aparm(2).
- aparm(3) indicates what reference frame the velocity is in; 0 means LSR and 1 means heliocentric. aparm(4) indicates what velocity definition the velocity is in; 0 means optical, and 1 means radio.
- The sum of aparm(5) and aparm(6) gives the restfrequency of the line in Hz.

CVEL	
inname,inclass	Input visibility file
inseq,indisk	
outname,outclass	Shifted visibility file
outseq,outdisk	
timerang = 0	Select all times
subarray = 0	Shift all subarrays
aparm(1)=5353E3	Specify velocity at
aparm(2)=256	this channel in
aparm(3)=0 or 1	this reference frame and
aparm(4)=0 or 1	this velocity definition
aparm(5)=1667.0E6	Rest frequency of line
aparm(6)=0.359E6	
aparm(7)=0	
aparm(8)=0	

3. You have now produced a shifted data base. CVEL will have told you (unless you set aparm(8)=1) if it considered any of the shifts to be unduly large. You can use POSSM to check some spectra before and after the shift (if you have a strong line that is seen easily). The frequency axis will be left labelled as it originally was (otherwise you will be unable to combine it with like data with DBCON (see § 14). You should be aware that this frequency axis description is now meaningless, as each spectrum has been shifted relative to every other one. It is the velocity description that is now meaningful, although, as mentioned above, only a velocity axis in the radio definition is entirely self-consistent.

4. There is a somewhat nasty-in-principle but generally negligible-in-practice side-effect of all this shifting about of visibilities, when it comes to imaging the data. As mentioned above, the labelling of the frequency axis is now less than precise. However, it is very important that the imaging tasks know how to work out the frequency of each channel. This is because the baseline coordinates, u and v, are stored in wavelengths at the reference frequency, ν_0. When the visibilities at each channel are Fourier transformed into an image, the imaging task must be able to compute what u and v are for that channel from the reference values, u_0 and v_0. Before CVEL is run, this is straightforward given the channel number, reference pixel, reference frequency, frequency increment, u_0 and v_0. But after CVEL this simplicity is lost. There is no scale factor that can be applied to u_0 and v_0 (which are stored in each visibility), so that a correct u and v would be computed for each channel from the mislabelled frequency axis. This problem is not addressed by $\mathcal{AIPS}$, but as shown below, it doesn't generally matter because in practice the error is usually negligible.

   In one dimension, the baseline coordinate is $u = u_0\nu/\nu_0$. Let us define a true $u^t = u_0\nu^t/\nu_0$ computed from the true frequency at that channel, ν^t, and a false $u^f = u_0\nu^f/\nu_0$ computed from the false frequency, ν^f as given by the mislabelled header. Each visibility has been shifted by some amount, $\delta\nu$, so that $\nu^t = \nu^f + \delta\nu$. Therefore, the error you make in computing u with the wrong frequency is $\delta u = u_0\delta\nu/\nu_0$. Now, what really counts is whether this would cause the visibility to fall into a different (u, v) cell when you grid the visibilities ready for the FFT. The size of a grid cell is $\Delta u = (N\Delta x)^{-1} \approx 3u_{\max}/N$ where Δx is the image cell size in radians, and $u_{\max}$ is the longest spacing. Therefore, if we confine our interest to the longest baseline where the effect is worst, $u_0 = u_{\max}$, and the fractional error is

$$\frac{\delta u}{\Delta u} = \frac{\delta\nu}{\nu_0}\frac{N}{3}.$$

   Now about the largest shift of the spectrum that you should encounter is that caused by making observations six months apart, so a shift of 30 km s^{-1} is an upper limit. This gives $\delta\nu/\nu_0 \approx 10^{-4}$ so that

$$\frac{\delta u}{\Delta u} < 10^{-4}\frac{N}{3}.$$

   For a maximum image size of 4096 pixels, this leads to $\delta u/\Delta u < 0.1$. Thus, in general, one can ignore the error, although for very large images and a need for high dynamic range, you might start to incur a small effect.

Last update : 27/11/93

14 HOW TO COMBINE MULTI-CONFIGURATION DATA

Because the ATCA has only a modest number of baselines, it will often be essential that you combine data that come from a different time and configuration. This is done to improve the *uv* coverage of the observation. In principle, you could combine multi-source data bases with DBCON, and calibrate the concatenated data base. However, for a number of reasons, I think this approach is more trouble than it is worth, and I suggest that the data for each configuration should be calibrated (ATCALIB, ATCLCAL etc.) separately, split (SPLIT) off into a single-source file, and then combined (DBCON). The imaging tasks can then be run on the combined data base (see § 15). I do not recommend concatenation of multi-source files after they have been calibrated, as the calibration tables are not always correctly merged. I also suggest that you combine TB-sorted data, as this should be the order of your multi-source file. It doesn't matter what time order the different configuration single-source files are concatenated in.

Recall that SPLIT applies the calibration directly to the visibilities and builds single-source files for *one* source and frequency. The only calibration table that might become associated with the single-source file in the future is the SN (solution) table, because new ones will be created in the self-calibration procedure by CALIB (see § 17).

14.1 Concatenation

Before you attempt to use DBCON, I suggest that it would be useful to read the DBCON EXPLAIN file. DBCON will concatenate two single-source files. To combine more than two requires you to run DBCON a number of times, always appending the new file to the output of the previous concatenation.

DBCON can make some checks to make sure that you are not combining data that it thinks do not merit concatenation. It can check consistency of frequency and position. At this point, however, I am only considering the combination of data for the same source at the same frequency from different configurations and observing dates. Discussion of how to combine data of different frequency is delayed until the discussion of imaging (§ 15).

- The dopos adverb is used to control consistency checks. Turn on the frequency check with `dopos(2,1)=1`. Set `dopos(1,1)=1`, and DBCON will check that the data bases have the same phase centre. If they don't, it will shift the second set to match the first. This is so that observations of the same field but with different pointing centres can be combined. Of course, for this to be successful, the primary beam correction difference must be negligible. If you try to combine different sources, you will notice messages about phase shifts, but it will still do it and you will be in a mess.
- You must be careful with the adverb doarray, which allows data from different configurations to retain their identity. It is *very* important that this identity be retained, otherwise, the self-calibration procedure, which must solve for the antenna gains as a function of time (similarly to the normal point source calibration procedure), will get more than confused; put `doarray=-1`. What really happens is that the input data bases are separated by 5 days in time in the output file, and they are treated as separate sub-arrays. This means that the output file will contain as many antenna (AN) files as there are input data bases to be joined up. The version number of the AN file is its sub-array number. You can often point at separate sub-arrays with the subarray adverb that many *AIPS* tasks recognize.

DBCON	
inname,inclass	First input file
inseq,indisk	to combine
in2name,in2class	Second input file
in2seq,in2disk	to combine
reweight=0	Leave weights alone
dopos(1,1)=1	Check phase centres
dopos(2,1)=1	Check for frequency agreement
doarray=−1	Separate input files in time by 5 days

The end result of all this should be one single-source file (for each source and frequency) containing data from two or more observing runs.

If you are combining multi-channel data bases (e.g., for spectral-line work), you can only combine data bases with identical frequency axes. This means they should have the same number of channels and the header items describing the axis should be the same. This is why CVEL does not change the frequency axis description.

14.2 A warning for users of NRAO *AIPS*

Finally, be aware of the following problem if you are NOT using ATNF *AIPS*, but are using standard NRAO *AIPS*. If you combine multi-channel and multi-IF files with DBCON and the frequency increment of the IFs are not all the same, then IFs other than the first may be a problem when you image. With the ATCA, you might have, for example, 13 and 20 cm as the two IFs, and the frequency increments might be 4 MHz and −4 MHz respectively. NRAO *AIPS* DBCON converts the data to have a frequency axis reference pixel of 1.0 and it scales u and v accordingly, but does not adjust the frequency offsets in the FQ table for each IF to reflect the new reference pixel. If the frequency increments of all the IFs were the same that would not matter. However, for ATCA data where they may differ, this means that if you now attempted to work out the frequency of a given channel from any IF other than the first using the reference pixel, reference frequency, frequency offset and increment stored in the FQ table, you would get the wrong value. This will cause a task such as MX when trying to do multi-frequency synthesis imaging (see § 15) to get the wrong scale for any IF other than the first.

Our solution with ATNF *AIPS* is to prevent DBCON from making that conversion to reference pixel 1.0. A better solution (and harder to implement) would be to write a correct FQ table on output from DBCON with the offsets appropriately changed. As I have been unable to convince the NRAO *AIPS* team (what little is left) to do anything about this problem, I am forced to write this stupid section in this manual.

14.3 Conversion of single-source files to multi-source files

It is possible to convert single-source files back into multi-source files. One reason for doing this is that the self-calibration procedure will then just generate the usual SN and CL tables. If you proceeded with single-source files, then a new copy of the data base is produced for each application of CALIB. This may overload the file system. To turn your new concatenated single-source file back into a multi-source file (if this seems cumbersome, you're right) use the task MULTI.

MULTI	
sources=' '	Pick up source name from header
aparm(1)=5	CL table time interval in minutes

Last update : 27/11/93

15 IMAGING

15.1 General

Following the initial calibration, your next task is a little more enjoyable; namely, making some images. With the ATCA you always get spectral information (unless you throw it away; see § 6) around your central observing frequency, allowing the possibility of multi-frequency synthesis (i.e., putting all the channels in the band in their correct places in the (u, v) plane and making one image) for continuum data. In addition, the temporal frequency agility of the ATCA enables observations with a number of widely spaced frequencies in the one broad band, so that again the (u, v) coverage can be improved.

New algorithms and tasks have been developed by Bob Sault to take advantage of these features (see the *MIRIAD* User's Guide and Bob's ATNF technical memorandum). However, this code resides in *MIRIAD* where you must work if you wish to make the most of your ATCA data in this regard. The algorithms in *AIPS* remain fairly basic; the biggest drawback in *AIPS* (but not *MIRIAD*) is that it cannot cope with a source brightness that changes substantially between the frequencies to be combined; i.e., a non-zero spectral index. Be aware of this, as it limits the usefulness of the multi-frequency synthesis procedure.

I will discuss a variety of different ways to image ATCA data. These will include multi-frequency synthesis and spectral-line imaging. No matter which of these you actually plan to do, you should read the first sub-section on how to image just one channel at one frequency, as it contains some technical discussion that is required background for all ensuing sub-sections.

The imaging tasks in *AIPS* are `MX`, `UVMAP`, and `HORUS`. `UVMAP` functions only on single-IF single-source files in XY sort order (XY order means that the data are put in order of increasing u and v). `MX` also only functions on single-source files, although it does IF selection and can take a TB sorted file for input. This is quite an advantage as the sorting procedure is very expensive on computer resources. Imaging TB sorted data directly does limit the size of the image, but there should be no problem for most applications. `HORUS` functions on both multi- and single-source files in either TB or XY order and has IF selection. Both `HORUS` and `MX` can do multi-frequency synthesis imaging including gridding multiple IFs together.

There is a vast body of literature discussing the digital 2-D Fourier transformation of visibility data into images, and I will not replicate much of it here. Excellent references are the NRAO workshops on imaging. The basic concept to grasp is that the Fourier Transform of the visibilities (called the dirty image) is the sky brightness (multiplied by the primary beam response of one telescope of the array) convolved by the point-spread function of the synthesis array (called the dirty beam). The better your (u, v) coverage, the more your dirty beam approaches a delta function which is what you would get with complete (u, v) coverage.

The first major step is referred to as gridding. This procedure interpolates the irregularly sampled (in u and v) visibilities onto a regular grid so that the data can be fast Fourier transformed. You could do a direct Fourier transform on the ungridded visibilities if you wished, but that would be computationally prohibitive for any but the smallest problems. Gridding is done by convolving the visibilities with a function that is just a few grid cells wide, and then resampling the convolved function at the centre of grid cells. The width of a grid cell (in u say) in wavelengths is $\Delta u = 1/(N\Delta x)$, where Δx is the pixel size in radians in the image, and N is the number of pixels.

The next step is the FFT, which produces a dirty image. However, because the visibilities were convolved by some function in the gridding procedure, the convolution theorem indicates that the effect on the image is to multiply it by the Fourier transform of the convolving function. Therefore, the 'grid correction' is made, which divides the dirty image by the Fourier transform of the convolving function. As well as the dirty image, the dirty beam is computed. This is achieved by replacing the gridded visibilities by a point source of unit amplitude at the phase centre and re-computing the grid corrected image. Thus, the dirty beam is just the Fourier transform of the sampling function of the (u, v) plane.

Note that the units of the dirty beam are not terribly useful. They are Janskys per dirty beam area. However, since the integral of the dirty beam (and dirty image) is zero, this does not mean much. The only real meaning is that an isolated point source of flux density S Jy will appear in the dirty image as a dirty beam shape with amplitude S. An extended source of total flux density S Jy will appear in the dirty image convolved with the dirty beam, but the integral won't in general be S Jy.

It is always a good idea, when you first image a field, to examine the entire primary beam FWHM (or more) on the sky. That is, you should find out what sources your data are responding to. If you have a high resolution observation, this may require you to initially lower the resolution (called "tapering"; see below) so that you can image the full primary beam in a modest sized (1024 by 1024 pixels) image. Note that the ATCA telescope primary beam (as distinct from the synthesised dirty beam) has quite high sidelobes, and you can find that the

array is responding to very distant bright sources.

Your dirty image generally requires some form of image reconstruction (deconvolution) to remove the effects of the dirty beam's sidelobes. This might be achieved with CLEAN or maximum entropy and will be discussed in § 16. Of course, if you have no signal in your image (e.g. a detection experiment that failed to detect anything) then there is no point deconvolving the noise.

15.2 Imaging one channel and one `freqid`

The 'one channel' in the title of this sub-section might refer to a channel 0 visibility file or a single channel from a multi-channel visibility file. This, the simplest of all cases, can be dealt with by the task **HORUS**, which can operate directly on the multi-source file. Alternatively, you could split off a calibrated single-source file with **SPLIT** (see § 12 above), and image it with either **HORUS**, **UVMAP**, or **MX**. If you have concatenated data from different observing runs, then you probably already have a single-source file with the calibration applied (see § 14).

15.2.1 *Sorting*

Any single-source file that you produce needs to be sorted into XY order for **UVMAP** and sometimes for **MX** for particularly big images. You will be informed if you have asked for too large an image. Use **UVSRT** for sorting the data. There is some discussion on **UVSRT** in § 5 that is worth reading.

UVSRT	
sort='xy'	Sort into xy order
outdisk=2	Select output disk
baddisk=3	Don't put scratch files on this disk

15.2.2 *Imaging*

Now you can image the data. **HORUS** and **UVMAP** both Fourier transform the visibilities and create a dirty image and a dirty beam. Alternatively, you could use **MX** which images and optionally deconvolves with CLEAN. **MX** can image up to 16 fields in the primary beam simultaneously (hence its name, because the MX missile makes [or used to] multiple warheads) whereas **UVMAP** and **HORUS** can only do one at a time. I recommend that you do not deconvolve (CLEAN or MEM) blindly. Until you become familiar with the source that you are imaging, *always* make a dirty image first and inspect it for emission. You can tell **MX** to make a dirty image without CLEANing, and then restart it for the CLEAN step later (see § 16). **MX** is very useful for discrete and widely separated emission, because you can use the multi-field capability to make small images centred on the sources of emission, rather than having to make one huge image. **MX** also has advantages in the CLEAN step (see § 16). Note that there is no **MX** like program in *MIRIAD*.

The inputs for **HORUS**, **UVMAP**, and **MX** (for just the imaging stage) contain many of the same adverbs, so I will ju show the **UVMAP** adverbs, and any additional ones for the other tasks that might cause confusion. Note of course, that if you use **HORUS** on a multi-source file, you should turn on the relevant calibration adverbs. You must apply the calibration just as in **SPLIT** (§ 12), so I will not discuss the calibration adverbs again here. For single-source files, the calibration adverbs should be turned off (as there are no calibration tables to apply anymore) For **MX**, I will defer discussion of the CLEAN step adverbs to the deconvolution section (except to explain how to turn the CLEAN off). Finally, note that **MX** also creates what is called a visibility work file. You will need this in the CLEAN step later on, so don't delete it. Now, on with the nitty gritty.

- Important adverbs in the imaging tasks are **imsize** and **cellsize**. The former is the size of the image in pixels and should be a power of 2. The latter is the size of each pixel in arcseconds. Both specify an x and a y dimension. They needn't be the same (e.g. if your beam is elongated significantly in the north-south direction). The product of **imsize** and **cellsize** should be large enough to encompass the emission. If you are using **UVMAP** or **HORUS**, this emission must fit into the inner quarter of the image for correct deconvolution. If you are using **MX** then it can fill virtually the entire image and still be succesfully deconvolved. Choose the **cellsize** by allowing about 3 pixels across the FWHM of the main lobe of the dirty synthesised beam. If you under-sample the beam, you will induce some unpleasant effects in your images, especially if you plan to deconvolve. If you do not plan to deconvolve, then you could probably (but only if necessary) set the pixel size so that there are 2 pixels across the FWHM of the synthesized beam without sacrificing too much.

Consider also the effect of making an image from too few visibilities. You are not doing anything very meaningful if you make an image with more pixels than you have visibilities to constrain the pixel values. This can easily be the case with ATCA snapshot data and large images.

Since $\Delta u = 1/N\Delta x$, the size of the gridded (u, v) plane is $1/\Delta x$ wavelengths. This must be big enough to encompass your longest spacings. If you find the imaging programs tell you that they have not used all of the visibilities (they may say "Note: using only 1 of 345430 visibilities" or the like), you may have set the cell size to too large a value. Essentially you have not matched your resolution to the resolution expected from the longest spacing. This means you would throw away some of your data.

- As mentioned earlier, it is *always* a good idea to make a low resolution image of the entire primary beam of new fields. This is done by weighting (tapering) the short spacing data more than the longer spacing data and imaging with a larger cell size. With the ATCA, unless you are imaging a visibility file with multiple configurations, you do not have a lot of spacings to play with, so you should do this rather carefully. If you taper too heavily, you will weight down too much data and there will be insufficient left to make a decent image. The tapering procedure multiplies the visibilities by a Gaussian, and you specify, via `uvtaper`, its 30% level in $k\lambda$ in the u and v directions. See the `HELP` file for more information.

- As well as applying a Gaussian taper, you also have control over how the visibilities are weighted through the `uvwtfn` adverb. You can select uniform or natural weighting. When gridding, N visibilities contribute to a 'summed' (really convolved and resampled) visibility in each grid cell. Uniform weighting means that the gridded visibility is normalized by N; it is weighted down by the local density of points. This means that each gridded cell contributes 'uniformly' to the Fourier transform. In natural weighting, the normalization by N is not done, so that each gridded visibility contributes its 'natural' weight to the Fourier transform. Uniform weighting gives slightly higher resolution and a better sidelobe response. Natural weighting gives better sensitivity and is generally preferred for detection experiments. The default is uniform weighting.

- `uvbox` controls the width of the box used in normalizing the visibilities for uniform weighting. The default of zero means that the normalization is determined from the number of ungridded points falling in each grid cell. If `uvbox=1` then the total number of ungridded points in a 3×3 box is found. If `uvbox=2` the sum is done in a 5×5 box. These bigger boxes do what is called super-uniform weighting. They are sometimes useful for sparsely sampled data. Note that if the box was as big as the longest spacing, then you would have natural weighting again, as all the visibilities would be normalized by the same number. Experiment with caution here.

- A somewhat capricious adverb is `zerosp`. This allows you to specify the total flux in the primary beam (should you know it from single-dish observations) and its weight. The total power is known as the 'zero spacing', because this is what the visibility function sampled at $u = 0$ and $v = 0$ corresponds to. It is the source of a great many problems in interferometry because we can't measure it. It causes sources that have spatial frequencies larger than that corresponding to the shortest spacing to sit in negative bowls in the image. Any deconvolution algorithm (CLEAN or MEM say) actually tries to *extrapolate* the visibility function to the zero spacing value. However, if you provide the correct value, the *interpolation* through the unsampled part of the visibility function (at spacings less than the shortest baseline) is helped considerably. If the visibility function is very steep for the innermost spacings, the presence of a correct zero spacing value can prove a great advantage. But there are limits. If the correct zero spacing is more than about 4–5 times the amplitude of the visibility on the shortest baseline, you will be asking too much of the zero spacing insertion.

  A more difficult problem is that you must also provide a weight for the zero spacing, and the correct weight has proved rather elusive in the past. One recommended recipe (Jacqueline van Gorkom) is that the weight should be the number of gridded cells in the unsampled region of the (u, v) plane interior to the shortest baseline in the data. Trial and error is usually the way to do it. The effect on the dirty map is simply to add an offset (as the Fourier transform of a delta function is a constant). This will almost be indiscernible. The benefit comes in the deconvolution stage, so the loop is rather tedious (see § 16) if you get it wrong a few times. For your first pass, leave `zerosp` out, but remember it if need be. It's potentially *very* useful.

- `UVMAP`, `HORUS`, and `MX` can all make a set of channel images for spectral-line work, although the channel selection adverbs are all rather inconsistently used and confusingly defined. However, since I am only discussing imaging of one channel here, there is some hope of getting it right. I have treated the channel adverbs, in particular, in separate adverb boxes.

- If you want to move the centre of the image from the observing phase centre, then set `shift` for the shift in x and y in arcseconds on the sky. Positive values shift the image centre to the North and East.

- The adverb **uvrange** allows you to exclude baselines outside of a certain uv distance ($\sqrt{u^2+v^2}$). For example, you may have a short ATCA configuration such as the 750 m configuration as well as the 6 km antenna. This leaves a very large gap between the spacings involving the 6 km antenna and all the rest. To image just the short spacings you could exclude antenna 6 with this adverb. For example, at 6 cm, 750 m corresponds to 12.5 Kλ, so setting `uvrange=0,20` would include the short spacings and exlcude all baselines involving antenna 6.
- Make sure you turn on the grid correction with `dogridcr`.
- An amusing diversion is to view the gridded tracks on the TV. If you would like to see this, set `dotv=1`.
- **xtype** and **ytype** specify the type of convolving function used in the gridding process. The recommended function is spheroidal, for which these adverbs should both be set at 5. This ensures maximum rejection of aliasing.

UVMAP	
inname,inclass	Select the XY sorted
inseq,indisk	visibility data
channel=0	For channel 0 data
channel=18	or select channel
nmaps=1	One image only
outname,outdisk	Output image name
outseq, outdisk	
stokes='I'	Image Stokes I
imsize=512,512	Image size (pixels) in x and y (power of 2)
cellsize=1,1	Cell size in arcseconds in x and y
shift=0	Don't shift image centre from phase centre
uvtaper=0	No taper
uvrange=0	Include all baselines
uvwtfn=' '	Uniform or
uvwtfn='na'	natural weighting
uvbox=0	Mimimum width for convolution function
dogridcr=1	Must make grid correction
dotv=$\pm$1	Optionally put gridded data image on TV
zerosp=0	Include total flux density of all sources in the field if you know it
xtype=5	Convolution function type for
ytype=5	gridding, use spheroidal function
xparm=0	Additional convolution
yparm=0	parameters
baddisk=0	$AIPS$ disks to not put scratch files on

The channel adverbs are slightly different for `HORUS`. In addition, you have the potential to average IFs together. For example, you may have set the two IFs to observe closely spaced frequencies in the same band and you would like to grid and image them together. Just set `optype='sum'` and select the desired IFs with `bif` and `eif`. If you only select one IF, then **optype** is ignored.

HORUS	
	Turn on the calibration adverbs for multi-source files
bchan=0	Image only one channel. 0
echan=bchan	for channel 0 data, or select
chinc=1	channel explicitly
bif=1	Select IFs to grid together
eif=2	and set the **optype**
optype='sum'	to grid IFs together

Because `MX` can make up to 16 images at once, some of the adverbs are used slightly differently. Note that `shift` is replaced by `rashift` and `decshift`, which allow you to specify a shift for each field, the number of which is given by `nfield`. You might want to use this option if it is not possible to fit all the emission present in the primary beam into one reasonably sized image (1024 pixels or less), and the emission is reasonably discrete. The channel adverbs are different again too. Once again you can grid IFs together, but this time, you just need to specify them with `bif` and `eif`, there is no equivalent to the **optype** adverb of `HORUS`.

MX	
in2name,in2class	Name of visibility work file; leave blank
in2seq,in2disk	when imaging only
bif=2	Select desired IF
eif=2	
bchan=0	Image only one channel. 0
echan=bchan	for channel 0 data, or select
chinc=1	channel explicitly
npoints=1	One image only
channel=0	No CLEAN restart
nfield=1	Just one field for beginners
fldsize=0	For CLEAN step only
rashift=0	Don't shift image centre
decshift=0	from phase centre
niter=0	Don't CLEAN yet

15.3 Imaging one channel and multiple `freqids`

Let us make the problem a little more complicated, and discuss data for which all the channels have been averaged to one (channel 0) and for which there is more than one time switched observing frequency. For example, in the 6 cm band, you may have data at 3 pairs of frequencies, spaced throughout the band, which you wish to combine to benefit from the enhanced (u, v) coverage (assuming the spectral gradients are sufficiently small).

1. Now, these data will reside in three `freqids` and two IFs (the simultaneous data go on the IF axis) within the one multi-source file. However, you can only access one `freqid` at a time. Thus, you *must* first split off the data associated with each `freqid` and IF into a separate *calibrated* single-source file. Apply `SPLIT` (See § 12) to the channel 0 file (or use the channel-averaging option in `SPLIT`). If you combined data from different observing runs, you will already have done this.

2. The next step is to join the files back up again ! Do this with the task `DBCON`. I have already discussed `DBCON` in § 14, and suggest you read that section. However, this time you must disable the frequency check.

DBCON	
reweight=0	Leave weights alone
dopos(1,1)=1	Check phase centres
dopos(2,1)=−1	Don't check for frequency agreement
doarray=−1	Separate input files in time by 5 days

3. Note that u and v are generally stored in the file in wavelengths at the reference frequency. This reference frequency will vary between the files that you care going to concatenate. However, `DBCON` does not recompute u and v for the new reference frequency of the output file. But all is not lost, for single channel data, the imaging tasks will not look at the reference frequency. They will simply take u and v as they are in the file so that the the image scales will come out correctly.

   Similarly, you must split the IFs separately because the concatenated file only has one FQ table, so that the second IF would not be correctly described if the 2 IFs were left together. As an example, if you had 3 `freqids` and 2 IFs and one channel, following `SPLIT` you should have 6 separate files, one for each `freqid` and IF combination. Then you join them all up again with `DBCON`.

4. Following all this messing about, you can now image the data with `UVMAP`, `MX`, or `HORUS` in the same way as described in § 15.2 above.

Again I point out that *MIRIAD* handles multi-frequency data in a much better fashion than *AIPS*.

15.4 Imaging multiple channels

The third category of data that you might wish to image is a band of channels. That is, you wish to make a multi-frequency synthesis image and grid together all the channels in the (u, v) plane making use of the variation of u and v across the band to improve the sampling of the (u, v) plane. The output is still one image.

Both MX and HORUS can do this kind of imaging. Therefore, once again, you can either image directly from the multi-source file with HORUS or you can use SPLIT (§ 12) (you may have already done this if you have combined data from different observing runs). If you want to make a very large image you may need to sort the data into XY order with UVSRT (§ 15.2), but generally the internal sorting that HORUS and MX can do allows big enough images.

1. Now, only small changes to the inputs of MX and HORUS as described in § 15.2 are required. For both MX and HORUS, bchan and echan specify the range of channels that will be selected, and chinc specifies the channel increment. For MX, you have independent control over the number of channels that are gridded together through the npoints adverb. Thus, you might set chinc=3 and npoints =2. MX would grid two adjacent channels in forming every output plane of the image, and the first channel in every pair gridded would increment by 3. Thus, for example, output images might be formed from channels 1 & 2, 4 & 5, 7 & 8 and so on. With HORUS, this fine control is absent. When you put optype='sum', all channels in the range bchan to echan are gridded together. I believe that chinc is ignored for this option, so that you always get one output image.

2. Again you can either select a single IF, or you could, if it was appropriate, grid multiple IFs together as well. For example, closely spaced IFs within the same band may usefully be imaged together. Use bif and eif to select the IFs.

   In the adverb box below, I give a simple example of gridding IFs 1 and 2 and all channels in the range 8–28 together to make one output image.

MX	
bif=1	Select desired IFs
eif=2	
bchan=8	First channel to grid
echan=28	Last channel to grid
chinc=1	Channel increment
npoints=21	Number of channels to grid together
channel=0	No CLEAN restart

HORUS	
bif=1	Select desired IFs
eif=2	
bchan=8	First channel to grid
echan=28	Last channel to grid
chinc=1	Not relevant
optype='sum'	Grid all selected channels and IFs

It would be interesting to compare the multi-frequency synthesis image with the channel 0 image, if you believe there is some (u, v) coverage benefit to be gained in the former.

15.5 Imaging multiple freqids and multiple channels

The final category of continuum data that you might wish to image is multiple bands of channels.

1. It will be necessary in this case to use SPLIT (§ 12) for each freqid of interest (or if you combined data from different observations you may already have done this). It may then be necessary to sort each visibility file into XY order with UVSRT as described in § 15.2 if you are making very large images (but try the internal sort of MX and HORUS first).

2. Now, we would like to concatenate the data with DBCON, but there is a problem. Each individual file will have a different reference frequency (use IMHEAD to see this). DBCON will create an output file with just one reference frequency, and it will be the reference frequency of the first freqid file used in the concatenation process. This is no good, because MX and HORUS will use this reference frequency for all the data when they try to work out the variation of u and v across the band so that the different channels can be gridded in the correct place.

   The solution is to image each file individually, and then sum the resultant images and beams. Since the Fourier transform is a linear procedure, there is no substantial difference in making the image in this way.

The only caveat to this is that this statement is really only true for natural weighting (`uvwtfn='na'` in the imaging tasks; see § 15.2). You should image the individual files with MX or HORUS just as described in § 15.4.

Again *MIRIAD* performs much better in the multi-frequency synthesis imaging area than *AIPS*.

3. Now average the images and beams with COMB. This image can be deconvolved with the averaged beam.

COMB	
doalign=1	Check images for alignment
blc=0	Combine all of images
trc=0	
opcode='sum'	Sum images
aparm=0	
aparm(1)=0.5	Divide input image 1 by 2
aparm(2)=0.5	Divide input image 2 by 2
bparm=0	

15.6 Imaging spectral-line data

Compared with all this complicated continuum multi-frequency synthesis, spectral-line imaging is rather easy. You just need to make one image per channel, or perhaps one image per group of channels. UVMAP is not very useful for spectral line imaging, so I will just discuss HORUS and MX. The end result is a three-dimensional image, the third dimension being frequency. This is referred to as a cube. In spectral-line imaging, it is important that you do not make images at a frequency resolution that is much too great compared to the width of the features you are interested in. Otherwise you run the very real risk of not detecting your line. Make sure the frequency resolution is such that there are about 3–4 points across the line at most. This may require you to use the channel binning option of MX. If you are making a detection experiment, the best sensitivity to the putative line is obtained by matching the channel width to the line width.

A preceding chapter, §13 discusses other spectral-line issues, including continuum subtraction which you should do first.

1. As usual, HORUS can be run on a multi-source file with the calibration applied directly (§ 12), or on a single-source file (any sort order) which has already had the calibration applied to it. The channel control is fairly rudimentary, and all you can do is give a channel start, end, and increment. As far as I am aware, the increment just skips channels, it doesn't average them together. However, you might like to apply some spectral smoothing with the `smooth` adverb. You can Hanning smooth by setting `smooth=1,0` and then throw away every other channel with `chinc=2`.

   All other parameters should be as described in § 15.2.

HORUS	
	Apply calibration adverbs
	to multi-source files
optype='line'	Spectral line imaging of Stokes I
smooth=1,0	Hanning smooth
bchan=50	Image channels 50
echan=450	to 450 at increments
chinc=2	of 2 say

2. For MX, you must first run SPLIT (§ 12) and possibly sort to XY order with UVSRT (§ 15.2) for very large images. MX provides more flexible channel selection than HORUS. I have described how the channel-selection adverbs work for MX in § 15.4.

   In the example below, pairs of channels are gridded together, starting at channel 50 and ending at channel 450. There is no `smooth` adverb in MX.

MX	
in2name,in2class	Name of visibility work file, leave blank
in2seq,in2disk	for first time, else fill in
bif=2	Select desired IF
eif=2	
bchan=50	Grid pairs of channels
echan=450	from 50 to 450
chinc=2	Skip every other channel in output
npoints=2	Grid 2 channels together
channel=0	No CLEAN restart
nfield=1	Just one field for beginners
fldsize=0	For CLEAN step only
rashift=0	Don't shift image centre
decshift=0	from phase centre
niter=0	Don't CLEAN yet

Last update : 27/11/93

16 IMAGE RECONSTRUCTION (DECONVOLUTION)

Because synthesis arrays sample the (u, v) plane at discrete locations, there is incomplete knowledge about the Fourier transform of the source intensity distribution. The measured (u, v) data can be thought of as the true distribution, $V(u, v)$, in the (u, v) plane multipled by some sampling function, $S(u, v)$. The convolution theorem states that the Fourier transform of the sampled distribution (the dirty map, I_D) is equal to the convolution of the Fourier transform of the true source (u, v) distribution (the true image, I) and the Fourier transform of the sampling function (the dirty beam, B_D):

$$I_D(x, y) = I(x, y) \otimes B_D(x, y) \rightleftharpoons V(u, v) \times S(u, v)$$

where $\otimes$ indicates convolution, and $\rightleftharpoons$ indicates the Fourier transform. Deconvolution algorithms attempt to account for the unsampled regions of the (u, v) plane. If it was fully sampled, there would be no sidelobes, since the sampling function would be a constant, and the Fourier transform of a constant is a delta function; a perfect beam. Thus, deconvolution tries to remove the side lobes of the dirty beam that are present in the image. It is important to realize that in doing so, the algorithm is guessing at what the visibilities are in the unsampled part of the (u, v) plane. The solution to the convolution equation is not unique, and the problem of image reconstruction is reduced to that of choosing a plausible image from the set of possible solutions.

You should be extremely cautious when deconvolving images formed from a small number of snapshots. In these cases, there will be large areas of the (u, v) plane that are unsampled, because of the poor instantaneous (u, v) coverage of the ATCA. If the source is complicated, the deconvolution algorithm may go badly wrong in its guess of what the source really looks like in the gaps. The best way to make a decent image of an object is to observe it, not allow a deconvolution algorithm to guess what it looks like.

There are two techniques used commonly in radio atronomy; CLEAN and maximum entropy (MEM). CLEAN is rarely used outside of radio astronomy, but MEM is more far reaching. For detailed discussion on the 'pros' and 'cons' of these algorithms, see the NRAO imaging work shops and references therein. Much blood has been spilt over their relative merits in the last decade or so.

It is probably fair to say that in general, CLEAN is easier to drive than MEM, although use of MEM can result in reduced processing times for large problems. However, if you need to use **MX** as your imaging tool, there is no choice; CLEAN is what you get. A dirty image made with **HORUS**, **UVMAP**, or **MX** can can be deconvolved with one of **HBCLN**, **APCLN**, **SDCLN** (CLEAN), or **VTESS** (MEM). **VTESS** can also be used for mosaicing, although I will not discuss this further.

16.1 Deconvolution with CLEAN

16.1.1 *The Högbom CLEAN*

The CLEAN (Högbom 1974) algorithm represents the image as a number of point sources in an otherwise empty field of view. A simple iterative procedure is used to find the locations and strengths of these point sources.

The Högbom CLEAN algorithm looks for the brightest (absolute sense) pixel (called a CLEAN component) in a specified region in the image (called the CLEAN window which must include all the real emission in the image) and subtracts a fraction (called the loop gain, and generally about 0.1 or less) of the dirty beam from the dirty image at the location of that brightest pixel. This subtracted image is called the residual image. The search and subtraction loop is repeated until the sidelobes in the image are reduced to below the noise level. It is easy to see how CLEAN works if you just think about a single point source. For extended sources, one thinks of the emission as a collection of point sources.

After CLEANing is finished, the model of the source that you have constructed consists of a list of CLEAN components at various locations in the image. Looking at the CLEAN component *image* is a sobering experience (you can't do this easily with *AIPS* though). To improve the qualitative appearance of this model, and to suppress what is essentially unmeasured high spatial frequency structure (i.e., structure smaller than the resolution), CLEAN generally finishes its work by convolving the CLEAN component image by a Gaussian. This convolved image is then added to the residual image. The Gaussian is chosen to match the main lobe of the dirty beam, and is called the CLEAN beam. The *AIPS* task **HBCLN** performs a Högbom CLEAN.

I mentioned that the units of the dirty image are not very useful (§ 15). However, convolved CLEAN components do have meaningful units of Jy per CLEAN beam area, which can be converted to Jy per unit area, because the equivalent area of the CLEAN beam can be calculated.

16.1.2 *The Clark CLEAN*

The *AIPS* task APCLN performs what is known as a Clark CLEAN (Clark 1980). This is a variant on the original Högbom algorithm, which spends a lot of time shifting and scaling the dirty beam. This is essentially a convolution, and may in some circumstances be more efficiently performed with an FFT. For very small images though, the Högbom CLEAN may still be faster.

The Clark CLEAN plays a few tricks which you should know about. It has what are termed major and minor cycles. In the minor cycle, clean components are searched for, and subtracted from the image, in a fashion very similar to that of the Högbom CLEAN. The difference is that only the central portion of the dirty beam is used for the subtraction. The rationale is that for dirty beams that have a fairly good sidelobe pattern, this will be good enough to find the CLEAN components. At some designated time, this minor cycle is terminated and a major cycle computed. This means that the list of CLEAN components from the current cycle are 'FFTd', and subtracted from the FFT of the residual image that resulted from the *previous* major cycle. In this way, errors that might have accumulated in the subtraction phase of the minor cycle with only a part of the beam are largely set to rights. Because of the FFT, a Högbom CLEAN can be more efficient than a Clark CLEAN on small images.

This use of a small beam patch in the Clark CLEAN is a potential danger if the beam is poor. Images made from ATCA snapshot data, even if there are a few cuts, often have a beam in which the first side lobe response is quite strong and outside of the beam patch that the Clark CLEAN typically uses. I will discuss this more in the discussion below of the *AIPS* tasks that do a Clark CLEAN.

16.1.3 *The SDI CLEAN*

CLEAN sometimes runs into an instability when working on extended sources of fairly uniform surface brightness. It produces obviously wrong 'CLEAN stripes' in your image. The Fourier transform of these stripes tends to be found in unsampled parts of the (u, v) plane. This is a complicated way of saying that the visibility data do not constrain CLEAN's interpolation between the measured visibilities sufficiently well to prevent it putting in badly wrong values.

One change to the Clark CLEAN that helps suppress CLEAN stripes is the SDI CLEAN (Steer, Dewdney, and Ito 1984). In this variant, implemented in SDCLN, any point in the residual image greater than some factor times the maximum residual point is taken as a CLEAN component. The factor is less than one. Thus, when the residual image has become very smooth, this avoids introducing ripples into it (which occur when subtracting the beam from just one location) which lead to the stripes. This algorithm is quite successful, although the implementation in SDCLN is slow.

Note that the Högbom, Clark, and SDI versions of CLEAN only allow you to CLEAN the inner quarter of your image properly because the image and beam are the same dimensions. For example, consider images that are 256 pixels in size (one dimension will do), with a point source located at pixel 192. A beam image centred on that point source will extend only to pixel 64, so that the full image cannot be CLEANed.

16.1.4 *The Cotton-Schwab CLEAN*

Now, there is a further variant on the Clark CLEAN that is used by MX (Schwab 1984). During the major cycle, the Fourier transformed CLEAN components are subtracted from the *ungridded* residual (u, v) data, instead of the residual gridded (u, v) data (i.e., the FFT of the residual image). This means that the subtraction is more accurate, and allows CLEANing of virtually the whole image, instead of just the inner quarter.

16.1.5 *Prussian helmets*

A second mention of CLEAN stripes. (Cornwell 1983) suggested that this instability is suppressed by the addition of a delta function to the dirty beam. For obvious reasons, this is popularly termed the 'Prussian helmet' CLEAN. The spike is usually set at about 0.3 (see EXPLAIN APCLN). Together with this, one usually makes the loop gain smaller, to perhaps 0.05 or even 0.01. However, recent wisdom suggests that the Prussian helmet is probably a red herring. I suggest that rather than mixing with teutonic hats, you use SDCLN, which takes a more reasonable approach to CLEAN stripes.

The EXPLAIN files for APCLN, SDCLN, and MX contain lots of information about CLEAN, should you like to read some more.

16.1.6 *Computation*

1. Let us proceed with the actual use of CLEAN and start with APCLN. The inputs for HBCLN, SDCLN, and the CLEAN stage of MX are very similar to those of APCLN, and I won't show them in any detail. You need to provide APCLN with the dirty image and beam, and it will provide you with a CLEANed image, or if you so desire, a residual image. This residual is often useful to examine, when trying to work out if you need to CLEAN some more; if there is source left in the residual image, you probably aren't finished yet.

   - An important parameter in CLEANing is the CLEAN window. This is the region of the image in which CLEAN looks for CLEAN components. It is very helpful to CLEAN if this window snugly encompasses the real emission in the image. At first, you may not know where this is. However, after some quick and dirty (sic) CLEANing, you could redo it properly with a better idea of the CLEAN window. Essentially, you are giving CLEAN *a priori* information about the source, which better constrains the deconvolution problem. You can specify up to 10 CLEAN windows with the adverbs nbox and box. Make sure they don't overlap and put them tightly around the real emission. You can set the CLEAN window(s) interactively with the TV and the command TVBOX.
   - A good number for the loop gain is 0.1. If CLEAN stripes occur, then try lowering the loop gain by a factor of about 10, or give SDCLN a go.
   - You can tell CLEAN when to stop (some say you should never start) in two ways. You can tell it to quit once the brightest pixel in the residual image reaches some multiple of the r.m.s. noise level (3–σ or so). Specify this level with the adverb flux. Otherwise, CLEANing will proceed until niter CLEAN iterations have been performed. For small and simple sources, a few hundred iterations are usually sufficient. For complicated and large sources, you can CLEAN forever.
   - There is a knob that protrudes from all Clark CLEAN algorithms, usually with the inspiring name of factor. This controls how often major cycles are performed. The more major cycles, the more thorough is the CLEAN, and the longer it takes. For point sources, this should be about 1. For extended sources, something in the range -1 to -0.5 is good. Use 0 for in-between sources.
   - The parameter minpatch is the minimum half-width of the beam patch used in the minor cycle; the default is the maximum value of 127 (this number is for historic reasons). If your beam has a good sidelobe response, you can set minpatch to something smaller; 51 is the usual number. It is a good idea to examine your beam first before using the Clark CLEAN, so that you can assess whether the small beam-patch algorithm will be likely to cause you trouble or not. If it does, use HBCLN.
   - If bmaj and bmin are zero, the main lobe of the dirty beam will be fitted by a Gaussian. This is the CLEAN (or restoring) beam used to convolve the CLEAN component image before it is added to the residual image. The fitting algorithm is simple and often fails for messy or elongated beams. If so, make a contour plot (with CNTR) and measure the FWHM of the beam, and fill in bmaj, bmin, and bpa explicitly. This is a reasonable way to make your beam circular if you so desire. Don't make the CLEAN beam much smaller than the real FWHM of the main lobe of the dirty beam. Super-resolving is a nasty business, and best left to MEM. Similarly, don't make the restoring beam much bigger than the natural resolution of the image, because the residuals are not convolved by the restoring beam.

     If you want the final image to be the residual image rather than the restored image, make bmaj negative.

     If you really want to see exactly where the CLEAN components come from in the image, make bmaj and bmin very small (10^{-5} say). The restored image will then be the residuals plus delta functions where the CLEAN components were found. If you are truly a masochist, and want to see an image of the CLEAN components (the true CLEAN model), then subtract the residual image from this image with COMB.

     For beams with a high sidelobe response, the simple algorithm that fits the beam is prone to failing. Sometimes it knows it fails, and tells you, other time the program crashes. For poor beams, you should measure the FWHM of the beam yourself (make a contour plot) and enter the beam parameters into bmaj, bmin and bpa. When the beam fitting algorithm fails, it gives an arbitrary value for the FWHM. The usual signature is that the major and minor axis FWHM are identical.
   - Note that APCLN can only CLEAN one plane of a cube at a time, but that the output image will have the same dimensions as the input cube, no matter which plane you happen to be processing. You select the desired plane with blc(3). You might write an *AIPS* procedure to loop over all the planes. Make sure that you specify the output file completely in the INPUTS so that APCLN does not generate as many output files as there are planes when CLEANing planes after the first one.

- The CLEAN components are stored in the CC extension file. For each run of **APCLN** on the same cube, you should generate a new CC file.
- If you CLEAN an image, and find that you need to CLEAN some more, **APCLN** can be restarted from where you left off. Do this by specifying the output file completely (use **GETONAME**), and setting **biter** at the number of compnents already CLEANed.

APCLN	
inname,inclass inseq,indisk	Dirty image
in2name,in2class in2seq,in2disk	Dirty beam
outname,outclass outseq,outdisk	Fully specify when restarting CLEAN
blc(3)=10	Clean plane 10, say, of input image
invers=0	Make CC file next highest version
gain =0.1	Loop gain
phat=0	No prussian helmet
flux=0	Terminate CLEAN at this residual level or
niter=500	specify total number of CLEAN components
biter=0	Number already CLEANed if restarting
bmaj=0	CLEAN beam. If 0, program tries
bmin=0	to fit main lobe of dirty beam to
bpa=0	work it out. If **bmaj** negative the output image is the residual image
nbox=0	CLEAN window defaults
box=0	to inner quarter of image
factor=0	Speedup factor; −1 for extended sources, +1 for point sources
minpatch=127	Minimum beam size for minor cycles
dotv=1	Display residual images on TV

The total CLEANed flux density (i.e., the cumulative sum of the CLEAN components) should eventually settle down to a roughly constant number (you can look at this with the task **PRTCC**). This indicates that you are just picking up noise, and that there are no side lobes left to remove. If the total CLEANed flux starts to decrease again, this usually indicates that you have been a bit heavy handed, and CLEANed too much. You might also look at the result and see if you can see any side-lobes left over.

Some discussion of the zero spacing was given in the imaging section (§ 15). It is the deconvolution step that really tells you whether you managed to get the combination of the zero spacing and its weight correct. If you did, the total CLEANed flux should be equal to the zero spacing. If you got it wrong, this will not be the case, and one often sees the CLEAN window region badly offset from the rest of the image. You will know when you need to try again.

2. Note that for **MX**, you can specify a restart CLEAN component number for each field through **bcomp** instead of **biter**. When restarting the **MX** CLEAN, you must also fill in the correct work file (use **GET2NAME** to get it). This contains the current residual visibility data base (i.e., the original ungridded visibility data with the Fourier transform of the CLEAN components subtracted). Also in **MX**, you have the choice of a DFT or gridded FFT when Fourier transforming the CLEAN components. This is controlled by the adverb **cmethod**. The DFT is more accurate but very slow for large CC lists. Leave **cmethod** blank, and **MX** will make an informed guess on which is the better choice. Note that you cannot restart **HBCLN**, you always have to CLEAN from the beginning.

3. If you use **SDCLN**, there are three additional controls which you will probably need to experiment with. See the **EXPLAIN** file for details on these. First, all pixels brighter than **cutoff** times the current residual image peak are considered CLEAN components. Second, **stfactor** is the loop gain in the SDI phase of the CLEAN (**SDCLN** switches from a normal Clark CLEAN to an SDI CLEAN). Third, a normal Clark CLEAN is used until **ncount** pixels are included in an SDI iteration. That is, the residual image must reach some degree of flatness before the SDI CLEAN cuts in. See the **EXPLAIN** file.

16.2 Deconvolution with maximum entropy algorithms

16.2.1 *Theory*

This discussion was lifted from Tim Cornwell's article in the NRAO imaging workshop (1988).

CLEAN approaches the deconvolution problem by using a ***procedure*** which selects a plausible image from the set of feasible images. This makes a noise analysis of CLEAN very difficult. The Maxiumum Entropy Method (MEM) is not procedural. The image selected is that which fits the data, to within the noise level, and also has maximum entropy. This has *nothing* to do with physical entropy, it's just something that when maximized, produces a positive image with a compressed range of pixel values. The latter aspect forces the MEM image to be smooth, and the positivity forces super-resolution on bright, isolated objects. One general-purpose definition of entropy is

$$H = -\sum_k I_k log\left(\frac{I_k}{M_k}\right)$$

where I_k is the brightness of the kth pixel, and M_k is some default image. An example might be a low-resolution image of the object. This allows *a priori* information to be incorporated into the problem.

The requirement that each visibility be fitted exactly by the model usually invalidates the positivity constraint. Therefore, data are incorporated with the constraint that the fit, χ^2, of the predicted visibility to that observed, be close to the expected value:

$$\chi^2 = \sum_r \frac{|V(u_r,v_r) - \hat{V}(u_r,v_r)|^2}{\sigma^2_{V(u_r,v_r)}}$$

Maximizing H subject to the constraint that χ^2 be equal to its expected value leads to an image which fits the long spacings too well, and the zero and short spacings poorly.

16.2.2 *Computation*

- As with CLEAN, you give VTESS a dirty image (**inname** etc.) and a dirty beam (**in2name** etc.). You also have the option to give it a default image (**in3name** etc.) which is an initial model of the source that the algorithm begins with. This might be the result from a previous run of VTESS so that you can deconvolve some more, or it might be, say, a low-resolution image of the source.
- VTESS produces two output images. One is the MEM image (**outname** etc.), which has units of Jy per pixel. This image has a resolution that is signal-to-noise ratio dependent. You can also produce a convolved image (**out2name** etc.) by specifying convolving Gaussian through **bmaj**, **bmin**, and **bpa**. If the beam is left at zero, VTESS will try to fit the dirty beam's main lobe and use the resulting elliptical Gaussian for the convolution. This is the same procedure as CLEAN uses for its final convolved image.
- VTESS can actually do a very sophisticated mosaicing deconvolution as well as a simple single image deconvolution. This is what **nmaps** is for. Leave this at one for the moment, as I am not going to discuss mosaicing.
- **niter** specifies the number of iterations to perform. Start off with 10 or so.
- A crucial parameter is **noise**. This determines how well the algorithm will try to fit the model to the data. It should be comparable to the r.m.s. value of a blank region of a CLEAN image. If **noise** is too large, the output image will be too smooth. If it's too small, convergence will be prevented. A useful trick is to underestimate **noise** and then stop after a few iterations, and reset **noise** to the level achieved up to that point. Do not leave **noise** at zero.
- **flux** specifies how the zero-spacing flux is to be estimated. There are three modes of use. First, you specify a known value which must be fitted to within 5%. Second, if you have no idea what the zero-spacing flux is then leave **flux** at zero. VTESS will attempt to estimate it. Third, if you have a rough idea (within a factor of 2 say) then set **flux** to the negative of your guess. For example, **flux**= −2. VTESS will then do a reasonable job of estimating the true value.

VTESS	
inname,inclass	Dirty image
inseq,indisk	
in2name,in2class	Dirty beam
in2seq,in2disk	
in3name,in3class	Default image, leave
in3seq,in3disk	blank if nothing better
outname,outclass	MEM image. Units
outseq,outdisk	are Jy/pixel
out2name,out2class	Convolved image. Units
out2seq,out2disk	are Jy/beam
nmaps=1	No mosaicing
niter=10	Number of iterations
noise	Specify noise level carefully
flux=5	Specify zero spacing to 5% or
flux=0	let program work it out or
flux=−5	Specify minus zeros apcing to 50%
bmaj=0	Convolving beam If 0,
bmin=0	program fits main lobe of
bpa=0	dirty beam to work it out.

16.3 Correcting for the Primary Beam

The image you have carefully nurtured through a variety of complex procedures requires a finishing touch. Because each telescope in the synthesis array has its own beam response (the so called 'primary' as opposed to 'synthesised' beam), the image you have produced is really the sky (hopefully) multiplied by the primary beam. To correct for this, so that flux densities that you measure are not biased low, you must divide the image by the primary beam. In doing this, we assume that the primary beam is the same for each telescope, and that it is circularly symmetric. I refer you to the ATNF memo by Mark Wieringa (1992) if you wish to examine how good these assumptions are and what the true shapes of the ATCA primary beams are.

Note that you should not make the primary beam correction on a dirty image and then try to deconvolve it. You should also be aware that the noise, as well as any source will be amplified by the primary beam correction.

The *AIPS* task `PBCOR` has knowledge of the ATCA primary beam as well as the VLA primary beam. It works out which telescope to use by looking in the header, in which the TELESCOP keyword should be either 'ATCA' or 'VLA'. Use `PUTHEAD` if it doesn't say what it should (use `IMHEAD` to view the header). `PBCOR` also uses the frequency axis of the image to work out what the parameters of the primary beam correction should be, so make sure the header correctly reflects the frequency as well.

PBCOR	
inname,inclass	Image to correct
inseq,indisk	
outname,outclass	Corrected image
outseq,outdisk	
blc = 0	Do entire
trc = 0	image
dparm = 0	Defaults OK
gpos = 0	Defaults OK

Last update : 27/11/93

17 SELF CALIBRATION

17.1 General

Because the basic interpolated calibration procedure does not perfectly determine the antenna gains at each time stamp, the quality of the resultant image suffers. The technique of self calibration is often used to make additional corrections to the gains as a function of time. It is very similar to the basic calibration in which the model of the calibrator was a point source at the phase centre. In self calibration, a model of the source that you are interested in imaging, is used to refine the antenna gains. Like basic calibration, self calibration can only correct antenna based errors.

If the source is a point source, then, again, a point-source model could be used, although the restriction that it be located at the phase centre is lifted. The model for a more complicated source is more sophisticated. Generally, it is a set of CLEAN components. Recall that this is a list of delta functions at specified locations in the image. Alternatively, you might use the model provided by some other deconvolution technique such as maximum entropy. A useful trick is used in self calibration to reduce the problem to one very similar to the basic calibration, as already discussed in § 3. The model is first Fourier transformed to the visibility domain, and then the data are divided by the Fourier transform of the model. This reduces the data to a pseudo point source, and the gains are easily determined from the equations given in § 3.

The self-calibration procedure is applied iteratively, each time with a better model, until finally the sequence converges and no more improvement in the image quality can be made.

Self calibration is not a technique that should be applied blindly. This is especially true for ATCA data, because the problem is only slightly overdetermined, compared with the VLA. This is because the ATCA has, at most, 15 baselines to determine the gains for 6 antennas, whereas the VLA has, at most, 351 baselines to determine the gains for 27 antennas. Because we can set the phase of one antenna to zero (see § 3.1), the problem reduces to finding 11 real numbers from 30 real equations. With the 5-antenna compact array, we must find 9 real numbers from 20 real equations. This problem is exacerbated when one antenna is absent from the data for a period of time, or you have flagged it out because of poor data quality.

There are other problems to do with the east-west nature of the ATCA and I refer you to the document by Bob Sault (see references) which discusses some of these. One important point to keep in mind is that self calibration with the ATCA depends crucially on the initial model that you start with — much more so than with the VLA where you can start with quite a poor model but arrive at the correct result after just a few iterations.

Self calibration, like basic calibration, requires that the signal-to-noise ratio on each baseline be of the order of at least 5 or so. For weak sources, this may require a long solution interval, within which the gain offsets for each antenna are assumed to be constant. If, in reality, the gains are changing on a time scale significantly shorter than your solution interval, and these changes are degrading your image quality, then you will be unable to improve the image quality. The time scale on which you will often find it necessary to correct the gains is approximately one minute. Thus, weak sources often cannot be self calibrated.

Note that it is not the receivers that cause the gains to be unsteady with time, but the atmosphere. In a similar way to the degradation of an optical image by the atmosphere, a radio image is defocused by the phase (and amplitude) noise that atmospheric cells induce into the wavefront. Self calibration can be thought of as an off-line mimicry of adaptive optics in optical astronomy.

17.2 Computation

This section is only relevant for self calibration of images that have *not* been made with multi-frequency synthesis techniques. That is, you have just one channel (usually channel 0) that you are imaging and self calibrating, not a band of channels. For self calibration of multi-frequency synthesis images, you should use *MIRIAD*.

The task you will need for self-calibration is `CALIB`. Once again, I have provided some procedures to help you do self-calibration with a reduced set of **CALIB**'s inputs. There is one procedure for multi-source files called `ATMCAL`, and one for single-source files called `ATSCAL`. If these are insufficient for your purposes, you know what to do.

When you self-calibrate with a multi-source file, all you do is produce a new CL table for application to the data; first you run `CALIB` to generate the solutions, and then you interpolate them to a CL table with `CLCAL` in the usual way. This means that the fundamental time scale on which you can alter the gains is the CL table entry interval. `ATLOD` builds the CL table with 3-minute intervals. You may have something different if you have rebuilt it with `INDXR` (it defaults to 5-minute intervals).

On the other hand, with single-source files, each application of CALIB produces a new and corrected data base. This has the advantage that, because there are no CL tables with single-source files, the fundamental time scale on which you can correct the gains is the integration time of the data (usually 10 seconds). The disadvantage is that you must duplicate the data with each pass. If you are working on a large data set, this may cause difficulty according to the available disk space.

1. Let us continue now with a discussion of some of the adverbs for multi-source self-calibration with ATMCAL. Many of the adverbs have already been discussed in § 8, so I will not cover them all.

   - Select the source to self-calibrate with calsour and calcode.
   - Select the desired frequency with freqid.
   - Calibrate the data by setting docalib=1 and by selecting the correct CL table with gainuse. This will probably be version 2 if this is the first self-calibration pass following basic calibration. If this is a later pass, then gainuse will be a higher version number.
   - Select the model, a list of CLEAN components, by filling in the CLEANed image name with in2name, in2class, in2seq and in2disk (use GET2NAME). The version number of the CLEAN component file is selected with invers (generally there is only one CC table, so leave this at 0).
   - ncomp selects the number of CLEAN components in the CC table to use as the model, starting at one; this is a very important parameter. Your goal is to provide CALIB with the best possible model of the source, so you should not include CLEAN components that are obviously wrong. One obvious example of this is a negative component, as the source must be positive. A common technique is to use all the components up to the first negative. However, if you have just one bad component in a much more extensive list of positive components, then it may prove advantageous to include more components than this simple approach suggests.

     This makes it clear why good windowing is so important during the CLEAN step; it helps exclude the possibility of including wrong components in the CLEAN model.
   - uvrange is also a fairly important parameter. Your model must match the data. Thus, as is often the case, if your model only represents a source with high spatial frequencies and your data contain a more extensive range of spatial frequencies, you *must* exclude the data that are not represented in the model. Two numbers are required, a minimum and maximum uv distance, in $k\lambda$. See also the task KALIB discussed at the end of this section.

     One good way to set this range is to make a plot of visibility amplitude versus uv distance with UVPLT. Find the total CLEANed flux density in your model by listing the CLEAN component table, and look up the corresponding uv distance on the plot. Set the uvrange to exclude the baselines interior to this value.

     With VLA data, this is a fairly safe recipe, because there will be many baselines in the selected uv range. However, with ATCA data, you must be careful not to exclude too many baselines, such that a good solution cannot be found. You must compromise between matching the data to the model and and having as high a level of over-determination in the problem as possible.
   - It may be that your source is a point source, or that by suitably restricting the uvrange, can be viewed as a point source (make sure you still include enough baselines to get a decent solution). In this case, rather than using a list of CLEAN components, you can specify a point-source model with the smodel array. smodel(1) is the flux density of the point source, smodel(2) is the x offset of the point source from the phase centre in arcseconds, and smodel(3) is the y offset. Note that if you leave the offsets at zero, the self-calibration procedure has the freedom to phase-shift the data so that the source appears at the phase centre (this can be a useful way to move a point source to the centre of a grid cell). Absolute position information is lost with self-calibration.

     smodel can also be used for elliptical Gaussian models. See HELP smodel for more information.
   - solint controls the solution integration time (in minutes) over which the gain changes are assumed to be constant. If you have sufficient signal-to-noise ratios, integration times of the order of one minute are a good starting place. Depending on the phase stability of the data, this number may need to be shorter or longer. You will probably need to experiment, but remember, for weak sources you should make sure you set this long enough to ensure a signal-to-noise ratio of at least a few on all baselines. However, this may preclude any useful modification of the gains, if it is too long.
   - The conventional wisdom at the VLA has always been that you should first do a couple or a few phase-only self-calibration passes (solmode='p') and then, once the phases are in order, attempt an

amplitude and phase self-calibration (`solmode='a&p'`). In my experience with ATCA data, it is sometimes better to try and do phase and amplitude straight away. I believe the reason is, that once errors in the model get frozen in, they are particularly hard to remove with ATCA data owing to the small number of baselines. Thus, you are better off trying to remove both the amplitude and phase errors straight away. As always though, experiment !!

- If the source you are trying to self-calibrate is polarized, but you have loaded the data as polarizations rather than converting to Stokes parameters, then recall that $XX = I + Q$ and $YY = I - Q$, so that you must average XX (called RR) and YY (called LL) before trying to work out a self-calibration solution. Do this by setting `doavg=1`. If the source is unpolarized, it may still be advantageous to do this, as it will boost your signal-to-noise ratio slightly. If you have converted to Stokes parameters, then set `doavg=-1`.

ATMCAL	
inname,inclass,inseq,indisk	Select data base
sources='0412 – 23'	Select source
qual=−1	Select all qualifiers or specify
freqid=2	One run per freqid
docalib=1	Calibrate data with
gainuse=2	the correct CL table
flagver=0	Select flagging table
in2name,in2class,in2seq,in2disk	Select CLEAN image for model
invers=0	Select CLEAN component file version
ncomp=250	Select number of CLEAN components to use for the model
smodel=1.3,−34.2,54.3	Use point source model rather than CLEAN components
uvrange=0	Specify *uv* range to match model
wtuv=0	Zero weight for points outside of *uv* range
refant=3	Specify the reference antenna
solint=1	Solution time in minutes.
soltyp=' '	Least squares algorithm. If too many failed solutions, try 'L1'
solmode='a&p'	Solve for amplitude and phase
doavg=1	Average XX and YY together

2. After the new solutions have been generated, examine them, as always, with `SNPLT`. Once you are happy with the results, apply them to the current CL table and write a new CL table; remember, CL tables are cumulative.

ATCLCAL	
inname,inclass,inseq,indisk	Select input data base
sources='0412 – 23',' '	Select source
calsour='0412 – 23'	
calcode=' '	Leave blank
freqid=1	One freqid per run
timerang=0	Select time range
interpol='2pt'	Select two point interpolation
gainver=2	Apply solutions to current CL table
gainuse=3	and update to next CL table
refant=3	Select ref. antenna

3. If you are using single-source files, then use the procedure `ATSCAL` which has inputs very similar to `ATMCAL`. I refer you to the discussion above for details. The main differences are the inclusion of the output corrected data base adverbs `outname`, `outclass`, `outseq`, and `outdisk`, and the absence of `flagver`, `gainuse` and the need to run `CLCAL`. Note that an SN table is still generated so that you can inspect the solutions. It is attached to the *input* data base.

ATSCAL	
inname,inclass,inseq,indisk	Select data base
sources='0412 – 23'	Select source
qual=−1	Select all qualifiers or specify
freqid=2	One run per freqid
docalib=1	Calibrate data with
gainuse=2	the correct CL table
flagver=0	Select flagging table
in2name,in2class,in2seq,in2disk	Select CLEAN image for model
invers=0	Select CLEAN component file version
ncomp=250	Select number of CLEAN components to use for the model
smodel=1.3,−34.2,54.3	Use point source model rather than CLEAN components
outname,outclass,outseq,outdisk	Specify output corrected data base
uvrange=0	Specify *uv* range to match model
wtuv=0	Zero weight for points outside of *uv* range
refant=3	Specify the reference antenna
solint=1	Solution time in minutes.
soltyp=' '	Least squares algorithm. If too many failed solutions, try 'L1'
solmode='a&p'	Solve for amplitude and phase
doavg=1	Average XX and YY together

4. You might also consider the task **KALIB**. This is a clone of **CALIB**, but it allows you to specify a **uvrange** (via the different adverb **uvlimit** for each subarray in the file (e.g. if you have concatenated configurations together with **DBCON** you will have several subarrays). With ATCA data, it is often crucial to finely tune the **uvrange** (e.g. to exclude just one shortest spacing in each configuration) and the **uvrange** adverb applies to all configurations in the file so that you end up excluding or including spacings undesirably. See the **EXPLAIN** file for more details.

5. It may occur that you wish to apply an SN table to a single-source file and write out the corrected visibilities (we apply CL tables to multi-source files and SN tables to single-source files). For example, you may be working on spectral-line data, and there is a strong maser at one channel. You can use this maser, which is easily self-calibrated, to generate solutions that you would then like to apply to say, a continuum data base. You simply copy the SN table to the desired single-source file, and then apply it with **SPLIT** (although in this case there is no 'splitting', just application of the SN table) by setting **docalib=1**.

Last update : 27/11/93

APPENDICES

A AVAILABLE TASKS in *AIPS*

A.1 Tasks to Create uv Data Files

1. ATLOD : Reads RPFITS files from tape or disk.
2. FILLM : Read a VLA archive tape. (post Jan88).
3. FILLR : Read a VLA archive tape. (pre Jan88).
4. MULTI : Converts a single-source file to multi-source format.
5. UVLOD : Copies Export or FITS tape UV data to disk.

A.2 General Calibration Tasks

- ATBPASS : BPASS with reduced inputs.
- ATCALIB : CALIB with reduced inputs.
- ATCLCAL : CLCAL with reduced inputs.
- ATSCAL : Self-calibration of single-source files with CALIB.
- ATMCAL : Self-calibration of multi-source files with CALIB.
- BLCAL : Determines baseline-based calibration.
- BPASS : Determine gains with frequency.
- CALIB : Central routine to determine antenna gains.
- CALCOP : Copy calibration tables from input to output.
- CLCAL : Apply solution tables to calibration tables.
- CLCOR : Apply various corrections to a calibration table.
- CLDEST : Procedure to delete the immediate argument CL table.
- CLSMO : Smooth CL tables.
- CLZAP : Delete all CL tables but the first.
- CLSNZAP : Delete all CL tables but the first and all SN tables.
- FRING : Fringe fit data
- GETJY : Find point source fluxes, fix SU and SN tables.
- INDXR : (Re)index a multi-source uv data file.
- SDCAL : Apply single dish calibration
- SNCOR : Task which applies various corrections to SN tables.
- SNSMO : Smooths SN tables.
- SNZAP : Delete all SN tables.
- SETJY : Enter calibration information into a source table.
- TABED : Edit an AIPS table.
- TACOP : Copy an AIPS table.
- TAFLG : Flag table entries.
- TAMRG : Merge an AIPS table.
- VSCAL : Perform self-calibration for VLBI.

A.3 Editing/data Examination Tasks

- CORER : Calculate correlator stats and flag bad ones
- CLIP : Flag uv data based on clip limits (single-source only).
- FFT : Task to FFT a real or complex image.
- IBLED : Interactive baseline editor on a TV device.
- IMHEAD : Examine the header of a file.
- LISTR : List data with optional calibration applied.
- POSSM : Averages and plots spectra from uv data with optional calibration.
- PRTAB : Print the contents of a table.
- PRTAN : Print the contents of an antenna table.
- PRTUV : Print selected uv data.
- QUACK : Flag bad data at beginning of scans.
- SHOUV : Displays uv data in various ways.
- SNPLT : Plots SN or CL table entries.
- SPFLG : Interactive, TV based data editor for spectral line data.
- SPLIT : Apply calibration and editing and write single-source file(s).
- TVFLG : Interactive, TV based data editor for continuum data.
- UVFLG : Flag selected uv data.
- UVFND : Find specified (bad) uv data points.
- UVHGM : Task to produce histograms summarizing uv data.
- UVIMG : Grids uv data into an image.
- UVPLT : Plot specified uv data.
- UVPRM : Plot specified uv data.
- VBPLT : Plots data on selected baselines with optional model.

A.4 Imaging and Deconvolution Tasks

- APCLN : Clark CLEAN of dirty image.
- BSMAP : Bi-Spectrum imaging of simple, weak objects.
- HBCLN : Högbom CLEAN of dirty image.
- HORUS : Fourier transform visibilties to dirty image directly from multi-source file
- IMERG : Merges high and low res. images.
- LTESS : Make linear combination mosaics.
- MX : Fourier transform visibilties to dirty image and CLEAN
- RSTOR : Restores a CC file to an image with a gaussian beam.
- SDCLN : SDI CLEAN of dirty image
- UVIMG : Grid UV data into an "image".
- UVMAP : Fourier transform visibilties to dirty image
- UTESS : Maximum Entropy deconvolution of dirty image that does not enforce positivity. Good for Stokes Q, U, and V.
- VTESS : Maximum Entropy deconvolution of dirty image

A.5 Display Tasks

- CNTR : Generate a contour plot as TV image or plot file.
- DFTPL : Plots the DFT of an arbitrary point using UV data.
- GRCLEAR : Clear graphic channel(s).
- GREYS : Do grey-scale plot with optional contours.
- HISEQ : Translates image intensities by histogram equalization.
- IMLHS : Display three maps coded as luminosity, hue, saturation.
- IMVIM : Plots one image's values against another's.
- IMWEDGE : Load a step wedge of image intensities on the TV.
- ISPEC : Plot spectrum of a specified portion of an image.
- LWPLA : Send a plot file to a PostScript printer.
- MIXPL : Display plot files on many devices.
- KNTR : Generate a plot file for a contour plot.
- PLCUB : Plot intensity vs x panels on grid of y,z pixels
- PLROW : Plot intensity of a series of rows with an offset.
- PROFL : Generate a plot file for a profile display.
- RGBMP : Create an RGB image from the 3rd dim of an image.
- SLPLT : Plot a SLice file on a chosen device.
- TACUB : Convert image cube for use by VoxVu on the TAAC.
- TAHUE : Load images to the TAAC & LUT/OFM operate.
- TAMOV : Converts image cube for use as a movie on the TAAC.
- TAPLT : Plots data from a Table extension file.
- TKAMODEL : Add slice model display directly on TEK.
- TKARESID : Add slice model residuals directly on TEK.
- TKASLICE : Add a slice display on TEK from slice file.
- TKBOX : Set a Clean box with the TK cursor.
- TKGUESS: Display slice model guess directly on TEK.
- TKMODEL: Display slice model directly on TEK.
- TKNBOXS: Set Clean boxes 1 - n with the TK cursor.
- TKPL: Send a plot file to the TEK.
- TKPOS: Obtain absolution coordinates under cursor.
- TKRESID: Display slice model residuals directly on TEK.
- TKSET: Set 1D gaussian fitting initial guesses.
- TKSLICE: Display slice file directly on TEK.
- TKVAL: Obtain value under cursor from a slice.
- TKWIN: Set BLC and TRC with green screen cursor.
- TKXY: Obtain pixel value under cursor.

- TV3COLOR: Initiate 3-color display using 3 TV channels.
- TVALL: Loads image to TV, shows labeled wedge, enhances.
- TVANOT: Load anotation to the TV image or graphics.
- TVBLINK: Blinks 2 TV planes, can do enhancement also.
- TVBOX: Set boxes with TV cursor & graphics display.
- TVCUBE: Load a cube into tv channel(s) & run a movie.
- TVFIDDLE: Enhances B/W or color TV image with zooms.
- TVFLG: Edit UV data using the TV display and cursor.
- TVFLUX: Displays coordinates selected on the TV.
- TVHLD: Load a map with histogram equalization.
- TVHUEINT: Make hue/intensity display from 2 TV channels.
- TVHXF: Calculate transfer function based on histogram.
- TVINIT: Return TV display to a virgin state.
- TVLABEL: Label the (map) image on the TV.
- TVLOD: Load an image into a TV channel.
- TVLUT: Modifies the transfer function of the image.
- TVMAXFIT: Find coordinates and values on the TV.
- TVMBLINK: Blinks 2 TV planes either auto or manually.
- TVMLUT: Modifies the transfer function of the image.
- TVMOVIE: Load a cube into tv channel(s) & run a movie.
- TVNAME: Fill image name of that under cursor.
- TVOFF: Turns off TV channel(s).
- TVON: Turns on TV channel(s).
- TVPL: Sends a plot file to the TV.
- TVPOS: Obtains TV pixel location under cursor.
- TVPSEUDO: Activate three types of pseudo-color displays.
- TVRESET: Clear the TV from zoom, colors, etc.
- TVROAM: Load upto 4 planes and roam any 512x512 part.
- TVSCROL: Activate scroll of image.
- TVSLICE: Set slice ends with TV cursor & graphics.
- TVSPLIT: Compares 2 TV planes, showing halves.
- TVSTAT: Find the mean, rms and extrema in a TV subimage.
- TVTRANSF: Modifies the transfer function of the image.
- TVWEDGE: Load a step wedge of image intensities.
- TVWINDOW: Set window with TV cursor & graphics display.
- TVWLABEL: Label the (wedge) image on the TV.
- TVZOOM: Activates the TV hardware zoom.
- TXPL: Task to send a plot file to a text file or terminal
- WEDERASE : Load a wedge portion of the TV with zeros.
- XPLOT : Plots image rows on the Graphics (TEK) screen.

A.6 Analysis Tasks

- BLANK : Blanks out non-signal portions of images.
- BLSUM : Sums images over blotch regions including spectra.
- CCMOD : Add model to CC list
- CLIP : Flag data whose amplitude is out of range.
- CNTR : Generate a contour plot as TV image or plot file.
- COMB : Combine in many ways two overlapping images.
- CONVL : Convolve an image.
- CVEL : Shift spectral line data to a given velocity
- FARAD : Add ionospheric Faraday rotation to CL table.
- GAL : Determine parameters from a velocity field.
- HGEOM : Make an image consistent with another image.
- IM2UV : FFT an image and convert to a UV data file.
- IMEAN : Print the mean, rms and extrema in an image.
- IMFIT : Fit gaussian models to an image.
- IMFLT : Flattens an image by fitting a plane to the image.
- IRING : Integrate in confocal ellipses.
- JMFIT : Fit Gaussian models to an image by least-squares.
- MAPIT : CLEAN and Self-Calibrate UVDATA.
- MATHS : Operate on an image with mathematical functions.
- MCUBE : Collect a set of n-dim maps into a (n+1)-dim map.
- MOMFT : Calculate moments of a component.
- MOMNT : Calculate profile moments.
- MWFLT : Applies a nonlinear lowpass filter to an image.
- NINER : Applies various 3x3 area operaters to an image.
- NNLSQ : Non-Negative-Least-Squares decomposition of spectrum
- PADIM : Increase image size by padding with some value
- PBCOR : Apply the primary beam correction.
- PCNTR : Generate plot file for contour plus pol. vectors
- PGEOM : Transform an image into polar coordinates.
- POLCO : Correct polarization maps for Ricean bias
- REGRD : Transforms image coordinate system and geometry.
- RM : Calculate rotation measure and magnetic field.
- SLFIT : Fit gaussians to slice data.
- SLICE : Make a slice file from an image.
- SMOTH : Smooth a subimage from upto a 7-dim. image.
- SQASH : Sums together or average planes in a cube.

- STESS : Finds sensitivity in mosaicing.
- STFUN : Calculate a structure function image.
- SUBIM : Select a subimage from upto a 7-dim. image.
- SUMIM : Sum overlapping, sequentially-numbered images.
- SUMSQ : Sum the squared pixel values of overlapping sequentially numbered images.
- UVFIT : Fit gaussian or spheres to UV data.
- UVLIN : Fit baselines to visibility spectra and subtract.
- UVLSF : Fit baselines to visibility spectra and subtract.
- UVMOD : Inserts a model into uv data.
- UVMTH : Corrects one uv dataset with the average of another.
- UVSEN : Determine rms sidelobe and sensitivity.
- UVSRT : Sorts UV data.
- UVSUB : Subtract CLEAN components from a uv data base.
- WARP : Model warps in galaxies.
- XBASL : Fits n'th order polynomials to rows of images.
- XGAUS : Fits 1-dimensional Gaussians to images.
- XMOM : Finds the moments of each row of an image.
- XSMTH : Smooth data on the x axis only.
- XSUM : Sum or average data on the x axis only.
- XTRAN : Create a new image with transformed coodinates

A.7 Utility Tasks

- ABORTASK : Aborts a task.
- ALLDEST : Destroy all or part of user's files
- ALTDEF : Modify velocity vs frequency relationship.
- AVEOT : Advances a tape to the end.
- AVER : Average 'BT' sorted UV data.
- AVFILE : Advances or backspaces files on a tape
- AVSPC : Average uv data in frequency.
- AVTP : Position tapes without making AIPS wait.
- AXDEFINE : Modify image axis description and values.
- BAKLD : Read image or uv data files written by BAKTP.
- BAKTP : Write maps or uv data with host backup format.
- BLOAT : Make line data base from pseudo-continuum.
- CCFND : Finds negative clean components.
- CCMRG : Merge CLEAN component files.
- CELGAL : Switch between Celestial and Galactic coords.
- CLRALL : Empty the message file.

- CLRZAP : Clear the status on a file and delete it.
- DBCON : Task to concatenate multi-source uv data files.
- DISKU : Find disk use for one or all users.
- DISMOUNT : Software dismount a tape
- EGETNAME : Fill in INNAME, INCLASS, INSEQ; return ERROR.
- EXTDEST : Delete specified extension files.
- EXTLIST : List extension files associated with an image.
- EXTZAP : Delete *all* extension file of given type.
- FETCH : Read an external image in a flexible format.
- FILZAP : Deletes files in a range of slot numbers.
- FITLD : Store an image or UV data from a FITS tape.
- FITTP : Write FITS multi- or single-source file.
- FREESPAC : List space available on disk.
- GET : Recover a full POPS environment saved in.
- GETNAME : Fill in INNAME, INCLASS, and INSEQ.
- GETnNAME : Fill in INnNAME, INnCLASS, and INnSEQ (n=2:3).
- GETONAME : Fill in OUTNAME, OUTCLASS, and OUTSEQ.
- GETTHEAD : Read table header keywords into adverbs.
- IMCOP : Task to copy image catalog file to another user.
- IMERASE : Load a map portion of the TV with zeros.
- IMHEADER : List the image header.
- IMLOD : Store an image from a FITS or IBM-CV tape.
- IMMOD : Alter images with a model.
- IMPOS : Determine the absolute coordinates under cursor.
- IMSTAT : Verb to find the mean, rms and maximum in an image
- IMVAL : Determine pixel value at specified position.
- IMXY : Determine pixel coordinates under cursor.
- ISCOMP : Lists the tasks which handle compressed data currently.
- MAXFIT : Find coordinates and value of an image extremum.
- MOUNT : Software mount a tape.
- MCAT : List catalogue of image (map) data files.
- NEGPH : Negate visibility phases.
- OFMADJUS : Modify the TV OFM lookup tables interactively.
- OFMCONT : Create/modify TV OFMs w wedge/level contours.
- OFMDIR : List all OFM files accessible to the user.
- OFMGET : Load the TV's OFM LUTs from OFM save file.
- OFMLIST : Print or type the TV's current OFM lut(s).
- OFMSAVE : Save the TV's current OFM lookup table in file.

- OFMTWEAK : Modify the current OFM LUTs with the trackball.
- OFMZAP : Delete an OFM save file.
- PATGN : Create a user specified test pattern.
- PCAT : List entries in the user's catalog (no log file).
- PLZAP : Delete all PL files.
- PRTAB: Print any table-format extension file.
- PRTAC : Print the accounting log file.
- PRTAN : Print the Antenna (AN) extension of a uv file.
- PRTCC : Print CLEAN component files.
- PRTHI : Print history file.
- PRTIM : Print the image intensities in digital form.
- PRTMSG : Print the message file.
- PRTUV : Print UV data stored on disk.
- PUTHEAD : Modify image header parameters.
- PUTTHEAD : Modify table header values.
- PUTVALUE : Store a pixel value at specified position
- QHEADER : Summarize the image header: positions at center.
- QIMVAL : Determine pixel value at specified position.
- REBOX : Reset boxes with TV cursor & graphics display.
- RECAT : Compress the entries in a catalog file.
- REMAG : Replace magic blanks with a user specified value.
- REMOVIE : Rerun a previously loaded (TVMOVIE) movie.
- RENAME : Rename a file.
- RENUMBER : Change the catalog number of an image.
- REROAM : Use previous roam image mode, then does roam.
- RESCALE : Modify image scale factor and offset.
- RESTART : Trim the message log file and restart AIPS.
- REWIND : Rewind a tape.
- SAVDEST : Destroy all save files of a user.
- SAVE : Save full POPS environment in named file.
- SCRATCH : Delete a procedure from the symbol table.
- SCRDEST : Destroy scratch files left by bombed tasks.
- SETAN : Read antenna file info from a text file.
- SETXWIN : Set BLC and TRC with TV cursor.
- SGDESTR : Destroy named POPS environment save file.
- SGINDEX : Lists SAVE areas by name and time of last SAVE.
- SHOW : Display the TELL adverbs of a task.
- SL2PL : Convert a Slice File to a Plot File.

- SLCOL : Collate slice data and models into text files.
- STARS : Generate an ST ext. file with star positions.
- STQUEUE : List pending TELL operations.
- TABGET : Return specified table entry
- TABPUT : Replace specified table entry
- TAFLG : Flags data in a Table extension file
- TAMRG : Merge table rows under specified conditions
- TASAV : Copy all extension tables to a dummy uv-file
- TASRT : Sort extension tables.
- TBAVG : Time averages all data on all baselines.
- TBIN : Read AIPS tables from text files.
- TBOUT : Write AIPS tables to text files.
- TELL : Send parameters to a task from AIPS in execution.
- TGET : Gets adverbs from last GO of a task.
- TGINDEX : Lists those tasks for which TGET will work.
- TIMDEST : Destroy all files which are too old.
- TPHEAD: List image header from FITS or IBM-CV tape.
- TPUT: Puts adverbs from a task in file for TGETs.
- TRANS: Transpose a subimage of an up to 7-dim. image.
- UCAT: List UV and SC data files in the user's catalog.
- USUBA : Assign selected data to a subarray.
- UVAVG : Average or merge 'BT' or 'TB' sorted UV data.
- UVCMP : Converts UV data to compressed format.
- UVCOP : Copy selected times and channels of uv data.
- UVDGP : Copy a visibility file, omitting the Mth Nth.
- UVDIF : Print differences between 2 UV disk files.
- UVFIL : Creates uv data with a user's subroutine.
- UVFIX : Recompute u,v,w terms for a uv database.
- UVGLU : Glues uv frequency blocks back together.
- UVSIM : Generate specimen u-v coverage.
- VBCAL : Multiply antenna amplitude gain factors.
- VBMRG : Merge 'BT' sorted VLBI data.
- WAITTAS : Halts AIPS until the task is finished.
- WTMOD : Modify weights in a uv data set.

Last update : 10/10/92

B DISCUSSION OF SOME SELECTED *AIPS* CALIBRATION ADVERBS

Here follows a description of some of the *AIPS* calibration adverbs. More details about adverbs can be found in the *AIPS* HELP files.

Sources can be selected in a variety of ways by referring to them by name, by frequency, and by coded strings and numbers. All of these methods can be useful, and you will probably evolve your own method which uses some subset of them.

- **calsour** is an array for selecting one or more calibration sources by name.
- **qual** is a number that qualifies which source you are selecting. For example, the same source name can have different source qualifiers for different setups in the observation. ATLOD makes use of this facility for different **freqids**.
- **calcode** is a very useful adverb, also used to make additional qualification of calibrator sources. It may be set by the online system and thus entered into the SU table when you load your data into *AIPS*. But whether it is or isn't set, you will be able to change it to whatever you like with SETJY (see § 5). Then, it is often convenient to select sources by just their **calcode** instead of having to type their names into the **sources** adverb list all the time.
- **selband** and **selfreq** allow you to select data by absolute bandwidth and frequency. I think they are cumbersome and suggest that the adverb **freqid** discussed below is more useful.
- **freqid** specifies which frequency by a code number. The **freqids** and their associated frequencies can be seen in the output from LISTR discussed in § 5. **freqid** is useful because each frequency must be calibrated separately, so keeping all other source selection parameters the same, but changing **freqid** allows simple change of frequency. This is handy when your observation has been set up with continual cycling through the same groups of frequencies (e.g., to obtain separated frequencies for multi-frequency synthesis). **freqid** values are stored in the FQ table, and must be specified one at a time; there is no way to get all of them selected together (except in UVPLT). Note that the **selband** and **selfreq** adverbs overide the **freqid** selection. However, any ambiguity must be resolved by **freqid**.
- **bif** and **eif** refer to the IF axis which is displayed in the header (use verb IMHEAD to see it) of the visibility data base. There is some confusion about information stored in FQ tables and the IF axis. The IF axis allows storage of one group of equally spaced frequency channels (i.e., spectral channels) per IF frequency, and unevenly spaced IF frequencies. The disadvantage of the IF axis is that, unlike the use of the FQ table, all IFs must have the same bandwidth and channel increments. The advantage is that it is an axis which the calibration software can loop over. The **freqid** is a random parameter which cannot be looped over (i.e. you cannot select *all* **freqids** together). AT data which contains simultaneous frequencies is put on the IF axis when possible.
- **docalib** specifies whether you want to apply a calibration table to the data before performing the operation of the current task. When you are calibrating, usually you want this set to -1 (false) because there is no calibration to apply yet. However, following successful calibration, you will want to apply it to the plotting and imaging routines, so you put **docalib** $= 1$ then.
- **gainuse** specifies the calibration (CL) table containing the calibration corrections for use when **docalib** is true.
- **gainver** specifies the input CL table to which an SN table is to be applied thus producing an output CL table (pointed at with **gainuse**.
- **flagver** is used to specify which FG table contains the flagging information. Note that the null value (0) is interpreted inconsistently in *AIPS*. Some tasks will take 0 to mean the highest version FG table (use IMHEAD to see how many you have), and others assume 0 means 1. Always check the HELP file to clarify this. If **flagver** is negative, no flagging table is applied. Make sure you have **flagver** set whenever you have done some editing.
- **doband** specifies whether you want to apply the bandpass correction BP table (computed by the task BPASS) to the data before performing the operation of the current task. For example, you would turn this on when imaging every channel in a spectral-line data base. See HELP **doband** for some extra information about fine distinctions in what value to give to **doband**. This is only necessary if you retain all the channels across the band. In simple continuum observations, you may end up with just one channel (see § 6), in which case this is irrelevant.

- **bpver** specifies which version of the BP table to apply if **doband** > 0.
- **smooth** specifies what sort of spectral smoothing to apply to the data before performing the operation of the current task. For example, it may be advantageous to apply Hanning smoothing to reduce ringing. See **HELP smooth** for details.

Last update : 27/11/93

C COMMONLY ENCOUNTERED TABLES IN *AIPS*

AIPS files generally have a number of tables associated with them which contain ancillary information about the file. In this Appendix I will describe briefly what the most common tables you will encounter are for. The *AIPS* command **IMHEAD** will enable you to see which tables you have associated with a file and you can list the contents of these tables with the task **PRTAB**.

PRTAB	
inext='FQ'	Select table type, FQ for example
inv=0	Highest table version
xinc=1	Print all rows
docrt=−1	List on line printer
docrt=132	List on terminal, 132 char per line, must switch terminal to 132 char output too
dohms=1	HH MM SS times instead of decimal days

1. Primarily, the antenna (AN) table keeps geometric array information such as the locations and names (e.g. CA03) of each antenna. It also contains some time (e.g. earth rotation rate and conversion from atomic to sidereal time) and polarization information (such as the polarization states of the feeds; circular or linear). This table is needed should you have to recompute the (u, v, w) coordinates in a single-source visibility file with **UVFIX**, for example. All the calibration routines will access this table for its more mundane entries about antenna numbers, but in general the user does not need to worry about it. It is created by any *AIPS* task that reads in UV data (specifically **ATLOD** for ATCA data) and can be printed in a nice format with **PRTAN**.

2. The baseline (BL) table contains complex gains for each baseline necessary to correct for errors that are not antenna based (see § 3.1). These errors are assumed to consist of a multiplicative and an additive portion. We try to correct for baseline-based errors only if extremely high dynamic range is required as they are usually small. The basic calibration will not require the BL table; it is created by the task **BLCAL**. The desired version of the BL table is selected with the adverb **blver**.

3. The bandpass (BP) table contains the response across the band. Since the complex antenna gains are frequency dependent you need to know how they change with channel (frequency) as well as time. If you are making standard VLA continuum observations then there is only one channel and no bandpass to correct for. However, the ATCA, even when making continuum observations, produces multi-channel data so that unless you choose to average them all into one channel, you should determine the response across the band. The BP table is created by the task **BPASS** or the procedure **ATBPASS**.. The desired version of the BP table is selected with the adverb **bpver**.

4. The calibration (CL) table contains calibration information and the corrections which should be applied to the data to calibrate them. Essentially, this is where the complex gains as a function of time are stored for each source in the data base that you are interested in, whether it be a program source or a calibrator.

   You can have many versions of CL tables, and select the appropriate one with the adverb **gainuse**. For ATCA data, CL table 1 is a pristine copy with values appropriate to ideal gains. It is created by **ATLOD**. You should not delete this pristine version, but if you do, you can regenerate it with the task **INDXR**.

   The gains are stored in the CL table at time intervals generally of the order of minutes (although the integration time might be seconds). They are arrived at by smoothing and interpolating the gain solutions stored in the SN table with the task **CLCAL** or the procedure **ATCLCAL**.

   Note that CL tables are *cumulative*. This means that when you generate a new CL table, you *always* do it by first reading a previous version CL table (selected with adverb **gainver**), apply the gain solutions in the desired SN table (selected with adverb **snver**) to it and write out the new CL table (selected with adverb **gainuse**). In this way you can incrementally build up a better and better calibration which is useful for self-calibration applications (see § 17).

5. The flagging (FG) table contains information specifying which visibilities are bad so that they may be ignored. It is generated by the editing tasks (**TVFLG**, **SPFLG**, **IBLED**, **UVFLG**). The desired version of the FG table is selected with the adverb **fgver**.

6. The frequency (FQ) table contains information about the different frequencies and bandwidths in the multi-source file for different sources. It is possible in this way for the multi-source data base to contain

multi-frequency and multi-bandwidth data. For ATCA data, the FQ table is generated by ATLOD. It cannot be rebuilt without rerunning ATLOD if you delete it (although if you are careful, you could copy it from a like file with TACOP).

Each different group of simultaneously observed frequencies (such as the two simultaneous IFs you get with the ATCA) is designated by an integer number (the FREQID). Each FREQID has one logical row in the FQ table. For each simultaneous frequency (IF) contributing to the FREQID, the logical row has an offset in Hz from the reference frequency in the header, a channel width, a total bandwidth and a sideband indicator. Thus, the logical row is multi-dimensional; there are as may actual rows in the logical row as there are simultaneous frequencies (IFs).

If you observed, say, at 3 cm (IF 1) and 6 cm (IF 2), and then changed to 13 cm (IF 1) and 20 cm (IF 2), you would get two FREQIDs in the FQ table. The first would describe the 3/6 combination, the second the 13/20 combination.

You select the desired FREQID in the FQ table with the adverb freqid.

For tortuous historical reasons, the channel width is now a signed quantity, and the sideband indicator is now always written as +1. The true sideband indicator is written to the screen when you load your data with the task ATLOD. It is also written into the history file if you forget it. It varies from band to band.

Occasionally you may come across CH tables. These are old deprecated versions of the FQ table. The FQ table is created by ATLOD. Do not delete it or you will have to make a new one with ATLOD.

7. The index (NX) table indexes the multi-source data base for rapid access by containing information about scan boundaries such as times and visibility number ranges. It is created by ATLOD or INDXR. If deleted, it can be regenerated with INDXR. There is only one version of the NX table and you should not need to concern yourself with it.

8. The solution (SN) table contains the complex gain solutions determined from the calibrator sources. It is generated by the tasks CALIB or KALIBand the procedure ATCALIB. The values in the SN table are interpolated to the times where you have observed your program source. YOu select the desired version of the SN table with the adverb snver.

9. The source (SU) table contains source specific information such as flux densities, positions, and calibration codes. This table is built by ATLOD. It **cannot** be regenerated by any other task so don't delete it. Each source in the multi-source visibility file is given a number which is an index to the SU table entry. There is only one version of the SU table. Do not delete it.

Last update : 27/11/93

D EFFECTS OF AVERAGING IN FREQUENCY AND TIME

There are many descriptions of these two averaging processes in the literature (e.g., Thompson, Moran, and Swenson 1986, NRAO Synthesis workshop 1988) so I will just give an outline and show the pertinent results.

D.1 Averaging in frequency

If we average all the channels in the band together, then we must assign a single frequency and a single (u, v) coordinate to each visibility. Clearly this means a loss of information, because u and v change across the band (they are the orthogonal baseline components measured in wavelengths in the plane perpendicular to the direction of the source, with v in the northern direction).

The simplest way to understand the effect upon the final image is via the similarity theorem of Fourier transforms, which states that if $h(\mathbf{u})$ and $H(\mathbf{x})$ are a Fourier pair in n dimensions, then rescaling the coordinates in one domain by a factor of α corresponds to rescaling the transform by the reciprocal scale factor, and renormalizing the amplitudes such that

$$\frac{1}{|\alpha|^n} H\left(\frac{\mathbf{x}}{\alpha}\right) \rightleftharpoons h(\alpha \mathbf{u})$$

In our case, we associate the visibilities (V) with h and the image (I) with H. Thus, we measure a visibility $V(u, v)$ at some frequency ν in the band, and then scale its coordinates to (u_0, v_0) (appropriate to the frequency at the centre of the band) such that

$$(u, v) = \left(\frac{\nu}{\nu_0} u_0, \frac{\nu}{\nu_0} v_0\right).$$

The similarity theorem shows that the image made from visibilities that have been scaled in this way is stretched and scaled such that the contribution to the visibility from a narrow band of frequencies centered on ν is

$$V(u_0, v_0) = V\left(u\frac{\nu_0}{\nu}, v\frac{\nu_0}{\nu}\right) \rightleftharpoons \left(\frac{\nu}{\nu_0}\right)^2 I\left(\frac{\nu}{\nu_0} l, \frac{\nu}{\nu_0} m\right),$$

where l and m are the image coordinates (direction cosines). Each image is the true brightness distribution scaled in (l, m) by ν/ν_0 and in brightness by $(\nu/\nu_0)^2$. The derived distribution is convolved by the dirty beam $B_D(l, m)$. The beam does not vary with frequency because the same transfer function for the telescope (i.e., the sampling of the array in the (u_0, v_0) plane) is used to represent the whole frequency passband. The overall response over the whole band is simply obtained by integrating over the passband with the appropriate weighting.

$$I_D(l, m) = \left[\frac{1}{\int_0^\infty G(\nu) d\nu} \int_0^\infty \left(\frac{\nu}{\nu_0}\right)^2 G(\nu) I\left(\frac{\nu}{\nu_0} l, \frac{\nu}{\nu_0} m\right) d\nu\right] \otimes B_D(l, m),$$

where $G(\nu)$ is the passband function. This integral can be thought of as a process of averaging many images, each with a different scale factor. The images are aligned at the phase centre, so that the effect is to produce a radial smearing of the brightness distribution (and consequent loss of intensity) before it is convolved with the beam. The response to a point source at position (l, m) is radially elongated by the factor $\sqrt{l^2 + m^2}\Delta\nu/\nu_0$. For distances from the origin at which the elongation is large compared to the synthesized beamwidth, features on the sky become suppressed by the smearing; the integrated flux density remains unchanged, but the surface brightness is reduced. The measured brightness is the smeared distribution convolved by the beam.

It turns out that the results are only weakly dependent on parameters such as the exact passband shape and beam shape, so, as a good example, the figure shows the relative amplitude of the peak response to a point source as a function of distance from the field centre (r), the beamwidth (θ_b), and fractional bandwidth ($\Delta\nu/\nu_0$). It can be used to determine whether your data will suffer too badly should you average across the band. As a rough guide, you probably don't want to accept a bandwidth that results in an amplitude drop to less than ~ 0.9.

Note that since the AT always produces spectral data, you are free to average channels in any combination you choose. You are not restricted to averaging the whole band and might desire to average it into three sections say, none of which suffers from bandwidth smearing.

D.2 Averaging in time

The output from the correlator is integrated for some period, T_a, and then assigned the mean time for that integration period. For a source at a celestial pole, this would amount to a rotation of a visibility through the

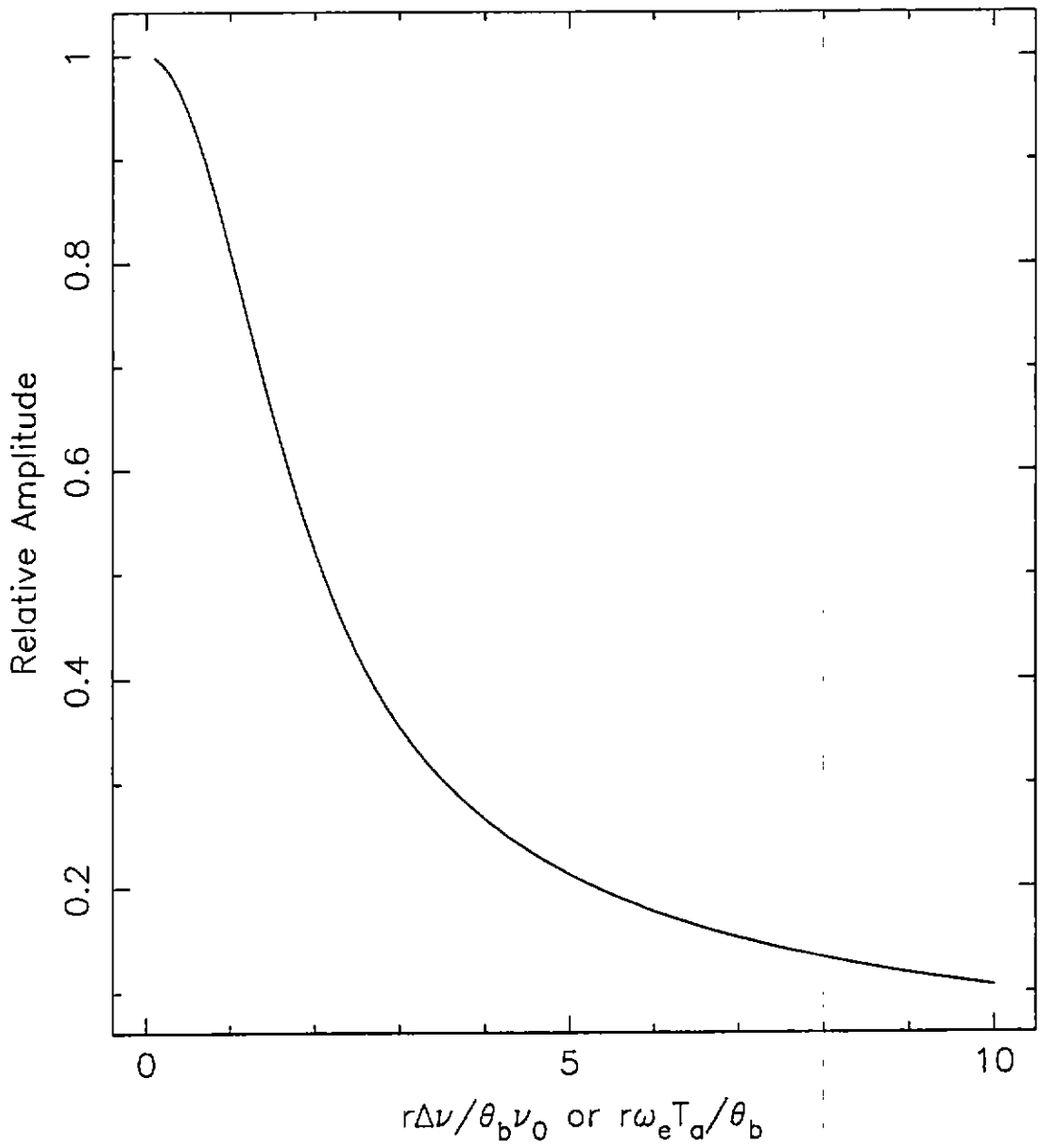


Figure 1: Relative amplitude of the peak response to a point source as a function of the distance from the field centre and either the fractional bandwidth or the averaging time

angle $\omega_e \delta t$, where ω_e is the earth's angular velocity (7.27×10^{-5} rad s^{-1}) and δt is the timing error in the assignment of u and v. This would cause the image to be rotated through the same angle. For a pole source then, this error is the same as averaging a series of images aligned at the phase centre, but with angular offsets up to $\pm\omega_e \tau_a/2$. The final image would thus reflect an azimuthal distortion which is more severe with increasing distance from the phase centre. This is in some sense complementary to bandwidth smearing which produces a radial distortion. The general case is of course more complicated than this. A useful result that is valid for a variety of source locations and array types is very similar to that of bandwidth smearing; the same figure can be used for the decrease in amplitude of a point source with distance from the phase centre, with a change of abcissa, as shown in the figure.

Last update : 27/11/93

REFERENCES

1. *AIPS* EXPLAIN file CALIBRAT
2. Clark, B.G., 1980, A&A, **89**, 377.
3. Cornwell, T.J.,1981, VLA Scientific Memorandum no. 135.
4. Cornwell, T.J.,1983, A&A, **121**, 281.
5. Cornwell, T.J., Uson, J.M., and Haddad, N., 1992, A&A, 258, 583.
6. Högbom,J.A., 1974, A&A Suppl., **15**, 417.
7. Langenvelde, H.J. van and Cotton, W.D., A&A, 239, L5.
8. NRAO synthesis workshop, 1986.
9. NRAO synthesis workshop, 1988.
10. Sault, R.J., Killeen, N.E.B., and Kesteven, M.J., 1992, ATNF scientific memorandum, Polarization Calibration of the ACA
11. Sault, R.J., 1991, ATNF scientific memorandum, Some simulations of self-calibration for the AT
12. Sault, R.J., 1992, ATNF scientific memorandum, Multi-frequency Synthesis with the ATCA
13. Sault, R.J., 1993, preprint.
14. Schwab, F., 1984, A.J., **89**, 1076.
15. Steer, Dewdney, Ito, 1984, A&A, **137**, 159
16. Thompson, Moran, and Swenson, 1986, "Interferometry and Synthesis in Radio Astronomy"
17. Wieringa, M., 1992, ATNF scientific memorandum, Measurements of the ATCA primary beam

Last update : 10/10/92

Index