8 Spectral-Line Software

This chapter deals with the analysis and reduction of spectral-line data after they have had the basic calibrations, described in Chapter 4, applied. Spectral-line software generally involves three-dimensional images, often called "cubes", in which one of the three axes is frequency or velocity. Special programs are available to build, manipulate, and transpose these cubes and to display them properly. Most of the continuum software will work on data cubes or on appropriate two-dimensional subsets from a data cube. Spectral-line *uv* data can be read into $\mathcal{AIPS}$ from "*uv*" FITS tapes. Often, however, the data already exist on disk having been calibrated first within $\mathcal{AIPS}$.

The data reduction process at this point has a number of stages, namely data preparation and assessment, self-calibration, continuum subtraction, imaging, display and manipulation of data cubes, and analysis. This chapter will address each of these areas in varying, but modest, detail on the assumption that the reader is somewhat familiar with the contents of previous chapters in this $\mathcal{C}ook\mathcal{B}ook$. Some aspects of the art of spectral-line imaging are discussed in Chapters 17 and 18 of *Synthesis Imaging in Radio Astronomy*[1]. A brief outline of the basic calibration process is given in Appendix A of this $\mathcal{C}ook\mathcal{B}ook$ and an outline of spectral-line analysis and calibration is given in Appendix B.

8.1 Data preparation and assessment

In the following sections, it is assumed that the *uv* data are on disk and that the basic calibrations have been applied to them. If your data are not currently on disk, you will need to read them in from magnetic tape or FITS disk files. For tape, mount the tape in hardware and software (§ 3.9) and then position the tape to the desired data files:

> NFILES *n* ; AVFILE C_R	to advance *n* files.
> TPHEAD C_R	to check that the tape is correctly positioned.

Use `FITLD` to read in the data:

> TASK 'FITLD' ; INP C_R	to review the inputs.
> OUTCL ' ' ; OUTN ' ' C_R	to use the file names recorded on tape (the default).
> NFILES 0 C_R	to read the current file on tape.
> NCOUNT 2 C_R	to load two files (if desired).
> GO C_R	to run `FITLD`.

For FITS disk files, set `DATAIN` appropriately and the run `FITLD`. Examine the *uv* data set header with `IMHEAD` after `FITLD` finishes. It should show multiple pixels on the `FREQ` axis, like the *uv* header illustrated in § 3.3.4.

If your data are not yet calibrated, consult Chapter 4, especially § O.1.7. When the `CL`, `BP`, and other calibration tables are complete, apply them to the line data with `SPLIT`:

> TASK 'SPLIT' C_R	
> SOURCE '*sou1*' , '*sou2*' , . . . C_R	to select sources, ' ' means all.
> TIMERANG 0 C_R	to keep all times.
> BIF 1 ; EIF 0 C_R	to keep all IFs.
> BCHAN 1 ; ECHAN 0 C_R	to keep all spectral channels.
> FREQID 1 C_R	to set the one `FQ` value to use.
> DOCALIB TRUE C_R	to apply calibration.

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

> GAINUSE 0 C_R	to use the highest numbered CL table.
> FLAGVER 1 C_R	to apply the flag table.
> DOPOL TRUE C_R	to correct for feed polarization.
> DOBAND 1 C_R	to correct bandpass.
> BPVER 1 C_R	to select BP table to apply.
> STOKES ' ' C_R	to write the input Stokes type.
> DOUVCOMP TRUE C_R	to write visibilities in compressed format.
> APARM 0 C_R	to clear VLBA options, including the one to calibrate the data weights.
> INP C_R	to review the inputs.
> GO C_R	to run the program when inputs set correctly.

The *uv* data produced by this process should have applied to them the full calibration and editing determined from the calibration sources. Whether these are adequate or will need to be enhanced from the source data themselves is now to be determined.

Your data may need to be edited further at this point. If your calibrator data were almost free of problems, save for simple matters like dead antennas, then your source data are not likely to need much editing either. If your calibrator data required detailed and erratic data flagging, then your source data will also need attention. All of the considerations of § 4.3.5 apply here too. In either case, the first step is to check for narrow-band interference which is a fairly common problem at some wavelengths. The best task for this is SPFLG (§ 10.2.2), a spectral-line version of TVFLG (§ O.1.6). SPFLG displays spectral channels for all IFs along the horizontal axis and time along the vertical axis, one baseline at a time. This is tedious to use for editing all of the data from a large interferometer. Instead, select a few of the shorter baselines (with the ANTENNAS and BASELINE adverbs) and do a quick check for interference. If your data has sufficient signal-to-noise, then this quick examination should let you check which channels have signal and whether the Galaxy, for example, has contributed unwanted signal to your data. If you find significant terrestrial interference, you may flag the bad data with SPFLG (consider using the "ALL-BL" option to avoid looking at hundreds of baselines) or with UVFLG (entering "by hand" what you find with SPFLG). Tasks UVLIN and UVLSF (below and § 8.3) both offer the option to flag all channels of a particular sample if any one channel in the continuum-fitting portion of the spectrum has excessive flux after the fit. UVLSF also offers the option to flag when the rms of the continuum fit is excessive. The task CLIP (§ 5.5.1) works in a channel-dependent way on line data. It can be instructed to flag a full spectrum if more than a user-specified number of channels are flagged in that spectrum. FLGIT is designed to work on line data sets that have a great deal of interference, generating channel-dependent flagging in the output file or in a flag table. UVMLN may also be used to generate flag tables for multi- and single-source files to flag seriously deviant data. FLAGR uses multi-channel data to estimate rms and flags any times having excessive rms, amplitude, weight, etc. RFLAG will flag data in detailed ways based on excessive variability in time and/or frequency. REFLG will compress the large flag table made by RFLAG. UVCOP will apply even a very large flag table if so instructed. These tasks will eliminate certain classes of problems without bothering you with the details, which may — or may not — be a good thing.

To do serious editing of a non-spectral nature and to do self-calibration, you will need to create a single-channel data set that represents your multi-channel data in some fashion. If your data contains little continuum signal, but does have a strong spectral-line signal, then use UVCOP to copy the data for the channel with the strongest signal to a single-channel data set. Alternatively, the task AVSPC may be used to average a few adjacent channels together to create the single-channel data. If, on the other hand, your data set contains a useful level of continuum emission — and most do — then your single channel should be some estimate of this continuum. If your line signal is weak and you have done no frequency-dependent editing, then you can use the so-called "channel-0" data set created by FILLM when you first loaded your VLA data to disk. This data set is the vector average of the center 3/4 of the selected spectral channels, but remains useful only if you have applied all of the same calibration, editing and SPLITing to it that you applied to the line data. It may be safest to create a new continuum estimate from the line data in any case.

The classical method for doing this is the task `AVSPC`:

> TASK 'AVSPC' ; INP C_R	to review the inputs needed.
> IND *m* ; GETN *n* C_R	to specify the input *uv* file.
> CLRONAME C_R	to use the default output names and disk.
> AVOPT ' ' C_R	to average channels only within IFs.
> ICHANSEL *b1* , *e1* , *i1* , *if1* , *b2* , *e2* , *i2* , *if2* ,. . . C_R	to average every *i1* channel from *b1* through *e1* in IF *if1*, plus every *i2* channel from *b2* through *e2* in IF *if2*, etc.
> ICHANSEL 10 , 50 , 1, 0, 81 , 119 , 1 C_R	*e.g.*, average channels 10 through 50 and 81 through 119 in each IF, omitting the band edges and the channels with line signal.
> INP C_R	to review the inputs.
> GO C_R	to run the program when you're satisfied with inputs.

A non-classical method for estimating the continuum is provided by the task `UVLSF` It fits a linear "spectral baseline" in the real and imaginary parts of each visibility spectrum. By adding a slope to the channel-independent bias found in `AVSPC`, it allows for small changes in the continuum visibility with frequency and spatial resolution across the observed band, as well as small, time-variable changes in the spectral passband not fully removed by `BPASS`. The option to fit even higher order baselines was added. These are seldom useful and can run into problems since they are not constrained by data in the channel ranges containing the line signals. The principal output of `UVLSF` is a multi-channel data set with this baseline subtracted, *i.e.*, a continuum-free line data set. However, the user may also get a single-channel "continuum" data set evaluated at any one channel from the fit baselines. Thus:

> TASK 'UVLSF' ; INP C_R	to review the inputs needed.
> IND *m* ; GETN *n* C_R	to specify the input *uv* file.
> CLRONAME C_R	to use the default output names and disk.
> BCHAN 0 ; ECHAN 0 C_R	to include all channels.
> ICHANSEL *b1* , *e1* , *i1* , *if1* , *b2* , *e2* , *i2* , *if2* , . . . C_R	to average every *i1* channel from *b1* through *e1* in IF *if1*, plus every *i2* channel from *b2* through *e2* in IF *if2*, etc.
> DOOUTPUT TRUE C_R	to get the continuum data set.
> CHANNEL 0 C_R	to select the channel at which the baseline is evaluated — in this case, the default which is the reference channel. Choose one near the center of the band.
> INP C_R	to review the inputs.
> GO C_R	to run the program when you're satisfied with inputs.

The continuum data set will have AIPS class `BASFIT` and be on the same disk as the output line data set. You may also use `UVLSF` to flag data with excessive residual flux and rms in the channels used for the fitting. You may apply the usual data selection and calibration and flag tables to the input data, meaning that `SPLIT` need not have been run on the data previously. Note that neither `AVSPC` nor `UVLSF` should be used if the continuum visibility changes substantially across the passband, due to large bandwidth and complexity in the continuum source. If the phase of the continuum changes substantially across the passband due to the centroid of the continuum being considerably separated from the pointing center, you should instruct `UVLSF` to shift the phase reference position before fitting the linear function and then to shift the phases back afterwards.

Your spectral-line data may need further modification and preparation. If you did not observe with "Doppler tracking" (and even the VLA changes frequency only on a scan-by-scan basis), then your data channels are not at the same velocity through the data set. Task `CVEL` may be used to convert the data to full Doppler tracked form, or simply to shift the center velocity of your reference channel to, for example, the velocity used in another data set. `CVEL` works on multi-source as well as single-source data sets. It applies any flagging and bandpass calibration to the data before shifting the velocity (which it does by a carefully correct Fourier transform method). Note, the use of Fourier-transforms means that one *must not* use CVEL an data with channel separations comparable to the widths of some of the spectral features.

Three other tasks allow you to change the number of spectral channels in your data set. The simplest, AVSPC, allows you to average every *n* adjacent channels together, producing a data set one $n^{\rm th}$ as large. The second, called SPECR, uses Fourier transform methods to increase or decrease the number of channels with a corresponding change in channel spacing. SPECR can also use simple interpolation methods in case the Fourier methods produce undesirable results (such as ringing when a feature is very narrow in the input data set). The third task is called UVFRE and it attempts to convolve the data from one frequency structure onto another frequency structure defined in a second data set (the uv equivalent of HGEOM). These tasks, along with CVEL may be required to convert one data set to the same channel velocity and separation as another data set, before the two can be concatenated (task DBCON) and used together in imaging. If simple frequency smoothing is needed, the calibration routines will apply various convolutions to the spectrum under control of the adverb SMOOTH. IMAGR, allows you to "average" channels together by gridding them into the correct (channel-dependent) locations in the uv plane. This is seldom important to narrow-band observations, but is critical in wide "bandwidth synthesis" observations. Nonetheless, it allows you to keep full spectral resolution in the uv data set, but abandon it for some or all of the imaging.

8.2 Editing and self-calibration

Self-calibration can improve the "dynamic range" of your images significantly but **only** if there is sufficiently good signal-to-noise in the uv database. It works by comparing the input uv data with the predicted visibilities from a model of the source; from this a set of complex gain (amplitude, phase) corrections are generated for each antenna in the array as a function of time. Before engaging in a potentially long and useless exercise, it is wise to look at the continuum or single-channel data set you have created to determine whether there is sufficient signal (with respect to the noise) to enable detailed editing and/or self-calibration. Even if the continuum data would profit from these things, you must also decide whether there is sufficient signal-to-noise in the line data to benefit from the improved editing and calibration. Consult the sections on editing (§ 4.3.5 and § 4.3.13) and on self-calibration (§ 5.4) before deciding to continue with this section.

In fact, there is very little to be written here. The editing and self-calibration of the continuum or single-channel data set are precisely those described in Chapter 4 and Chapter 5. The goal is to create and fill an FG table (if flagging is needed) and an SN table (if self-calibration is needed) attached to the single-source, single-channel data set. Note that, for single-source data, we use an SN table containing the accumulated calibration while we use, for multi-source files, a number of SN tables with incremental calibrations and a single CL table with the net corrections. When all editing and self-calibration are done, you use TACOP (or TAPPE if the flag table must be appended to an existing flag table) to copy the two tables to the multi-channel data set and then use SPLIT to apply both tables to the multi-channel data (and to the single-channel data set too if desired).

In 31DEC22, task CLOSE may be used to plot the rms closure phase or amplitude as a function of channel. Use this on your calibration source or your target data after division by the continuum model. Values should be near zero but will probably rise at the edges of spectral windows (IFs). See § 5.4.4.

In general, you may keep the flagging information in a flag table (specified with adverb FLAGVER) and add to and apply the table whenever it is needed. Be aware that flag tables are applied only by those tasks that have the FLAGVER adverb, but a number of tasks have been upgraded to understand calibration and flagging tables. In fact, data that are not in time order may have flag tables applied (if they are small enough) and time-independent calibrations (*e.g.*, DOBAND=1 and DOPOL > 0) may also be done.

8.3 Continuum subtraction

Most spectral-line observations contain a certain amount of frequency-independent, continuum radiation in addition to the frequency-dependent line signals. In most (all?) cases, it is probably best to separate the two signals at this stage of the data processing. In this way, a *single* continuum image can be constructed to apply to all frequencies. It will probably be necessary to apply Clean or other expensive deconvolution techniques to determine the best estimate of this continuum image. To do this for each channel individually would not only be considerably more time consuming, but would result in different models for the continuum in each channel. Since Clean is a data-adaptive non-linear algorithm, minor disturbances in its progress causes it to converge to surprisingly different solutions. Such "disturbances" could be caused by differing noise and line signals in different channels. (It can be caused simply by differing computational order on identical data using different computers.) Clean also increases the noise in an image in the areas at which there are sources. If the continuum is removed from the line data before Cleaning, then this increase in noise is also removed. To determine the continuum from separately Cleaned channel images is to use noisier individual determinations to get the final estimate. If the imaging and deconvolution were a linear process, then this would not matter. But the deconvolution is non-linear, making it better to start with the best possible estimate of the continuum.

In many cases, the continuum signal is stronger by far than the line signal. In such cases, it is best to use as many channels as possible to estimate the continuum and to remove that estimate as early as possible. The remaining line signal may be of fairly low signal-to-noise ratio and, therefore, not need the same processing that the continuum signal requires. In particular, it may not be necessary to Clean the channel images particularly deeply, if at all.

All of the arguments above suggest that the continuum should be estimated and subtracted in the visibility domain. This has the unfortunate attribute that we can use only those channels which are free of line signal over the entire field of view. (This is because the Fourier transform relation mixes signals from all directions in the field into every visibility sample.) In the image domain, the dirty images also have this unfortunate attribute, in this case because the dirty beam mixes signals from all parts of the field into every pixel. The above arguments suggest that the only time one should determine the continuum in the image domain is when the observations have essentially no channels which are free of line signal over the full field of view. In that case, the continuum at each position will have to be determined from deconvolved images using those channels which are free of line signal at that position.

AIPS provides three separate tasks for fitting and subtracting the continuum in the *uv* plane. All three fit a linear "baseline" to a selected group of channels and subtract that from the data. The reasons for a linear baseline are (1) a continuum source offset from the field center will produce a linear phase slope across the passband, (2) minor other changes in the visibility with baseline may be approximately linear over narrow passbands, and (3) minor passband variations with time appear also to be approximately linear. Higher-order fits can be done by `UVLSF`, but they do not seem to be justified and are often not well constrained. The first reason suggests that one should do the fit in amplitude and phase, an algorithm implemented by the task `UVBAS`. This should be used *only* if you have good signal-to-noise in *all* of your visibility samples. The reason for this very stringent requirement is that visibility amplitudes have Ricean rather than Gaussian noise statistics. As a result, very biased estimates are produced in moderate to low signal-to-noise cases. For these, the two tasks `UVLSF` and `UVLIN` should be used. They do the fits in the real and imaginary parts of the visibilities and, hence, do not produce biased estimates in the absence of signal. Both tasks allow you to shift the visibilities to move a strong source to the center of the field before doing the fits (and then shift the visibilities back to the original phase center). This changes sinusoidal real and imaginary parts to linear for an accurate fit, but only for those cases where a single source dominates the field. Both tasks offer the option to flag discrepant data, but they differ in the details of how they do this. `UVLSF` offers the option to write out a "best-fit" single-channel data set as well as subtracting it from the line data; see § 8.1 for an sample inputs to `UVLSF`.

There are also a number of ways to remove the continuum in the image domain, all of which assume that the channels have been imaged similarly and placed into a three- or more dimensional "cube." The "classical" method to subtract the continuum, which is the least useful, is to average those image planes which contain no line signals using SQASH on each set of line-free planes and COMB to average those averages. Finally, COMB is used to subtract the resulting plane from each plane of the initial cube. In this method, the cube is in the "natural" transposition with the frequency axis third. The other two methods require you to use the task TRANS (see § 8.5.2) to transpose the frequency axis to the first (most-rapidly varying) axis. Then IMLIN may be used to fit polynomial baselines (linear is usually all that is justified) to the line-free parts of each spectrum. This task, like all of the other tasks so far mentioned in this section, requires you to use a fixed set of channels for all positions in the field. Many objects (*e.g.*, rotating galaxies) have spatially-dependent spectra in which each pixel has a rather narrow line width compared to the object or objects as a whole. In such cases, the task XBASL may be used. This task requires some endurance on your part to complete, but it allows you to specify the baseline region for each pixel interactively using the AIPS TV or Tek graphics windows.

8.4 Imaging

There are several tasks which may be used to convolve spectral-line data to a rectangular grid and then Fourier transform and Clean that grid to make an image. The old tasks, which we no longer recommend, are called UVMAP, and APCLN and may be used if you insist. Today, IMAGR should be used for all normal imaging and Cleaning. Read § 5.2 for basic information about this task and § 5.3 for information about using it to deconvolve your images. It can read either single- or multi-source data, can apply calibration, and writes all channels into an image cube. IMAGR uses much more flexible and correct methods for data weighting (§ 5.2.3) and uses the superior multi-field method of Cleaning in which source components are subtracted from the visibility data and the images recomputed in every "major cycle." IMAGR grids each channel in exactly the proper place in the *uv* plane and can be used to do frequency smoothing ("averaging") in this most correct of ways. It also offers options to correct your data for various effects of importance over wide fields and bandwidths. It even has experimental variations on Clean (§ 5.3.4) to deal with extended sources and the Clean bias.

To produce a cube of spectral line images with IMAGR from channels *n1* through *n2* use:

> TASK 'IMAGR' ; INP C_R	to review the inputs needed.
> IND m ; GETN n C_R	to specify the input *uv* file.
> CLRONAME C_R	to use the default output names and disk.
> BCHAN *n1* ; ECHAN *n2* C_R	to include a range of channels.
> STOKES 'I' C_R	to make total intensity (unpolarized) images.
> NCHAV 1 C_R	to avoid averaging channels.
> CHINC 1 C_R	to do every channel, setting the channel increment.
> CHANNEL 0 C_R	do not restart.
> CELLSIZ Δx , Δy C_R	to set the image cell dimensions in arc seconds. Cells do not have to be square.
> IMSIZE N_x , N_y C_R	to set the image size in pixels (must be powers of two).
> NITER 0 C_R	to do no Cleaning.
> GO C_R	to run IMAGR.

This makes "dirty" images and beams of the specified channels. If Cleaning is needed, set the Clean adverbs NITER, NBOXES, CLBOX, GAIN, BOXFILE, etc. Note that you must image one IF at a time in order to avoid combining multiple frequencies. The task NOIFS might be useful depending on your data.

Because of the upgrade of the VLA in 2010, IMAGR is more careful with image units. Unless you force the Clean beam size with BMAJ, BMIN and BPA, IMAGR will image each channel at its "natural" (frequency-dependent) resolution. It now carefully scales each image plane so that it is actually in units of Jy per the Clean beam listed in the header. The Clean beam for each frequency is recorded in a CG table and used by MCUBE and CONVL among other tasks.

At this point, it is a very good idea to determine the noise in your output images and to compare it to the theoretical noise you expect. If your images are significantly noisier than expected, it is a very good idea to stop processing, to think about what may have gone wrong, and then to check and correct that. The noise may be determined using task IMEAN:

> TASK 'IMEAN' ; INP C_R	to review the inputs needed.
> IND *m* ; GETN *n* C_R	to specify the input image file.
> DOHIST TRUE C_R	to plot the histogram.
> NBOXES 200 C_R	to use a significant number of boxes in the plot.
> PIXRANGE = -*x* , *x* C_R	to limit the histogram to the range x, where *x* should be about 5 times the expected noise.
> DOTV TRUE C_R	to put the plot on the TV rather than in a file.
> BLC *x1, y1, f1* ; TRC *x2, y2, f2* C_R	to select a sub-image of the cube that is free of signal.
> GO C_R	to run the task, plotting on the TV.

This will print two rms's on your message screen one computed using all the data and one done by fitting a histogram to the noise portion of the signal. Especially if the latter fails, the plot will allow you to estimate the true rms, ignoring those pixels significantly above and below the Gaussian noise part of the histogram. Using this plot, it may be better to include all of the data (BLC 0 ; TRC 0 C_R) rather than to limit the number of pixels contributing to the histogram. The AIPS verb IMSTAT does the all-pixel rms computation without the useful plot, while the verb TVSTAT allows you to mark one or more non-rectangular regions on an image on the TV over which the rms is computed. Task RSPEC will plot robust rmses computed on a per plane basis.

8.5 Display and manipulation of data cubes

8.5.1 Building and dismantling data cubes

Many spectral-line display and analysis functions make sense only on 3-dimensional (or more) images. In particular, image creation functions may create one image per spectral channel while spectral analysis requires the images to be ordered with the spectral axis first (most rapidly varying). To combine a number of *n*-dimensional images into an *n*- or $n+1$-dimensional image, use the task MCUBE. This task requires the images to have some one physical axis whose value varies between the input images in a manner consistent with a regular axis and it is that axis which is extended to make the "cube." If the output image specified already exists, then the input maps are inserted in the output image in the appropriate places. This allows the *n*- or $n+1$-dimensional output image to be built up a bit at a time and allows replacement of portions of the image with a corrected *n*-dimensional image. Type EXPLAIN MCUBE C_R to receive a variety of hints and suggestions for using this rather general program. As an example, to put 31 frequency channel images into one cube, type:

> TASK 'MCUBE' ; INP C_R	to review the inputs.
> INNA 'N315' C_R	to select the source.
> INCL 'IIM001' C_R	to select the input image class for *all* images.
> INSEQ 1 ; IN2SE 31 ; IN3SE 1 C_R	to set the first and last image and the step in the task's loop over input sequence number. The task handles any missing images gently.

> AXREF 1 ; AX2REF 31 C_R	to set the pixel coordinate of the first and last image on the third axis in the cube.
> NPOINTS 31 C_R	to set the total number of points on the third axis.
> OUTN ' ' C_R	to use the default OUTNAME.
> OUTCL 'LMFCUB' C_R	to specify, in OUTCLASS, the order of axes.
> GO C_R	to run MCUBE.

MCUBE will scale each image plane so that it is on the same brightness scale as represented by the Clean beam in the header.

Later on, you might want to replace some of these images by Cleaned images. *E.g.*, assume that you have Cleaned channels 10 through 20, one at a time. These images got the class ICL001. It is a good idea to give them sequence numbers that are the same as the channel numbers, thus 10 to 20. Putting them in the existing cube (make a backup copy with SUBIM if you are nervous) can then be done as follows:

> TASK 'MCUBE' ; INP C_R	to review the inputs.
> INNA 'N315' ; INCLA 'ICL001' C_R	to select the clean images.
> INSE 10 ; IN2SE 20 ; IN3SE 1 C_R	to set the first and last image and the step in the loop.
> OUTN 'N315' ; OUTCL 'LMFCUB' C_R	to select the existing cube.
> OUTSE 1 C_R	
> GO C_R	to run MCUBE.

IMAGR with its DOTV option allows you to Clean each channel with its own numbers of iterations and its own interactively set Cleaning boxes. There are new options to create boxes automatically on each image plane and an option to control whether the boxes for channel n are carried over to channel $n + 1$.

MCUBE can build a cube out of a set of images that do not have a suitable axis for image building. This can be a set of images that have all the same axis parameters (*e.g.*, identical images except for date) or they can be images at, for example, an arbitrary set of frequencies. Set DOCONCAT = 2 to force MCUBE to make a SEQ.NUM. axis or, if there are different frequencies, an FQID axis with accompanying FQ table. The original coordinate values on an axis that differs from image to image are recorded in the history file and, for frequencies, also in the FQ table. Task FQUBE does this operation more naturally and can combine two cubes each of which already has an FQID axis.

Most programs work on cubes. However, you may find it convenient, on occasion, to work with single image planes. To separate the channels from the cube, use task SUBIM typing, for example:

> TASK 'SUBIM' ; INP C_R	to review the inputs.
> INNA 'N315' ; INCL 'LMFCUB' ; INSE 1 C_R	to select the cube.
> OUTN 'N315' ; OUTCL 'IMAGE' C_R	to give it an output name and class.
> BLC 0 ; TRC 0 C_R	to select full planes.
> DOWAIT TRUE C_R	to run in synchronous mode.

> FOR J=10:20 ; BLC(3)=J ; TRC(3)=J ; PRIN J ; OUTSE J ; GO ; END

This runs the program 11 times, taking planes 10 through 20 and creating separate images for them. To put the images back into the cube after they have been modified, use MCUBE.

8.5.2 Transposing the cube

The task TRANS will transpose the cube, ordering the axes in any way you specify. Typically, one transposes images of spectral channels into spectra at image pixels for both display and analysis purposes. Thus:

> TASK 'TRANS' ; INP C_R	to review the inputs.
> INCLASS 'LMFCUB' C_R	to select the untransposed cube.
> TRANSC '-312' C_R	to make new axis order -3, 1, 2 in terms of the old axis order (*e.g.*, RA, Dec, Freq becomes Vel (opposite sign of Freq), RA, Dec).
> OUTCL 'VLMCUB' C_R	to give it an outclass reflecting the axis order.
> BLC 0 ; TRC 0 C_R	to transpose the whole cube.
> GO C_R	to run the program.

Numerous tasks such as IMLIN, XSUM, PLCUB, and XMOM act on the first axis and, typically, make sense only when that axis is frequency/velocity. Note that such *x*-axis analysis may also be useful in other cases, including angular and time *x* coordinates.

8.5.3 Modifying the image header

On occasion you may feel the need to modify or add to the information in the image header. For example, to add an alternate velocity description for the frequency axis of a cube, type:

> INDISK *n* ; GETN *ctn* C_R	to select the image.
> AXTYPE 'OPTHEL' C_R	to specify optical-convention velocities relative to the Sun.
> AXREF 16 C_R	to specify the velocity reference pixel (channel 16 is the center of the band in our example).
> AXVAL 5.E6 C_R	the velocity at the reference pixel in m/s.
> RESTF 1420.4E6, 5752 C_R	to specify the line rest frequency in Hz (1420405752 Hz).
> ALTDEF C_R	to add the information to the header.
> ALTSW C_R	to switch between frequency and velocity information in axis labeling, IMHEADER, etc.

The VLA now provides the velocity information to FILLM, making this operation unnecessary in many cases.

Observers may find the Galactic coordinates of their sources to be of interest. To switch the header between Celestial and Galactic coordinates, type:

> CELGAL C_R	to go to galactic coordinates.
> CELGAL C_R	to go back to celestial coordinates.

You may change many of the parameters in the image header and may fetch for use by the *POPS* language all header parameters. The verb PUTHEAD changes a header parameter and GETHEAD gets that parameter. Type EXPLAIN GETHEAD C_R for the details, including a list of recognized keywords, and EXPLAIN PUTHEAD C_R for the more restricted list of values which you may change in the header. For example, to move a header parameter from one image to another:

> KEYWORD '*keyword*' C_R	to specify one of the fully allowed keywords.
> GETN *n* ; GETHEAD C_R	to get the header value from image *n*.
> GETN *m* ; PUTHEAD C_R	to put it in the header of image *m*.

8.5.4 Displaying the cube

The easiest way to look at the cube is by using TVMOVIE. This verb loads portions of planes of a cube into the TV memory, and displays them in sequence at a variable frame rate. Type:

> INDISK *n* ; GETN *ctn* CR	to select the cube.
> INP TVMOVIE CR	to review the inputs; use defaults for a start.
> TVMOVIE CR	to load the images and start the movie.

Now follow the instructions on your screen on how to change the transfer function, change speed, look at single frames, etc. Having terminated the movie (button D), you may restart it with

> REMOVIE CR	to resume the movie.

With the default adverbs, TVMOVIE will zoom the display so that a single frame fills the display area. If you are using a workstation, this may seriously limit the maximum possible frame rate. To use a smaller zoom factor and smaller display area, enter:

> DOCENTER TRUE CR	to enable the option.
> FACTOR *m* CR	to use a zoom factor of *m*.
> REMOVIE CR	to rerun the movie.

Note that you may set these adverbs before the initial TVMOVIE command as well. If your cube has more than 3 axes with multiple pixels, TVMOVIE can display all planes if DOALL = 1.

TVMOVIE uses an image ordering on the screen designed to improve the performance of movies on "real" TV devices. To use a more pleasing ordering (left to right, top to bottom) that is nearly as efficient on workstations, use the TVCUBE verb instead. The verb REMOVIE works on these displays as well. One of the reasons to use TVCUBE is to capture a full TV plane's worth of channels and print them on a PostScript printer. The first step is to get TVCUBE to put the desired channels in one TV image plane. Set TBLC(3) and TTRC(3) to select those planes you want and set TVCHAN = 2 (or the highest TV plane allowed on your system) to force all the images into one TV image. Stop the move with button D, display the full TV memory (use the F2 button), adjust the enhancement (and color) of the display, and then run task TVCPS. If your images are too large for one channel, use TVCHAN = 1 and run TVCPS once for each channel.

Another way to see multiple image planes on a single page is with the task KNTR, which plots a page of contour and/or grey-scale drawings for each plane. This task allows you to include "star" positions and even an outline of the clean beam. A KNTR plot of neutral hydrogen emission from NGC6503 is shown in Figure 8.1.

It can be very informative to transpose the cube to look at position-velocity planes instead of position-position planes. A TVCUBE display of the NGC6503 data cube transposed to 132 (position-velocity-position) order and then captured with TVCPS is shown in Figure 8.2.

Finally, it is useful to look at the cube as a set of spectra. Use TRANS to transpose to 312 order (or -312, 321, and the like). The task XPLOT displays the spectra on the graphics display selecting only those with a sufficiently strong line signal (set by adverb FLUX) and ignoring those with really strong fluxes (controlled with PIXVAL). This can get rather tedious and it is easy to get lost in your cube. The task PLCUB, illustrated in Figure 8.3, allows you to see numerous spectra at one time plotted in correct orientation with respect to their position coordinate. It is necessary to experiment with the adverbs for PLCUB, especially YINC, ZINC, PIXRANGE, and the like, to get a good plot. Use the TV with DOTV TRUE CR to save trees.

In general, other display programs work on one plane at a time. Therefore, you must specify which plane in the cube you want to see. For example, if you want to do TVALL on channel 16, which is on pixel 16 of the third axis, you type:

> TBLC 0 0 16 ; TVALL CR

Note that adverbs TBLC and TTRC are used for TV displays while BLC and TRC are used for image operations.

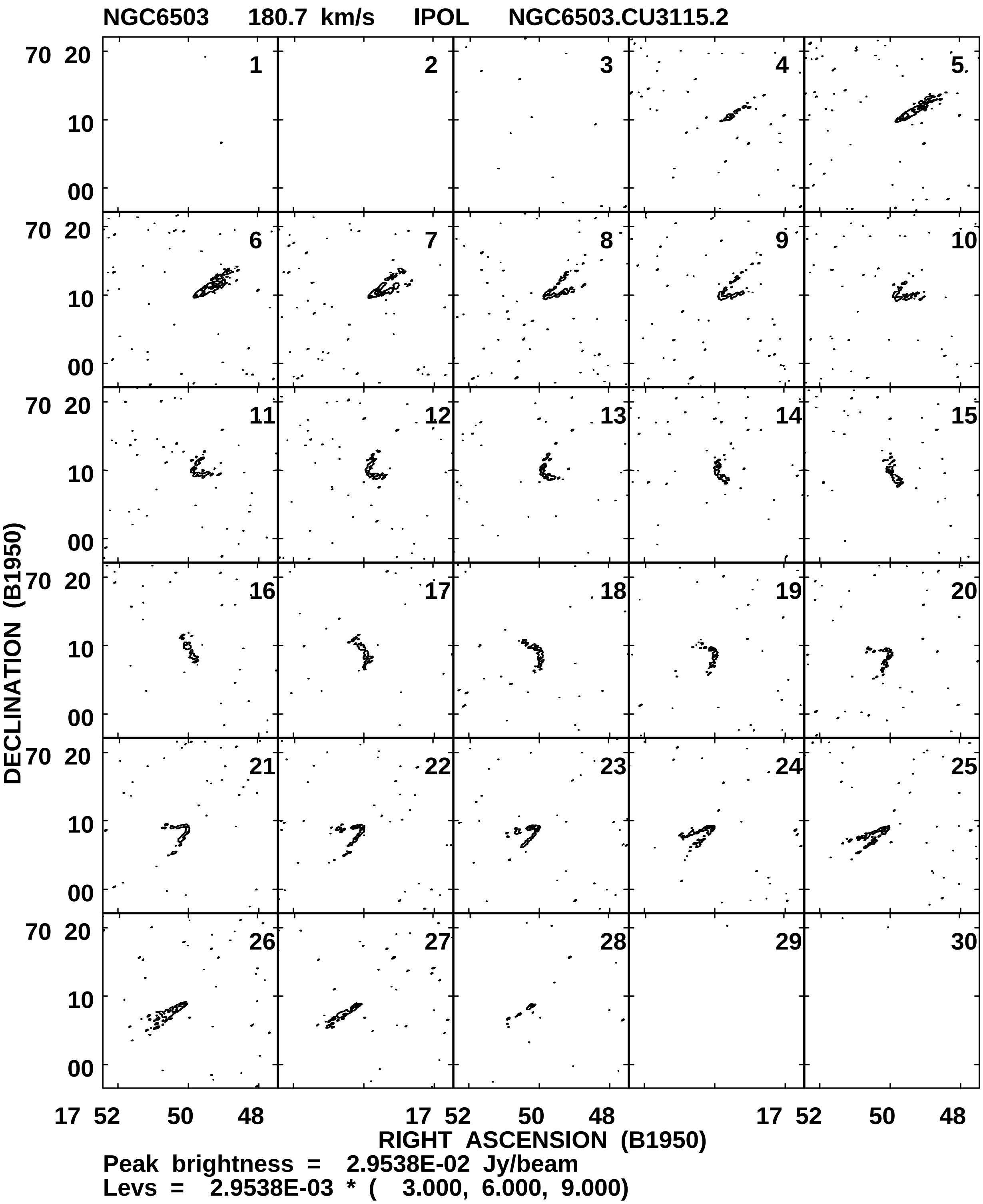

Figure 8.1: Contour images of spectral channels from a cube generated by 𝒜ℐ𝒫𝒮 task KNTR. Data on NGC6503 were provided by Don Wells from observations made with the VLA on 12 April 1983.

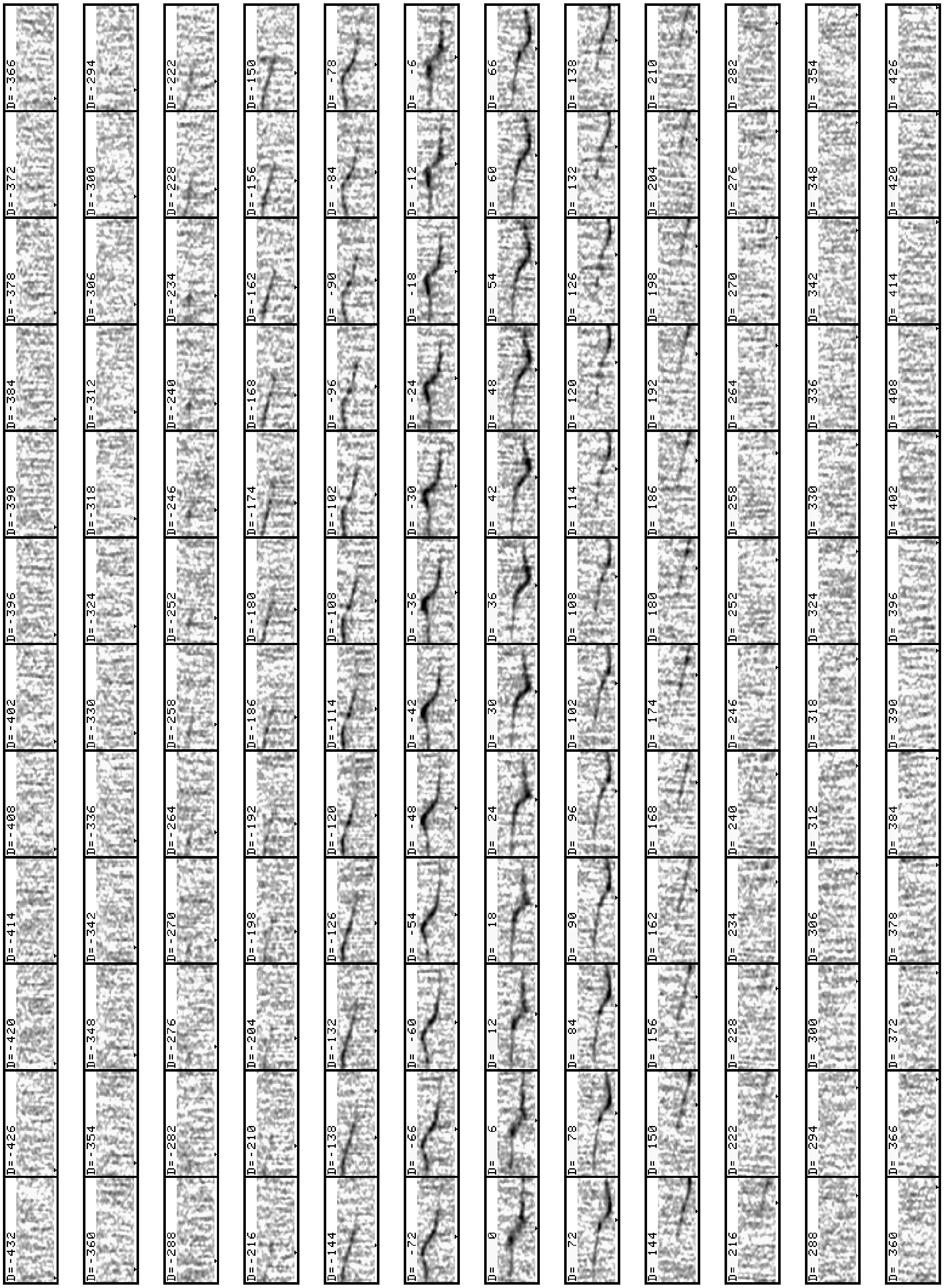

Figure 8.2: Display generated by transposing a data cube into 132 (right ascension, velocity, declination) order and then displaying the central declination position-velocity planes with AIPS verb TVCUBE. The TV display was then captured by *AIPS* task TVCPS for printing on PostScript printers. Data on NGC6503 were provided by Don Wells from observations made with the VLA on 12 April 1983.

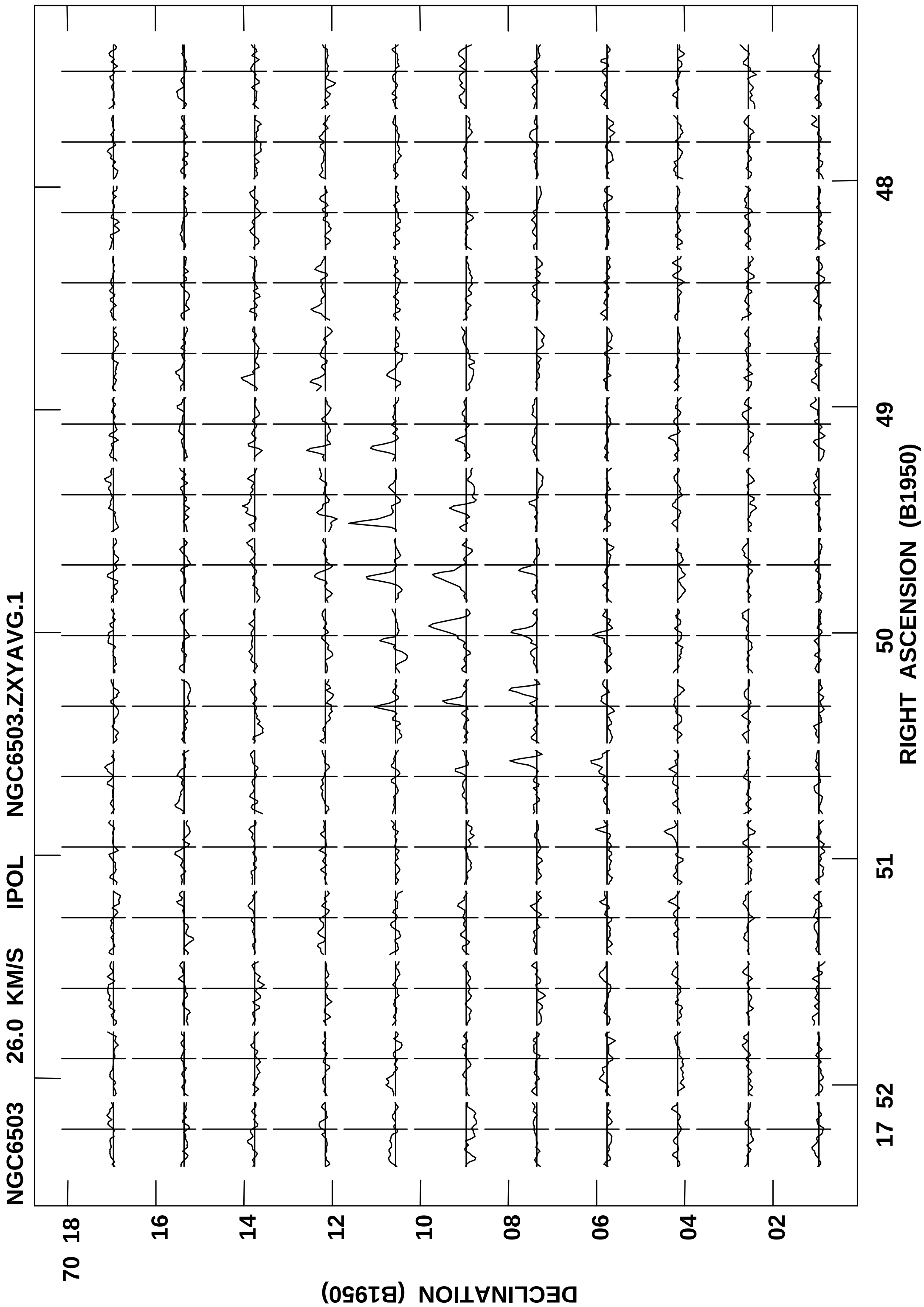

Figure 8.3: Display generated by transposing a data cube into 312 order (velocity, right ascension, declination) and then displaying selected spectra on a regular grid in the angular coordinates with AIPS task PLCUB. Data on NGC6503 were provided by Don Wells from observations made with the VLA on 12 April 1983.

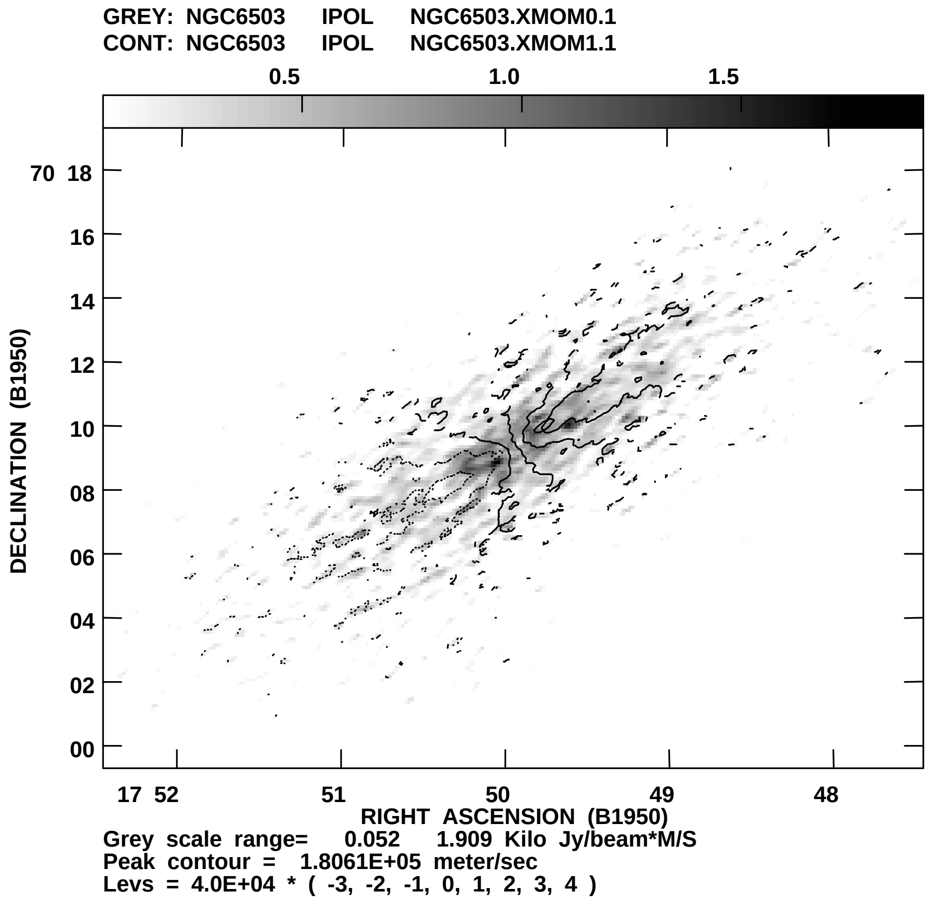

Figure 8.4: Display generated by GREYS and LWPLA using the moment 0 image for grey levels and the first moment image for contours. Data on NGC6503 were provided by Don Wells from observations made with the VLA on 12 April 1983.

8.6 Analysis

A wide variety of programs is available to do further analysis of the data. Exactly which ones you will need depends on the nature of your observations. To see the latest list of symbols related to spectroscopy, enter

`> ABOUT SPECTRAL C_R` to list them all on your terminal.

or consult Chapter 13 of the *CookBook* for a less current, but paper, copy of the list.

The subject of finding and removing the continuum (aka spectral baseline) was discussed in § 8.3 above. In the image domain, task `IMLIN` may be used to remove polynomial baselines using the same spectral channels throughout the cube. The task `XBASL` can be used to remove baselines in an interactive (hence position-dependent) fashion. Be aware, however, that if you have made an error in the calibration, this has most likely caused slopes in amplitude *and* phase. Therefore, it is generally better to track down the error and correct it than to decide (arbitrarily) to take out slopes in (Cleaned ?) image amplitude. Task `SQASH` may also be used to sum or average planes in a cube, which provides, among other things, a simple but crude way to determine a continuum image which can then be subtracted from the cube by `COMB`.

Smoothing and blanking are important for almost all analysis programs. `CONVL` works on cubes and does a spatial smoothing (on position-position-velocity cubes). `CONVL` will use the `CG` table to smooth all image planes to the same spatial resolution. This is preferable to forcing a resolution with BMAJ since it smooths the residuals as well as the components. Using all the defaults in XSMTH performs a Hanning smoothing in velocity (on transposed velocity-position-position cubes) and can be used to do other kinds of smoothing as well. This is not just useful for bringing out weak extended signals. Smoothed images can also assist in determining the boundaries of sources to set windows for subsequent spectral analysis. For example, the smoothed cube could be used to set the CLIP limits in task COMB to be applied to the unsmoothed cube.

In fact, finding all regions of significant line signal may be difficult in large image cubes. The task SERCH offers an algorithm by Juan Uson to find line signals which match specified line widths and exceed a specified signal-to-noise ratio. Histograms may be plotted or printed and a S/N hyper-cube may be written.

The task BLANK offers a variety of algorithms for "blanking" out regions of bad data or source-free regions in spectral-line cubes. It has an interactive mode, which allows you to indicate with the cursor on the TV what are "good" regions. Set everything but the image name to default values, use OPCODE 'TVCU' C_R and type GO BLANK C_R. Then just follow the instructions, pushing button A to lay out the polygon and button D followed by C_R to go to the next image. If marking "bad" regions is easier, set DOINV TRUE C_R before running BLANK. A less subjective way is to use BLANK with OPCODE = IN2C. Blanking is done based on the pixel values in a second cube, usually a smoothed version of the original cube. Task RMSD can be used to blank an image based on its rms in a window surrounding each pixel. It can also be used to compute images of rms using robust, histogram, or median-absolute-deviation ("MAD") methods.

The blanked cubes can be used to calculate integral profiles with BLSUM and to calculate moments 0 to 3 of the profiles with XMOM. Thus, the 0-moment image will be the integral under the profile (*e.g.*, total HI), the first moment is the velocity field, etc. OPTYPE='MAX' in XMOM will select the peak brightness on each row and make images of it and its coordinate. A comparison of the two types of output can be very illuminating. Task MOMNT does the smoothing, blanking and calculating of moments all in one run. This is very easy to use, but can be dangerous since you don't see what is going on. A display of the 0^{th} and 1^{st} moment images computed by XMOM is shown in Figure 8.4.

Moment images (or any other 2 or 3 images) may be combined in a display in which one image controls intensity, a second image controls hue (color), and a third optional image controls saturation (color richness or purity, done by varying the ratio of white to pure color). In the days of powerful television display devices, this display could be done at truly interactive speeds in the display hardware. Today, most workstations are capable of displaying full color images and the *AIPS* display program supports them. The verb TVHUEINT will display the two specified TV channels using one as the intensity and the other as hue; see § 6.4.7 *AIPS* task HUINT may also be used with either full-color TV displays. This task offers a small menu of interactive options to enhance the images and otherwise control the display in manners essentially the same as the verb. It however offers the option to write out an image cube having RGB as its third axis. It can also write an image of the 3-color step wedge. These images may then be processed with the usual display tasks such as KNTR and rendered in PostScript by LWPLA. *AIPS* task TVHUI may also be used with either full or pseudo color TV displays. This task also offers a small menu of interactive options to enhance the images and otherwise control the display. It also offers the option to write out an image cube having RGB as its third axis and using much greater mathematical accuracy than is allowed in the TV display.

RGBMP computes "integral" images another way — as three weighted sums representing the low, center, and high velocity parts of the cube. Like TVHUI, RGBMP writes its results as a cube with RGB as the third axis. Task TVRGB can display these outputs (or any other RGB cube or any three image planes), using one image plane to control the red image, one to control green, and one to control blue. It works on real TV displays and full-color workstations and, using an algorithm to minimize the loss of color information, on pseudo-color workstations. Like TVHUI, it offers a small menu of interactive enhancement options. TVRGB can write 24-bit color PostScript files beginning with the 15JAN96 release. TVCPS is another way to capture the displays generated by TVHUI and TVRGB to send to color PostScript printers.

If you prefer to fit Gaussians instead of calculating moments, the program XGAUS can be used. First, it is a good idea to become acquainted with your data cube. A highly interactive task called TVSPC can let you examine the spectra from one or two transposed cubes at positions selected from an image plane, all on the TV display. The image plane should be some image representing the full field, such as moment-zero or, in polarization, the I polarization image. This task can also display another, non-transposed cube with the displayed plane selected from the displayed spectra. In earlier releases, use XPLOT first to look at (a sample of) the profiles, before you do any Gaussian fitting. XGAUSS offers a non-interactive mode, but it is frequently unstable and depends on the fit channels and the initial guesses being virtually independent of position. Therefore, in most cases, it is preferable to use the interactive mode, so that you can see what is happening, but be aware that it might be rather time-consuming. XGAUS prepares a table of solutions which can be edited in various ways after all pixels have been fit once. Images of the fit parameters are examined during the editing process and are written out only after you are happy with the results. The task may be re-started with the same table as often as needed.

RMFIT is used to fit rotation measures to Q and U spectral cubes, using the cube of Faraday rotation output by FARS to provide initial guesses. It is far more capable than the Faraday rotation synthesis in distinguishing multiple components in the rotation measure. ZEMAN fits Zeeman splitting to spectral cubes of I and V polarization. It can fit the line directly or it can fit the individual Gaussians found by XGAUS individually. The latter allows multiple magnetic-field values to be found in the same direction. See *AIPS* Memo 118[2] for a detailed description of these tasks.

Two more similar tasks have appeared to handle the different mathematics required by absorption-line image cubes. When optical depths become significant, the noise in optical depth becomes a strong function of the optical depth, so the fitting is done in the observed absorption spectra rather than in optical depth even though it is the optical depth which is treated as Gaussian. These tasks, AGAUS and ZAMAN function very much like the tasks for emission-line cubes except that initial guesses are set on a plot of the optical depth spectrum in AGAUS and for details of the mathematics. See *AIPS* Memo 122[3].

The cube can be rotated with OGEOM (if the $\alpha - \delta$ pixels are square), *e.g.*, to align one of the axes with the major axis of a galaxy. If the pixels are not square, use OHGEO to re-grid the image instead. A single profile can be produced from these images with SLICE, then plotted using TKSLICE,TVSLICE, or SL2PL (see § 6.5.2, § 6.4.8, § 6.3.2.2, and Figure 6.5). PLCUB, PLROW and XPLOT are convenient programs for displaying multiple individual spectral profiles after the cubes have been transposed.

To compute and display spectral profiles summed over regions in the two angular coordinates, use ISPEC for circular and/or rectangular regions and BLSUM for irregular regions set interactively using the TV display and cursor. Both tasks print their results; both BLSUM and ISPEC make standard *AIPS* plot files, and BLSUM can also make a printer plot. Note that both tasks sum over areas in the first two axes and plot that as a function of position on the third axis, no matter what the three axes actually represent. This suggests a variety of interesting possibilities, such as time functions or line integrals versus position in the source. Both tasks can write the results as slice extension files which may then be printed, plotted and Gaussian fit in a variety of ways. ISPEC can plot the spectral derivative of the profile it has computed, which is of interest in Zeeman splitting experiments. Task RSPEC is similar to ISPEC except that it plots the rms rather than the data. RSPEC has options to write out a signal-to-noise image and/or a text file of channel weights as well.

Note that if all you require is a single spectral profile, it may be possible to use POSSM which works on the *uv* data directly. If the spectral profile is a function of position within a well-resolved source, then you will have to go to the image plane.

GAL fits models of galaxy rotation to images of the predominant velocity (*e.g.*, the first moment images written by XMOM, XGAUS, or MOMNT). It accepts a second input image to be used as weights for the first input

[2]Greisen, E. W. 2020, "Modeling Spectral Cubes in *AIPS*," AIPS Memo 118 revised, http://www.aips.nrao.edu/aipsdoc.html
[3]Greisen, E. W. 2020, "Modeling Absorption-line Cubes in *AIPS*," AIPS Memo 122, http://www.aips.nrao.edu/aipsdoc.html

image; it is common practice to use a the zero'th moment image for this. Plots may be produced. The task MODVF creates a model velocity field based on a user provided model for the rotation curve, orientation, and warping of the plane.

A much more powerful, but tricky to use, task named CUBIT was developed by Judith Irwin and contributed to AIPS [4]. This task fits a galaxy rotation and emitter (usually HI) distribution to the full data cube. It uses dynamic memory and so can work on any reasonable size cube. It is much more likely to work if the cube has been very carefully BLANKed and if you approach it with patience. Make the best initial guesses for all the parameters that you can. Then fit for one parameter at a time until you begin to get a reasonable result. Then allow the task to fit all of the parameters at once. And finally, check the fits going through the galaxy in halves or quadrants. But first, study EXPLAIN CUBIT C_R in detail. The results from this task are worth the trouble to get it to work on your data.

8.7 Additional recipes

8.7.1 Hot banana soufflé

1. Preheat oven to 375° F.
2. Select a 6-cup soufflé dish or other mold and grease it liberally with 1 tablespoon **butter**.
3. Place 6 **eggs**, 1/2 cup **cream**, juice of 1/2 **lemon**, 1 tablespoon **kirsch**, and 1/4 cup **sugar** in blender. Blend until the batter is smooth.
4. Peel 2 large **bananas**, removing any fibers and break into chunks. With blender running, add the chunks one at a time.
5. Break 11 ounces **cream cheese** into chunks and add them to the blender.
6. When all the ingredients are thoroughly mixed, run the blender at high speed for a few seconds.
7. Pour batter into prepared dish and place it in the hot oven. Bake 45–50 minutes until the top is lightly browned and puffy. You may quit when the center is still a bit soft or continue baking until the center is firm.
8. Serve at once. A whipped cream flavored with Grand Marnier makes a nice topping.

8.7.2 Curried bananas

1. Melt 2 tablespoons **butter** in saucepan and cook 2 tablespoons minced **onion** in it for 2–3 minutes.
2. Mix 1 tablespoon **curry powder**, 1 teaspoon **salt**, 1/4 cup **flour**, and a dash of **cayenne pepper** with a little **milk** to make a paste.
3. Add paste to onion, cooking gently for 10 minutes. Add balance of 2 cups **milk** slowly, stirring until it boils.
4. Slice 7 small green **bananas**, and cook gently in the sauce until tender.
5. Serve as a vegetable in a ring of hot cooked rice.

From *Everyday BANANA Recipes*, Banana Distributing Co., New Orleans, published by Bauerlein, Inc. New Orleans, 1927.

[4] Irwin, Judith A. 1994, "Arcs and bridges in the interacting Galaxies NGC 5775/NGC 5774," ApJ, 429, 618-633.

8.7.3 Banana crunch cake

1. Heat oven to 350° F. Grease and flour 10-inch tube (Bundt) pan.
2. In medium bowl, combine 1/2 cup **flour**, 1 cup **cocnut**, 1 cup **rolled oats**, 3/4 cup firmly packed **brown sugar**, and 1/2 cup chopped `pecans`. Mix well.
3. Using fork or pastry blender, cut in 1/2 cup **margarine** until mixture is crumbly. Set aside.
4. In a large bowl, combine 1 1/2 cups sliced very ripe **bananas**, 1/2 cup **sour cream**, and 4 **eggs**; blend until smooth.
5. Add 1 package **yellow cake mix**, Pillsbury Most Supreme is recommended. Beat 2 minutes at high speed.
6. Spread 1/3 of batter in tube pan, sprinkle with 1/3 of coconut mixture. Repeat layers twice more using remaining batter and coconut mixture, ending with coconut mixture.
7. Bake at 350° F for 50 to 60 minutes or until toothpick inserted near center comes out clean. Cool upright in pan 15 minutes; remove from pan. Place on serving plate, coconut side up. Cool completely.
8. HIGH ALTITUDE — above 3500 Feet: Add 3 tablespoons flour to dry cake mix. Bake at 375° F for 45 to 55 minutes.

8.7.4 Panecillos de Plátano

1. Sift together 2 cups **flour**, 1 teaspoon **salt**, and 3 teaspoon **baking powder**.
2. Add 4 tablespoons softened **butter**, mix well, add 3/4 cup **milk**, and stir only until dampened.
3. Roll to 1/2 inch thickness, cut into cookies about 2 inches in diameter, and place on greased cookie sheet.
4. Slice 2 **bananas** in 1/2 inch thicknesses and dip pieces in 2 tablespoons **lemon juice** and then in 2 tablespoons **sugar**. Place a slice on each cookie, pressing it down.
5. Bake in a 425° F oven for 12 minutes or until golden brown.

Thanks to Ruth Mulvey and Luisa Alvarez *Good Food from Mexico*.

8.7.5 Banana-Rhubarb Crisp

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

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