

A manual of Difmap help pages

Martin Shepherd (mcs@astro.caltech.edu)
Copyright ©1993 California Institute of Technology

December 6, 2021

The list of commands in difmap.

addcmp	34
Add a model component by hand to the tentative model.	
addhist	36
Append a new line to the recorded history of an observation.	
addwin	37
Append a new entry to the list of clean windows.	
antenna_beam	38
Specify a beam for one or more antennas (for primary beam calculations).	
basename	41
Returns the name of baseline 'ibase' of sub-array 'isub'.	
beamloc	42
Change when and where mapplot plots the clean beam ellipse.	
check_uvaver	44
Check if a given averaging time would cause time smearing.	
clean	48
Clean a residual dirty map with the dirty beam.	
clear_markers	51
Delete the current list of map markers.	
clear_models	52
Discard the current set of channel-range/polarization specific models.	
clrhist	53
Clear the history information recorded with the observation.	
clrmod	54
Clear one or more of the established, tentative and continuum models.	
clroff	56
Undo all resoff calibration-offsets applied during this difmap session.	
corplot	57
Display accumulated telescope based self-cal corrections versus UT.	
cpplot	59
Interactive display and editing of observed and model closure phases.	
delmarker	67
Delete the closest map marker to a given Right Ascension and Declination.	
delwin	68
Delete all clean windows.	
edmodel	69
Modify the established and tentative models with an external editor.	
fix_weights	72
Set whether amplitude corrections affect the visibility weights.	
flag	74
Flag a set of baselines over a selectable time range.	
get	76
Restore UV data, models and windows from files.	
gscale	78
Self calibrate to determine overall telescope amplitude corrections.	
header	81
List useful parts of the observation header on one's terminal.	
imstat	82
Return image plane map statistics.	

	2
invert	84
Grid selected UV data and Fourier invert it to a (residual) dirty map	
keep	87
Establish the latest CLEAN model and compute its UV representation.	
loglevs	88
Set logarithmic contour levels for mapplot.	
map_to_rad	90
Convert an image plane distance from map units to radians.	
mapcolor	91
Select the colormap type, brightness and contrast to use in mapplot.	
mapfunc	94
Change the transfer function and data range displayed in mapplot.	
maplot	96
Alternative name for the mapplot command - see help mapplot.	
mapplot	97
Display the dirty beam, residual or restored map and modify CLEAN windows.	
mapsize	106
Request a new map grid of given width and cell size	
mapunits	110
Change the units used to specify and label map and UV plane distances.	
mapvalue	112
Return the value of the map pixel nearest to a given coordinate.	
mark_radec	113
Add an optionally annotated map marker to be drawn at a given RA, Dec.	
mark_xy	116
Add an optionally annotated map marker to be drawn at a given X,Y offset.	
mjd	119
Return the Modified Julian Date corresponding to a given calendar date.	
model_flux	120
Return the flux in either the normal model or the continuum model.	
modelfit	121
Fit image-plane model components to visibilities in the UV plane.	
multi_model	125
Specify whether to keep one model per channel-range/polarization selection.	
nbase	127
Return the number of baselines in a given sub-array.	
nchan	128
Return the count of the number of spectral-line channels per IF.	
nif	129
Return the number of IFs in the current observation.	
nsub	130
Return the number of sub-arrays in the current observation.	
ntel	131
Return the number of telescopes in a given sub-array.	
observe	132
Read UV data from a random-groups UV FITS file.	
peak	135
Return details of the min,max or absolute max flux in a map.	
peakwin	137
Place a CLEAN window around the brightest pixel in the residual map.	

	3
planet_geometry	138
Return the geometry of a given planet.	
planet_temp	140
Find the brightness temperature implied by the measured flux of a planet.	
pointing_center	143
Specify the original pointing center of the observation.	
polvec	144
Configure how polarization vectors are displayed.	
primary_beam	146
Specify a primary beam to use on all baselines.	
projplot	148
Plot visibility amplitudes and/or phases versus projected UV distance.	
rad_to_map	154
Convert an image plane distance from radians to map units.	
radplot	155
Plot visibility amplitudes and/or phases versus UV radius.	
read_models	161
Read a table of models, indexed by channel-range/polarization, from a file.	
resoff	162
Determine and apply a baseline based amplitude and phase correction.	
restore	164
Add the CLEAN model to the residual map to form a clean map.	
rmodel	167
Read a new CLEAN model from a VLBI model file.	
rwins	168
Read windows previously saved CLEAN windows from a text file.	
save	169
Save UV data, models, windows, the restored map and a command file.	
scangap	172
Change the time gap used to delimit neighboring scans.	
select	174
Select a polarization and ranges of spectral-line channels to be processed.	
selfant	179
Set antenna based constraints for subsequent self-calibration.	
selfcal	182
Self calibrate for telescope phase (and amplitude) corrections.	
selfflag	186
Control the identification and fate of un-correctable visibilities.	
selflims	189
Set limits on acceptable amplitude and phase corrections in selfcal.	
selftaper	191
Sets a (1 - gaussian) taper to weight down SHORT baselines.	
setcont	193
Add the current model to the continuum model, or vice versa.	
shift	196
Shift the observation phase-center, CLEAN window and model positions.	
shiftto	198
Move the phase-center, CLEAN window and model positions to a given RA,Dec.	
showhist	199
Display the history that was recorded with the current observation.	

	4
showmodel.....	200
Report detailed information about the current models.	
showpar.....	203
Display a list of difmap configuration parameters.	
specbase.....	204
Select baselines to be displayed by specplot.	
specopt.....	207
Preset specplot display options.	
specorder.....	209
Set the order in which specplot plots spectra.	
specplot.....	212
Plot time-averaged visibility spectra.	
specpol.....	218
Select the list of polarizations to be displayed by specplot.	
specsmooth.....	220
Preset the spectral resolution displayed by specplot.	
spectime.....	223
Select the range(s) of times to be averaged by the specplot command.	
specuvr.....	225
Select the UV radius ranges to be displayed by specplot.	
startmod.....	227
Read a starting model from disk and phase self-calibrate against it.	
telname.....	229
Returns the name of telescope 'itel' of sub-array 'isub'.	
tplot.....	230
Plot time sampling for each telescope of an observation.	
uncalib.....	234
Undo recorded telescope calibrations (eg. self-cal corrections).	
unflag.....	236
Unflag a set of baselines over a selectable time range.	
unshift.....	238
Undo accumulated position shifts from the data, windows and models.	
uv_to_wav.....	239
Convert a UV plane distance from UV units to wavelengths.	
uvaver.....	240
Time average a UV data set.	
uvplot.....	243
Plot observed UV points on the UV plane.	
uvrange.....	247
Set the min and max UV radii of usable visibilities.	
uvstat.....	249
Return UV plane visibility statistics.	
uvtaper.....	251
Set a gaussian taper to weight down long baselines and reduce sidelobes	
uvweight.....	253
Set the type and degree of weighting for subsequent UV gridding.	
uvzero.....	255
Set the estimated flux expected from a baseline of zero length.	
vis_stats.....	257
Compute the statistics of a given observable.	

	5
vplot	259
General visibility plotting and editing command.	
wav_to_uv	267
Convert a UV plane distance from wavelengths to UV units.	
wbeam	268
Save the dirty beam to a FITS file.	
wdmap	269
Save the residual dirty map to a FITS file.	
winmod	270
Delete all components outside (or inside) the current clean windows.	
wmap	271
Save the clean (restored) map to a FITS file.	
wmarkers	272
List the commands that would create the current list of map markers.	
wmodel	273
Write CLEAN models to a VLBI model file.	
wobs	274
Save UV data to a random-groups UV FITS file.	
write_models	275
Write a table of models, indexed by channel-range/polarization, to a file.	
wtscale	276
Change the scale factor that multiplies the raw visibility weights.	
wwins	278
Save CLEAN windows to a text file.	
xyrange	279
Set the bounds of the sub-image to be displayed in mapplot.	

The list of difmap help topics.

antenna_names	7
A description of the format of telescope specifications.	
editing	11
General discussion of editing in difmap.	
models	14
An overview of how models are used in difmap.	
multi_if	19
An overview of multi-IF mapping.	
polarization	22
Overview of Difmap polarization mapping.	
spectral_line	25
An overview of the current spectral-line facilities.	
subarrays	29
A discussion of sub-array handling in Difmap.	
whatsnew	N/A
The details of recent difmap changes.	

Difmap help topics

The following pages show the contents of each of the Difmap help topics. The same pages can be seen from within Difmap by typing:

`help name_of_topic`

antenna_names

A description of the format of telescope specifications.

Many Difmap commands expect you to select one or more sub-arrays, antennas, baselines, or closure triangles. To make this easier, a consistent syntax is used to specify all such telescope aggregates. This specification of an aggregate of $n \geq 0$ telescopes is of the form:

```
[sub-array:][antenna_1 antenna_n...]
```

Where "sub-array" is a number between 1 and nsub, and antenna_n is an exactly matching or an otherwise unambiguous prefix of an antenna name.

[Sub-array]

The sub-array index is optional and defaults to a context dependent value. From the command line it generally defaults to sub-array 1. Within interactive commands, such as vplot, it defaults to whatever sub-array is currently being viewed.

If both a sub-array index and one or more antenna names are specified, the first antenna name must be separated from the sub-array index with a colon (:). Otherwise the colon is optional. A colon without a preceding sub-array index implies selection of the default sub-array.

[antenna_1 ...antenna_n]

If a telescope aggregate is made up of n telescopes then there may be up to n antenna names specified. Omitted trailing antenna names are treated as wild-cards.

Omitting trailing components of a specification

For non-interactive commands, omitting one or more trailing components from a specification is interpreted to mean that all telescope aggregates of the expected type (eg. a sub-array or baseline aggregate...) that match in the initial specified components, are acceptable. The omission is thus effectively a wild-card. For interactive commands, such as vplot, which won't plot all the wild-carded aggregates without further user direction, the first aggregate that matches in the initial given indexes is sought and the selection is then forgotten. Thus to the vplot command an empty specification string "" means plot all baselines of all sub-arrays when plotting to a non-interactive device, whereas on an interactive device the same specification would mean start plotting from the first baseline in the default sub-array.

Escaping special characters

If the name of an antenna contains a special character that would normally be interpreted as a separator between antenna names, or as a set-operator, then to tell difmap that the character is actually part of the antenna name, it is necessary to precede it with a backslash character. Special characters that need this treatment, are the space character, the plus character, the minus character, and the exclamation mark. Beware that since the command line undergoes backlash interpretation before the telescope specification is passed to the function that interprets telescope names; at the command line, it is necessary to use a pair of backslash characters.

For example, suppose that you have an antenna name like "AN WEST", which has a space in it. If you were to attempt to specify this to the 'selfant' command, by typing:

```
O>selfant AN WEST
```

then difmap would interpret this as a baseline specification containing two antennas, one called "AN" and the other called "WEST". To prevent it from doing this, you would need to type the following:

```
O>selfant AN\\ WEST
```

When specifying a telescope name at the prompt of an interactive plotting function, such as radplot, you would only use one backslash, like AN\ WEST, since the entered text doesn't undergo backslash interpretation before being sent to the telescope parsing function.

EXAMPLES

The following examples illustrate how specifications are interpreted in different contexts. Note that while each specification is shown enclosed within quotes, this is to make it possible to display an empty string. In practice quotes are only legal for specifications that are given as command-line arguments to commands, and even there they are rarely necessary. Also note that all difmap commands interpret the omission of a trailing telescope specification argument as equivalent to specifying an empty string.

Sub-array specifications

" "	All sub-arrays.
": "	The default sub-array.
"2"	Sub-array 2.

Telescope specifications

```

""           All telescopes of all sub-arrays.
":"         All telescopes of the default sub-array.
"2"         All telescopes of sub-array 2.
"an1"       Telescope AN1 of the default sub-array.
"2:an1"     Telescope AN1 of sub-array 2.

```

Baseline specifications

```

""           All baselines of all sub-arrays.
":"         All baselines of the default sub-array.
"2"         All baselines of sub-array 2.
"an1"       All baselines of telescope AN1 in the default sub-array.
"2:an1"     All baselines of telescope AN1 in sub-array 2.
"an1 an2"   Baseline an1-an2 in the default sub-array.
"2:an1 an2" Baseline an1-an2 in subarray 2.

```

Closure triangle specifications

```

""           All triangles of all sub-arrays.
":"         All triangles of the default sub-array.
"2"         All triangles of sub-array 2.
"an1"       All triangles of telescope AN1 in the default sub-array.
"2:an1"     All triangles of telescope AN1 in sub-array 2.
"an1 an2"   All triangles of baseline an1-an2 in the
            default sub-array.
"2:an1 an2" All triangles of baseline an1-an2 in subarray 2.
"an1 an2 an3" Triangle an1-an2-an3 in the default sub-array.
"2:an1 an2 an3" Triangle an1-an2-an3 in sub-array 2.

```

SPECIFIC EXAMPLES

1. Inspection of the currently selected self-cal constraints for one or more telescopes.

```

0>selfant ""
1:AN1      self-cal  status=correctable  weight=1
1:AN2      self-cal  status=correctable  weight=1
1:AN3      self-cal  status=correctable  weight=1
2:AN1      self-cal  status=correctable  weight=1
2:AN2      self-cal  status=correctable  weight=1
2:AN3      self-cal  status=correctable  weight=1

0>selfant 2
2:AN1      self-cal  status=correctable  weight=1

```

```
2:AN2      self-cal  status=correctable  weight=1
2:AN3      self-cal  status=correctable  weight=1
```

```
0>selfant an3
1:AN3      self-cal  status=correctable  weight=1
```

2. Non-interactive visibility plotting to a hard-copy device.

```
0>dev /ps
```

```
0>vplot 0, ""
Using default options string "efbm3"
Page 01: baselines of 1:AN1
Page 02: baselines of 1:AN2
Page 03: baselines of 1:AN3
Page 04: baselines of 2:AN1
Page 05: baselines of 2:AN2
Page 06: baselines of 2:AN3
```

```
0>vplot 0, 1
Using default options string "efbm3"
Page 01: baselines of 1:AN1
Page 02: baselines of 1:AN2
Page 03: baselines of 1:AN3
```

```
0>vpl 0, 2:an2
Using default options string "efbm3"
Page 01: baselines of 2:AN2
```

```
0>vpl 1, 2:an2 an3
Using default options string "efbm3"
Page 01: baselines of 2:AN2
```

editing

General discussion of editing in difmap.

EDITING VISIBILITY DATA

It is rare to find an interferometer data set that is not contaminated with interference, correlator errors or other unwanted observational artifacts. So it is important to be able to flag selected parts of the data to be ignored by future processing, and to be able to undo such flags if it becomes apparent that prior editing was unwarranted.

In Difmap there are currently three types of flagging.

Deleted data.

Visibilities that have been given zero weight, and visibilities which do not appear in the input UV FITS file are regarded as "deleted". These visibilities are not displayed in any plots. None of the commands in Difmap change good or flagged data to deleted data, and such data can only come from input UV FITS files. Similarly, there is no way to undelete such data.

Visibility flags.

Individual visibilities can be interactively flagged or unflagged in a number of difmap plot commands, such as vplot, tplot, radplot, projplot and uvplot. When flagged visibilities are displayed, they are plotted in red. When unflagged (good) visibilities are plotted, they are plotted in green.

Visibilities of one or more baselines, can also be flagged over a given time range non-interactively using the 'flag' and 'unflag' commands.

Correction flags.

Telescope based self-calibration corrections consist of three parts. A phase correction, an amplitude correction and a correction flag. Initially the phase correction is set to zero, the amplitude correction to unity and the flag is not set. To fill in amplitude and phase corrections the 'selfcal' command must be used. This determines new incremental telescope corrections based on the input visibilities and a model. Where there is insufficient information to determine a correction, the correction-flag member is set.

When self-calibration corrections are applied, if a correction is flagged, the flag is also applied to all affected visibilities. These correction flags are recorded in addition to other visibility flags

and can be removed without affecting normal visibility flags by running the 'uncalib' command, or by interactively toggling them in the 'corplot' command. In plot commands, visibilities that are flagged due to self-calibration are displayed in blue, to differentiate them from good visibilities plotted in green and otherwise flagged visibilities that are plotted in red. Other commands don't differentiate between the flag types, and thus any visibilities that are flagged for any reason are ignored.

TELESCOPE VERSUS BASELINE EDITING.

Telescope based editing.

Many observational artifacts are telescope based. For instance loss of cooling and other receiver problems are always telescope based, whereas external interference will appear as telescope based problems for VLBI and VLBA work, but as array-wide problems for compact arrays such as the VLA.

The 'vplot' command groups baseline plots by telescope. This gives one the opportunity to determine whether a problem is telescope based, and to home in on the bad telescope. Telescope highlighting in the 'radplot' command can also be used to search out bad telescope based artifacts. When such problems are identified, one should display all baselines of the bad telescope in 'vplot' and select the default telescope-based editing mode before editing.

Baseline editing.

Fortunately, very few observational artifacts appear as baseline based errors. This is fortunate because self-calibration is unable to remove such problems, and correcting them with resoff is a bad idea. Known ways in which baseline errors can be introduced, are correlator problems (particularly analogue correlators), telescope-shadowing and poorly matched receiver bandpasses.

If you intend to use self-calibration then it is always best to use telescope based editing where pertinent. This is because self-calibration can get confused by incomplete sampling of a given telescope. The 'selfcal' command attempts to detect such cases and flag them rather than make a correction, but there are some cases that it is not able to detect. One such case is where a telescope is sampled by two effectively disconnected arrays, because visibilities that would have connected the two arrays have been flagged. In such cases there are two possible telescope correction solutions.

Multi-IF editing scope

In plot commands that individually show data of all IFs on a single

plot, edits are applied only to the source IFs of the selected points. These include the following commands:

radplot - Plot visibility amplitudes and/or phases versus UV radius.
uvplot - Plot observed UV points on the UV plane.

In plot commands that by their nature can only plot data from one IF at time, edits are by default extended over all IFs (except when restricted by channel-range selections as described below). To optionally constrain edits to only be applied to the displayed IF, all of these commands provide a toggleable option, bound to the 'I' key. The relevant commands are:

cpplot - Interactive display and editing of observed and model closure phases.
projplot - Plot visibility amplitudes and/or phases versus projected UV distance.
radplot - Plot visibility amplitudes and/or phases versus UV radius.
tplot - Plot time sampling for each telescope of an observation.
vplot - Plot visibility amplitudes and phases versus time.

Polarization editing.

In Difmap all edits applied to one polarization are always applied to all other recorded polarizations. However, be warned that unflagging a visibility which has one or more polarizations deleted, can not unflag deleted visibilities, and will thus result in different UV plane sampling in different polarizations. For the most part this is irrelevant, since the select command will not attempt to derive polarizations from combined stokes parameters if one of the stokes parameter visibilities is flagged or deleted, but if you are making maps of the individual stokes parameters to later be turned into percentage polarization maps, the results might be adversely effected.

Multi-channel editing scope

By default visibility edits are applied to all spectral line channels, but all plot commands that allow editing, allow one to constrain edits to the ranges of channels selected with the 'select' command. This option is toggled with the 'W' key.

Self-calibration correction flag scope.

A single self-calibration flag is always effectively applied to all spectral-line channels and polarizations of a single IF, so the 'corplot' command does not have either the 'I' or 'W' options provided for normal visibility editing.

models

An overview of how models are used in difmap.

DIFMAP MODELS

The term 'model' refers to a mixed list of delta, gaussian and other simple components, which when superposed in the image plane, form a facsimile of the source being mapped. In Difmap this model is split into three parts, an established model, a tentative model and a continuum model.

1. The established model.

The established model is an incrementally built up model of the source. Not only is it represented by a component list, but its representation in the UV plane is also calculated at the UV coordinate of each observed visibility in the selected processing stream, to form model visibilities.

2. The tentative model.

The tentative model is a separate component list that represents the latest attempt at modelling the difference between the observed visibilities and the established model visibilities. Since it is expensive to calculate model visibilities, the tentative model is maintained only as a component list. This list is separate from the established model, and it can be discarded at any time via the 'clrmod' command without affecting the established model.

Alternatively, when the tentative model is deemed acceptable, it can be installed as part of the established model, via the 'keep' command, which transfers the tentative model to the end of the established model. Note that the 'keep' command is automatically called whenever a command is invoked that requires model visibilities, but it can also be called explicitly by the user to force an earlier transfer.

The tentative model has its origin in a number of commands:

The 'clean' command incrementally builds up the tentative model, by clean deconvolution of the latest residual map (formed by inverting the difference between the observed and established model visibilities).

The 'modelfit' command also uses the tentative model, placing all the variable model components there (after establishing any non-variable components of any existing tentative model.).

When the 'rmodel' command reads a model from an external file, it

places it directly in the tentative model.

When the 'select' or 'uvaver' commands are called, the UV representation of the established model is no longer valid, so the established model is temporarily demoted by pre-pending it to the tentative model, so that it will be re-established when the 'keep' command is next invoked.

3. The continuum model.

Spectral-line observations are often corrupted by unwanted continuum radiation. The function of the continuum model is to model this continuum emission such that it can dynamically be subtracted from the observed visibilities and then not included in clean maps of spectral-line channels. It is created via the 'setcont' command. In practice, it can be viewed as being part of the established model, in that the model visibilities used to form (observed-model) visibility residuals are actually formed from the superposition of the established and continuum models. The main difference is that the 'restore' command ignores the continuum model when it forms clean maps. A second difference is that since the continuum model is recorded as a separate model component list, the established and/or tentative models for a given spectral-line channel can be discarded without touching the continuum model.

EXAMPLES OF HOW MODELS INTERACT WITH DIFMAP COMMANDS.

The following example shows a typical sequence of commands and how they affect the three difmap models.

First we read in a new observation with the 'observe' command. This clears any existing established, tentative and continuum models.

```
observe file.uvf
```

After setting the map-size and other weighting options, we display the dirty map by typing:

```
mapplot
```

This invokes the 'invert' command to Fourier-invert the observed data and create a dirty map of the source, which is then displayed, allowing us to set up clean windows etc.. At this point there is no model, so all model visibilities have amplitudes and phases of 0, and the map of the residual between the observed and model visibilities is identical to a direct map of the observed data.

Next we decide to clean the map and so type:

```
clean
```

This partially cleans the area inside any windows that we set in `mapplot`, or cleans the whole dirty map if no windows were set. The delta-function model components that this generates are appended to the currently empty tentative-model component list.

Again we display the modified residual map by typing:

```
mapplot
```

but this time we are presented with a map formed from the difference between the new tentative model and the observed data. In this case, since `'clean'` itself subtracted these components from the dirty map, there is no need to re-invert the data, so the residual map is displayed directly.

Next, having looked at the residual map, we decide that there are some clear problems with the data, and decide to look at the observed visibilities by typing:

```
vplot
```

But the `'vplot'` command also allows one to display model visibilities alongside the observed visibilities. Thus the `'vplot'` command needs the UV representation of the current tentative model. To get this, it automatically calls the `'keep'` command, which removes the components of the tentative model, and appends them to the currently empty established model list - at the same time it also calculates their UV representation as model visibilities.

Having finished with the `'vplot'` command, probably having edited bad sections of the data, we decide to again look at the residual map by typing:

```
mapplot
```

but since the model visibilities have been changed by adding in the tentative model, the old residual map is no longer in step with the data, so first the `mapplot` command calls the `'invert'` command to form a new residual map by inverting the residual between the observed and the new established model visibilities. It then displays the new residual map, so that we can set windows etc..

Next we decide to clean the map further by typing:

```
clean 1000
```

This again adds to the newly emptied tentative model, by performing up to 1000 iterations of `clean`. Each iteration creates a new

delta-function model component, which is entered into the tentative model. This may not however lead to there being 1000 components listed in the tentative model, because any delta-components that share the same position are combined into a single component.

To see the effect of the latest clean we type:

```
mapplot
```

Since the clean command didn't change either the established model visibilities or the observed visibilities, mapplot directly displays the residual map as the 'clean' command left it. The 'invert' command is not invoked.

At this point from looking at the map we decide that 1000 iterations of clean was far too many, and decide to discard the tentative model that clean generated, by typing:

```
clrmod
```

This doesn't touch the established model, but again, the residual map that clean produced is no longer in step with the data, so the next command that requires the residual map will automatically re-invert the data, as before. Thus when we decide to repeat the clean with fewer iterations, the 'clean' command re-inverts the data for us first.

```
clean 100
```

At this point we look at the residual map again with the mapplot command and notice that our choice of clean windows has allowed 'clean' to place components where they don't make sense. So in mapplot we tighten the windows to only include the areas where we want components and then type:

```
winmod
```

This command deletes all model components that lie outside the current set of clean windows. Again this invalidates the current residual map, which will automatically be re-inverted when the next command that requires the residual map is required. In the process, the current tentative model will be appended to the end of the current established model and its representation in the UV plane added to the existing established model visibilities.

There are many other commands that involve the three models, but the above example demonstrates the main idea, that in general the user does not have to worry about what is in which model, since each Difmap command knows what it needs and how to get it, without being told.

Other important commands that directly effect the models include:

The 'rmodel' command, used to read in previously generated models from external disk files. This deletes all existing models before reading in the new one.

The 'wmodel' command, used to save models in external disk files.

The 'showmodel' command can be used to display detailed information about each model component in a more human usable form than the wmodel command. It can also be given a filename to write this information to disk.

The 'modelfit' command, used to fit the free parameters of a model to the observed visibilities.

The 'edmodel' command, used to display the current model in your choice of external editor, so that you can edit it directly by hand. This is mainly used when model-fitting to change the set of free parameters.

The 'maplot' command, used as above, but it can also be used to display model components, delete individual model components, and create new model components using the cursor.

The 'setcont' command, used to specify that the current combined model should be moved into the continuum model.

MODEL FILE FORMATS

For documentation of the format of the model files that are read by rmodel and written by wmodel, please see 'help modelfit'.

multi_if

An overview of multi-IF mapping.

Mapping multiple IFs in Difmap.

CONTEXT

The word "IF" is as an acronym for "Intermediate Frequency", which refers to the frequency of the receiver stage at which most signal amplification and filtering occurs. In Radio Astronomy the term has been generalized to mean any single signal path between a telescope and the correlator. Thus, a telescope that simultaneously receives two bandpasses centered at different frequencies, is said to have two IFs. This should be distinguished from the multiple frequency channels of spectral-line observations, where a single bandpass is split into multiple channels during, or after correlation.

UV FITS RESTRICTIONS

Random groups UV FITS files impose certain constraints on IFs. In particular, although each IF is usually centered at a different frequency, the number of spectral-line frequency channels, and the number and types of stokes parameters in each IF, must be identical.

DIFMAP RESTRICTIONS

In addition to the general restrictions imposed by random-groups UV FITS files, there are other restrictions imposed by Difmap.

While Difmap does accept most multi-IF UV FITS files, it rejects those in which the frequencies assigned to the IFs change during the observation. In AIPS parlance, this can be restated by saying that Difmap rejects any file that has multiple frequency IDs in its FQ table.

Mapping multiple IFs in Difmap.

Before mapping a multi-IF observation, one should decide which IFs to map first. For continuum this will normally be all IFs, whereas for spectral-line it is more likely to be a group of spectral-line channels from a single IF. The 'select' command caters for both of these choices, along with others. It takes a polarization name, and one or more spectral-line channel ranges for each IF. IFs of continuum observations are regarded as single-channel spectral-line IFs. For each IF in which channels are selected, the 'select' command forms a psuedo-continuum channel by averaging together visibilities of those channels. One is left with one or more psuedo-continuum IFs, known as the current processing stream. Most commands operate on this

stream. The major exception to this rule is that edits and other corrections are applied to the original data as well as to the current stream. In addition the following commands work directly on the complete data-set and ignore the current stream selection:

```
uvaver   - Time average a UV data set.
wobs     - Save UV data to a random-groups UV FITS file.
specplot - Plot time-averaged visibility spectra.
```

The operations performed by most Difmap commands are automatically applied to all the currently selected IFs, one after the other. Exceptions are certain plotting commands where it only makes sense to display data from one IF at a time. However such commands provide both a command-line argument to explicitly specify a start IF, as well the option to interactively step through sampled IFs, one at time. The relevant plot commands are:

```
vplot    - Display baseline visibility amplitudes and phases versus time.
tplot    - Display the time-sampling of each telescope.
corplot  - Display accumulated self-calibration corrections.
cpplot   - Display observed and model closure phases.
```

Other plot commands separately display data from all IFs on the same plot. Edits in those commands are applied to the source IFs of the individual points selected for editing. Such plot commands include:

```
radplot  - Display visibility amplitude vs. UV radius.
projplot - Display visibility amplitude/phase vs. projected UV distance.
uvplot   - Display the sampling of the UV plane.
```

The only command that explicitly combines IFs is the 'invert' command. This is a command that is usually called for you by Difmap commands that require an up to date version of the latest residual map. The 'invert' command convolves all the residual UV data one IF at a time onto a single regular two dimensional grid of the UV plane, and then Fourier inverts the result to produce a residual map. Combining the data together in this fashion avoids the inter-IF bandwidth smearing that would otherwise occur if visibilities of the same baselines, taken at the same times but at different frequencies were simply averaged to a single point in the UV plane.

Given that Difmap maps are formed by combining all selected IFs, the model that is built up by 'clean' or 'modelfit', is a model that simultaneously represents the structure in all those IFs. This is clearly only a reasonable assumption if the actual structure being mapped doesn't change significantly over the frequency range covered by the selected IFs. The more general case is called multi-frequency synthesis, and is not catered for by Difmap. In such cases you should map each IF separately.

The end result of multi-IF mapping is a single clean map formed from the combination of data from all IFs.

Note that the fact that the selected channels of each IF are averaged to form psuedo-continuum IFs, currently procludes the spectral-line bandwidth synthesis mode of continuum mapping.

PERTINENT COMMANDS AND FUNCTIONS.

nif - Return the number of IFs in the current observation.

nchan - Return the number of channels per IF.

polarization

Overview of Difmap polarization mapping.

Linear polarization mapping in Difmap

The `mapplot` command is capable of plotting the intensity and polarization angle of linearly polarized data as vectors overlaid on a total intensity map. To do this you will first need to switch to multi-model mode (see `help multi_model`). This tells difmap to maintain different models for different polarizations. This means that after making a map of Q, one can select U without losing the Q map. This is important, because both U and Q are simultaneously needed to compute the polarization intensity and polarization angle. The basic procedure is as follows:

1. Switch to multi-model mode:

```
multi_model true
```

2. Set up the map dimensions and gridding weights:

```
mapunits arcsec
mapsize 256,1
uvweight 0, -2
```

3. Make a map of Q.

```
select Q
...clean or modelfit the Q map...
```

4. Make a map of U.

```
select U
...clean or modelfit the U map...
```

5. Make a total intensity map (eg. I, LL or RR).

```
select I
...clean or modelfit the I map.
```

6. Configure polarization vector plotting. In the example below, `mapplot` is being told to display 1Jy of polarized flux as a vector of length 5 arcsec, to only display vectors in pixels where the total-intensity map has at least 0.3Jy, to only display vectors in pixels where the polarized-intensity is over 0.0Jy, and only to only consider drawing vectors in every second pixel along both RA, and Dec.

```
polvec 5.0, 0.3, 0.0, 2, 2
```

6. Display the I map, overlaid with vectors representing the polarization intensity/polarization angle.

```
mapplot pc1n
```

The order in which the maps are made is unimportant, since all that matters is that one has made maps of 'Q', 'U', and in this case, 'I', before running the 'mapplot pc1n' command. However note that the mapplot command displays the currently selected polarization as the underlying grey-scale/contour plot, so regardless of when you made the total intensity map, you should use the 'select' command to select the total-intensity polarization before running 'mapplot pc1n'.

On running the 'mapplot pc1n' command you will see mapplot remake the Q and U maps, using the models that you last made when you mapped them, then make a map of the currently selected polarization. It will then display a map of the currently selected polarization, overlaid with polarization vectors computed from the Q and U maps.

You can also make a residual map overlaid with residual polarization vectors, by typing:

```
mapplot pmap
```

If you haven't made maps of U or Q, the result will be a dirty polarization map.

Beware that with 'mapplot pc1n' or 'mapplot pmap', if you set a clean box in 'mapplot' and then run the 'clean' command, the clean command still only acts on the map of the currently selected polarization. It doesn't clean the Q and U maps as well. To do further cleaning of these maps you should simply select and clean them individually using the 'select' and 'clean' commands. The next invocation of 'mapplot pc1n' or 'mapplot pmap' will then reflect the changes made to these maps.

Note that multi-model mode keeps separate models not only for different polarizations, but also for different spectral-line channel ranges, so that for example, if one makes maps of Q, U and I separately for two different channel-range selections, the mapplot command will correctly show polarization vectors for whichever of these two channel-ranges is currently selected.

Self calibration

In Difmap, self-calibration corrections and editing performed while making a map of one polarization are always applied to all other polarizations. This ensures that polarization calibrations and

consistent UV coverage over all polarizations are preserved. Consequently it makes sense to simultaneously self-calibrate all polarizations by self-calibrating on a map of the strongest polarization (usually I). The same is true of the scope of self-calibration corrections with respect to spectral-line data.

General details

The polarizations recorded in the originating UV FITS file can be displayed with the 'header' command.

If an observation is recorded with multiple polarizations, the 'select' command must be used to select the polarization to be processed. This selection affects what is displayed in all plotting commands, and forms the data upon which all processing is based until the 'select' command is again called upon to change the selection.

The selected polarization may either be one of these polarizations, or be the name of a polarization that can be derived from them.

The names of recognised polarizations are:

I Q U V RR LL RL LR XX YY XY YX PI

The following is a list of polarizations that may be constructed from other polarizations:

Name	Derived from:	Derived as:
I	RR and LL	$(RR + LL)/2$
Q	RL and LR	$(RL + LR)/2$
U	RL and LR	$i(LR - RL)/2$
V	RR and LL	$(RR - LL)/2$
PI	RR and/or LL	(Either RR, LL or the weighted mean of RR and LL, as explained below)

The PI polarization stands for pseudo-I polarization. It is intended for observations of non-circularly polarized sources, where one can then consider LL and RR as being independent measures of I polarization. This is particularly useful for observations made with interferometers with baselines that only sample one of LL or RR. In such cases, selecting 'I' would result in flagged visibilities on these baselines, whereas selecting PI results in the unmatched LL or RR visibility being used. Similarly, since PI considers both LL and RR to be independent measures of I, on baselines that do have both RR and LL, selecting PI results in the weighted mean of these visibilities being used, rather than $(RR+LL)/2$.

spectral_line

An overview of the current spectral-line facilities.

Difmap can make maps from spectral-line FITS files, but a full set of spectral-line facilities has yet to be implemented.

Major features that have not yet been implemented.

A major omission is a command to write spectral-line map cubes. At the moment, each spectral-line map must be written to a separate FITS file, and then be combined externally (eg. by MCUBE in AIPS). I hope to do something about this in the near future.

How to make spectral-line maps in Difmap.

It is only possible to make a map of one aggregate of channels at a time. This isn't as bad as it sounds, as I will attempt to show in the following discussion.

1. Looking at visibility spectra.

In order to determine which spectral line channels contain interesting features and thus which channels to map, the first thing to do is to make plots of visibility spectra. The 'specplot' command provides this facility. By default, this command shows individual spectra to each baseline in the observation. For each baseline it performs a vector average of the visibility spectrum over all integrations. This is the default mode of display. However, the specplot command also allows one to restrict the source of visibilities to specified time ranges, polarizations, UV ranges, or groups of baselines. One can plot multiple spectra per page for different values of these selections, so for instance, one can see how a spectrum of a given baseline falls off with UV radius.

Specplot is a complicated command, and further details of how to use it can be found in the on-line specplot help pages.

2. Selecting one or more channels to be mapped.

The 'select' command allows one to average together one or more ranges of channels per IF. The resulting averages (one per IF) constitute the data that will thereafter be processed, until the 'select' command is again invoked to change the selection.

For instance:

```
0>select I, 10,20, 30,40
```

extracts the total intensity (I) over channels 10..20 and 30..40 and averages the results into a single psuedo channel, referred to as the current processing stream.

If the observation contains more than one IF, then channel numbers are assigned contiguously across all IFs. For example, if one has two IFs, containing 64 channels each, then channel numbers between 1 and 64 refer to channels of the first IF, and 65 to 128 refer to channels of the second IF. The same convention is used by the 'specplot' command. Note that this does not require that IFs be contiguous in frequency, or even that the interchannel frequency increment have the same sign in neighboring IFs. It is just a convenient way to specify IF-specific channel ranges without actually having to specify IF numbers. When doing spectral-line mapping the averaged channels are usually taken from a single IF. When this happens, the other IFs are unsampled and will not be displayed in any of the visibility plotting commands.

Note that channel averaging as described above precludes the possibility of bandwidth synthesis. Thus if the bandwidth enclosing the averaged channels is too wide, bandwidth smearing will result. Note that this is different from the way that multiple IFs are handled. Multiple IFs are kept apart, except during the gridding step of fourier inverting visibilities into the map plane.

3. Processing a selected stream.

Once the 'select' command has been used to average together a range of channels for processing, all difmap commands operate on the resulting psuedo channel, or processing stream. In particular all plot commands display just this stream. One may then make a map as though one were processing continuum data, using the 'clean', 'selfcal' and other commands to build up a model of the image plane representation of the data. See the cookbook for details on making continuum maps.

4. Selfcal calibrations.

A common practice in spectral-line work is to make a good map of one strong channel (or range of channels) and use the selfcal calibrations that were needed to produce that map, to calibrate the rest of the channels. In recognition of this fact difmap automatically applies selfcal calibrations identically to all channels (and polarizations), regardless of the selected data stream. This is an important feature. If instead, one were allowed to self-calibrate different channels separately, the relative positions of features in different channels would be lost.

A special case of the above technique is to map the background

continuum and use the calibrations needed to get a good continuum map as the basis for the spectral-line calibration. To do this, all you have to do is select the range of channels that contain just continuum data and map them, using `selfcal` wherever appropriate.

5. Editing.

Editing is accomplished interactively with various plot commands. By default, although these commands only display visibilities of the selected stream, edits are applied equally across all spectral-line channels of all IFs. The plot commands provide toggleable options to change this default, such that edits can be applied just to the displayed IF and/or just to the spectral-line channels of the selected stream. The key bindings for these two toggles are 'I' and 'W' respectively. The 'I' toggle is not provided in commands which display all IFs simultaneously since edits are then always applied just to the IFs of the specified visibilities.

6. Continuum subtraction.

Continuum subtraction in `difmap` exploits difference-mapping facilities to provide a more flexible scheme than is generally provided in other programs. To use it one must make a clean map of the continuum channels to build up a model of the continuum. Once this has been done the model can be installed as a background continuum model using the `'setcont'` command. The UV representation of this model is then subtracted from the current and future processing streams, until deliberately discarded with the `'clrmod'` command. Important advantages of this technique are that self-calibration is still applied to the unsubtracted data, that the continuum model can be incrementally improved by later additions with the `'setcont'` command, and that the continuum model can even be removed with the `'clrmod'` command when no longer required. For further details read the help information on the `'setcont'` command.

7. Making a cube of maps.

To produce multiple channel maps it is necessary to map each channel range separately. At the moment, the only way to save these maps is in separate FITS files, using multiple calls to the `'wmap'` command. I hope to support map cubes in the future. To create a map cube, it is thus necessary to produce multiple map files and then combine them into a cube in another program, eg. `MCUBE` in `AIPS`.

If you follow the scheme described above, where one psuedo channel is mapped in order to solve for self-calibration solutions for all channels, then all that is then required is a way to run the `'select'`, `'clean'` and `'wmap'` commands in a loop to produce multiple

channel maps. This can be done using difmap 'do' loops. For example:

```
0>integer channel
0>do channel=10,20,4
1> select I, channel, channel+3
1> clrmod true
1> clean 300,0.01
1> wmap "map_" // strnum(channel) // ".fits"
1>end do
0>
```

This would produce maps of channels 10..13, 14..17, and 18..21, and write them to fits files called: map_10.fits, map_14.fits and map_18.fits.

These maps could then be read into AIPS via a one-line loop on the AIPS command line, of the form:

```
>for i=10 to 20 by 4;infile='dir:map_'!!char(i)!!'.fits';go fitld
wait fitld; end
```

Where 'dir' is the name of an environment variable that you must define before running AIPS to point at the directory in which the files reside. The individual maps could then be combined into a cube via AIPS MCUBE.

The above Difmap loop is not very sophisticated and is just an example. A better example would include the addition of the mapplot command to allow interactive selection of clean windows. This in turn could be put in a nested loop along with the clean command (using a smaller number of clean iterations per loop), so that the clean windows could be adjusted during cleaning, as opposed to setting them once before cleaning each map plane.

The ' ' syntax in the above simply allows a string expression to be used where a literal string (one without enclosing ""s) is expected.

subarrays

A discussion of sub-array handling in Difmap.

Sub-arrays in Difmap.

CONTEXT

To increase the UV coverage and sensitivity of an experiment it is common to combine one or more observations from different times, telescope arrays, or different configurations of a single interferometer. In each case, the antennas in the different observations usually have to be calibrated separately, so the telescope arrays are treated as being different sub-arrays of a single hypothetical interferometer.

Difmap details.

Difmap does handle multiple sub-arrays if they all appear within a single input file, but it does not currently provide facilities for merging two or more UV FITS files to produce a multi-sub-array data-set. This should instead be done before entering Difmap, say with DBCON in AIPS (DOPOS(1,1) > 0; DOARRAY < 0).

Sub-arrays are actually effectively unrelated, so the visibilities for each sub-array must actually be separately processed. Most Difmap commands do this automatically, processing each sub-array one after the other. Exceptions to this rule are certain plot commands, along with a few sub-array manipulation and informational commands. In these cases, sub-arrays must be referred to by number. The association between sub-array numbers and arrays can be ascertained with the 'header' command.

Telescope specification strings.

Since a single telescope may appear in more than one sub-array, Difmap usually requires that when you name a telescope, baseline, or closure-triangle, that you also specify the number of the sub-array in which the telescope(s) belong. The only exception to this rule, is when the telescopes belong to the default sub-array, which is 1 if the specification is given on the command line to a command, or equal to the number of the sub-array that is currently being displayed when given interactively in a plot command. In these cases the sub-array index can be omitted. In particular, if the observation only has a single sub-array, then the default sub-array always refers to that sub-array.

The general form of all supported telescope specifications is:

```
[sub-array-number:]telescope_name_a[-telescope_name_b[-telescope_name_c]]
```

Elements of the specification that are enclosed in [] are optional.

The actual number of telescope names expected depends upon whether a telescope, baseline or closure-triangle specification is required by the Difmap command. In each case, trailing telescope names can be omitted. Usually when this is done the first recorded telescopes in the sub-array that can be associated with the given telescope name(s), are substituted. Alternatively, commands that accept more than one type of telescope specification note whether a telescope, baseline or complete closure-triangle specification is given and act accordingly.

Example telescope specifications are:

1. To specify an antenna called OVRO, in sub-array 1 the specification would be:

```
1:ovro
```

2. To specify the baseline formed by telescopes named OVRO and WSRT in sub-array 1, the specification would be:

```
1:ovro-wsrt
```

3. To specify the triangle formed by telescopes named OVRO, WSRT and BONN in sub-array 1, the specification would be:

```
1:ovro-wsrt-bonn
```

Telescope names are matched using a case-insensitive min-match search of all telescopes in the given sub-array. Thus if sub-array 2 contained telescopes named BONN and BOLOGNA, then:

```
2:bon-bol
```

would be sufficient to select baseline BONN-BOLOGNA, whereas:

```
2:bo-bol
```

would evoke an error message.

Note that the : separator after a sub-array specification is only taken to denote such a separator if it is preceded by an integer at the start of the specification. This makes it possible to select VLA antenna names, which themselves contain a colon. Thus, if the default sub-array is sub-array 1, then:

```
1:vla:e8
```

and

vla:e8

are both equally acceptable specifications for telescope VLA:E8 in sub-array 1.

LIMITATIONS

At the moment all sub-arrays must all have been observed at the same frequency and bandwidth. This translates to the requirement that each sub-array be referred to by the same frequency ID in the input UV FITS file. If this restriction proves to be a problem then I will endeavour to implement the more general case of allowing a separate FQID per sub-array.

Since random-groups UV FITS files also impose the restriction that all visibilities have the same number of frequency channels, polarizations and IFs, removing the above restriction would not be as useful as it might be, so I have not implemented it.

Sub-array specific commands and functions.

The command that provides the most general information on sub-arrays is the 'header' command. This also shows other information about the observation. Its limitation is that it is given in textual form, so it can not be used as an aid to writing scripts. The following functions can be used to obtain a few bits of information about sub-arrays:

- nsub - Returns the number of sub-arrays in the data-set.
- nbase - Returns the number of baselines in a given sub-array.
- ntel - Returns the number of telescopes in a given sub-array.
- basename - Returns the name of the n'th baseline in a given sub-array.
- telname - Returns the name of the n'th telescope in a given sub-array.

Plot commands that display data for one sub-array at a time include:

- tplot - Plots the telescope versus time sampling of sub-arrays.
- vplot - Displays visibilities of sub-array baselines.
- cpplot - Display closure-phases of sub-array closure-triangles.
- corplot - Display self-calibration telescope corrections.

In these commands, capital N and P can be used to move to the Next or previous Sub-array in order of sub-array index. The 'T' key can also be used to select a given plot via a telescope specification string.

Plot commands that plot all sub-arrays at once, but allow one to highlight a given telescope, include:

radplot - Display visibilities versus radial UV distance.

projplot - Display visibilities versus projected UV distance.

uvplot - Display visibility sampling of the UV plane.

Each of these commands responds to the 'T' key to allow the user to request that a given telescope be highlighted, via a telescope selection string.

Difmap commands and functions

The following pages show the contents of the help files of each difmap command and function. The same pages can be seen from within Difmap by typing:

```
help name_of_command_or_function
```

addcmp *flux, vf, x, y, vxy, major, vm, ratio, vr, phi, vp, type, freq0, spcind,*
vs

Add a model component by hand to the tentative model.

EXAMPLE

```
0>mapunits arcsec
```

```
0>addcmp 0.1, false, 0, 0, true, 0.3, true, 0.5, true
```

The above adds a gaussian component of flux fixed at 0.1 Jy, positioned at the center of the current map but allowed to move during model fitting, having a major axis of 0.3 arcseconds FWHM and an axial ratio of 0.5 both of which are allowed to vary during model fitting, and an implicit major axis position angle of 0 degrees. The spectral index defaults to zero and is not allowed to vary.

PARAMETERS

- flux - The integrated flux of the component, measured in Jy.
- vf - If TRUE allow the flux to vary during model fitting.

- x - The relative X-axis position of the component centroid in the current map units. (See help mapunits).
- y - The relative Y-axis position of the component centroid in the current map units. (See help mapunits).
- vxy - If TRUE allow x and y to vary during model fitting.

- major - The FWHM major axis of the elliptical aspect of the component measured in map units (See help mapunits).
- vm - If TRUE allow major to vary during model fitting.

- ratio - The ratio of the minor axis to the major axis (0 -> 1).
- vr - If TRUE allow the axial ratio to vary during model fitting.

- phi - The Position angle of the major axis (degrees North -> East).
- vp - If TRUE allow phi to vary during model fitting.

- type - The component type. The default if not given is to assume that the component is a delta function unless a non-zero major axis has been provided.
Valid types include:
 - 0 - Delta function.
 - 1 - Gaussian.
 - 2 - Uniformly bright disk.
 - 3 - Optically thin sphere.
 - 4 - Ring.

- freq0 - The frequency for which the flux value is defined, specified in Hz.

This is only relevant when used with a non-zero spectral index.

spcind - The spectral index of the component, or zero if not relevant.
The flux of the component in a given IF is given by:

$$\text{flux}[\text{IF}] = (f[\text{IF}] / \text{freq0})^{\text{specind}}$$

Where $f(\text{IF})$ is the mean frequency of the channels that are currently selected within the specified IF.

vs - True if the spectral index should be varied during model fitting.

CONTEXT

This command is provided mainly for use in automatic mapping scripts where the normal interactive methods of adding model components (see `help mapplot` and `help edmodel`) are inappropriate. In such contexts this command can be used by a script to enter guesses at model component characteristics and then use the `modelfit` command to fit it to the data in order to parameterize maps after hands-off mapping. Note that there are some restrictions on which parameters can actually be fixed during modelfitting. These restrictions are silently enforced by the `modelfit` command and you should acquaint yourself with the `modelfit` help information before using this command seriously.

RELATED COMMANDS

`mapplot` - (In the sense that it can be used to interactively add components).
`edmodel` - Edit the current models in your favorite editor.
`modelfit` - Fit a variable image plane model to UV data.

addhist *text*

Append a new line to the recorded history of an observation.

EXAMPLE

```
O>addhis "DIFMAP This is a new history line."  
O>
```

PARAMETERS

text - The text of the new history line. This will be truncated to a maximum of 80 characters where necessary.

CONTEXT

The only history information that difmap writes is a line to say when the UV FITS file was read into difmap. If you wish to record more processing history this command may be used to do it.

RELATED COMMANDS

showhist - Display the history recorded with the current observation.
clrhist - Clear the history information recorded with the observation.

addwin *xa, xb, ya, yb*

Append a new entry to the list of clean windows.

EXAMPLE

```
0> addwin -1.5,1.0,-3.0,2
```

```
0>
```

PARAMETERS

xa - The relative Right-Ascension of either edge of the new window.
xb - The relative Right-Ascension of the opposite edge to '*xa*', of the new window.
ya - The relative Declination of either edge of the new window.
yb - The relative Declination of the opposite edge to '*ya*', of the new window.

Coordinates should be specified with the same sign convention and units as displayed in mapplot. (See help on the mapunits command).

CONTEXT

Normally clean windows should be set with the cursor in the mapplot command. Each new window added is appended to a list of clean windows for use in clean. The addwin command provides a way to add windows to the same list, but directly at the command line.

RELATED COMMANDS

mapplot - Allows adding and deleting windows with the cursor.
delwin - Deletes all clean windows.
clean - Cleans within the specified clean windows.

antenna_beam *antennas, samples, resolution, frequency*

Specify a beam for one or more antennas (for primary beam calculations).

EXAMPLE

To install a simple gaussian approximation of a beam profile, you can use the `gauss_array()` function to provide the array of beam samples.

```
0>mapunits arcmin
0>antenna_beam "1:A075 1:A080", gauss_array(45.1, 4.0, 0.5), 0.5, 30.0e9
A new antenna beam has been successfully installed.
0>
```

Here "1:A075 1:A080 1:A014" indicates that the beam should be used for three antennas of subarray 1, named A075, A080 and A014.

Regarding the arguments of the `gauss_array()` command, 45.1 is the FWHM of the beam profile (in arcminutes in this case), 4.0 specifies how far out to sample the beam profile, expressed as the number of sigma, and 0.5 is the resolution along the x-axis of the gaussian. 30.0e9 is the frequency at which the primary beam is being specified, (30GHz in this case).

For non-gaussian beam profiles, you must specify the array of samples explicitly. For example, to install the above gaussian beam without using the `gauss_array()` function, you could do the following:

```
0>mapunits arcmin
! Convert the FWHM of the gaussian to the equivalent standard deviation.
0>float sigma
0>sigma = 0.4246609001 * 45.1
0>antenna_beam "A075", exp(-0.5 * (ramp(0, sigma*4, 0.5)/sigma)^2), 0.5, 30.0e9
0>...
```

Breaking this down, note that the `ramp()` function,

```
ramp(0, sigma*4, 0.5)
```

returns an array of values which linearly increment by 0.5 from 0 to `sigma*4`. The resulting array is then divided by `sigma`, by writing:

```
ramp(0, sigma*4, 0.5) / sigma
```

This returns another array, which is then squared to produce another array:

```
(ramp(0, sigma*4, 0.5) / sigma)^2
```

Finally this is multiplied by -0.5 and passed to the `exp()` function:

```
exp(-0.5 * (ramp(0, sigma*4, 0.5)/sigma)^2)
```

This returns another array, which is the required gaussian array sampled out to 4 standard deviations.

PARAMETERS

- antennas - The list of antennas that have the specified beam profile. Multiple antenna names can be specified, separated by spaces.

- samples - An array of values which sample the beam profile.

- resolution - The angular distance on the sky between samples. The units can be changed with the 'mapunits' command. If the mapunits command hasn't been used, the units are milli-arcseconds.

- frequency - The frequency at which the beam in the samples array is being specified, measured in Hz. Note that the primary beam for other frequencies is computed under the assumption that the width of the beam width scales inversely proportional to frequency.

CONTEXT

Where the antennas that make up an interferometer have diameters that are comparable to the distance between them, the field of view of the interferometer is limited by the diameters of the point spread functions of the individual antennas, which we will refer to as voltage beams. The field of view is limited to the diameter of the product of the voltage beams of the individual antennas of each baseline. This product is known as the primary beam. In the case where all of the antennas are identical and aren't shadowed by neighboring antennas, all baselines have the same primary beam, and the resulting map is effectively multiplied by this beam, centered at the original pointing center. If one cleans or modelfits a dataset from such an interferometer, without taking account of the primary beam, model components away from the center of the primary beam have lower fluxes than their true values. If alternatively, one tells difmap about the primary beam, difmap will thereafter appropriately scale up the fluxes of new model components according to their distance from the pointing center.

This command sets the voltage beam of one or more antennas. If all of the antennas in the array have the same characteristics, then the alternate primary_beam command will be more convenient, because it

specifies a single primary beam for the whole array of antennas.

basename *ibase, isub*

Returns the name of baseline 'ibase' of sub-array 'isub'.

EXAMPLE

Example 1 - Print out all (nbase) baseline names in sub-array 1.

```
0>do i=1,nbase
1> print basename(i)
1>end do
1:WSRT-BOLOGNA
1:WSRT-HSTK
1:BOLOGNA-HSTK
1:WSRT-VLBA_KP
1:BOLOGNA-VLBA_KP
1:HSTK-VLBA_KP
1:WSRT-VLBA_LA
1:BOLOGNA-VLBA_LA
1:HSTK-VLBA_LA
1:VLBA_KP-VLBA_LA
0>
```

Example 2 - Print the name of the third baseline in sub-array 2.

```
0>print basename(3,2)
2:OVRO-WSRT
```

The "number:" prefix indicates the number of the sub-array to which the baseline belongs.

PARAMETERS

ibase - The number of the baseline whose name is required.

isub - Default: 1
The number of the sub-array from which the baseline name is required.

RELATED FUNCTIONS

nbase - Return the number of baselines in a given sub-array.

nsub - Return the number of sub-arrays in an observation.

beamloc *xcenter, ycenter, minsize, maxsize*

Change when and where mapplot plots the clean beam ellipse.

EXAMPLE

```
0>beamloc 1.0, 0.0, 0.5
```

```
Mapplot clean beam ellipse center: 1,0. Max size: 0.5.
```

```
0>beamloc -1
```

```
Mapplot will not plot a clean beam ellipse.
```

PARAMETERS

- | | | |
|---------|---|--|
| xcenter | - | The location of the beam center along the X-axis, measured in normalized plot coordinates (0 -> 1). The initial default is 0.0, the left of the plot. |
| ycenter | - | The location of the beam center along the Y-axis, measured in normalized plot coordinates (0 -> 1). The initial default is 0.0, the bottom of the plot. |
| minsize | - | The minimum allowed fractional extent of the beam ellipse with respect to the plot width and height. The beam will not be plotted when its size is smaller than this limit. The initial default is 0.01. |
| maxsize | - | The maximum allowed fractional extent of the beam ellipse with respect to the plot width and height. The beam will not be plotted when its size exceeds this limit. The initial default is 0.3. |

CONTEXT

When plotting clean maps, mapplot usually plots a filled ellipse representation of the clean beam in the bottom left corner of the map.

For esthetic and practical reasons, it is usually desirable to prevent the beam ellipse from being plotted when it is either very small with respect to the size of the plot, or so big that it comes to dominate or obscure important parts of the plot. The latter is particularly important when zooming in on fine detail. The 'minsize' and 'maxsize' arguments allow one to specify the allowable range of sizes as fractions of the width and height of the plot.

The position at which the center of the beam is plotted can also be changed via the 'xcenter' and 'ycenter' arguments. Both positions are

specified in normalized plot coordinates, with 0,0 referring to the bottom left corner and 1,1 referring to the top right corner. If either 'xcenter' or 'ycenter' lies outside the range 0 and 1, then no beam will be drawn. Otherwise the beam will be positioned at the position given, with the exception that if the position would place part of the beam outside the map, it will be moved inwards, suitably spaced from the edge.

RELATED COMMANDS

mapplot - Display map and beam images.
xyrange - Set the boundaries of the sub-image to be displayed in mapplot.
loglevs - Set logarithmic levels.
mapcolor - Select mapplot colormap type and transfer function.
mapfunc - Change the transfer function used in mapplot.

check_uvaver *aver_time, east, north*

Check if a given averaging time would cause time smearing.

EXAMPLE

```
0>check_uvaver 60,11,11
Maximum U extent: 849.469 on baseline 1:A074-A078 in IF 4, ch 8, pol XX
Maximum V extent: 71.7559 on baseline 1:A077-A078 in IF 4, ch 8, pol XX
Potential flux reduction at x=11,y=11 arcsec: 0.34 %
0>
```

In the above, any the lines that don't begin with a 0> prompt are messages displayed by the check_uvaver command.

PARAMETERS

aver_time - The duration of the new integrations in seconds.

east - The eastward offset of a position of interest, relative to the pointing center (ie. the center of the map). Unless the mapunits command has been used to change the default units used for distances, the eastward offset should be specified in milli-arcseconds.

north - The northward offset of a position of interest, relative to the pointing center (ie. the center of the map). Unless the mapunits command has been used to change the default units used for distances, the northward offset should be specified in milli-arcseconds.

CONTEXT

Time averaging of visibilities can smear sources, especially sources far from the map center. For this reason, before using the uvaver command to perform time averaging, it is recommended that the check_uvaver command be used to see whether any important sources within the map will be badly affected by the chosen averaging time. The amount of smearing suffered by a source depends on where it is in the map. It is worse for sources further away from the center of the map, and along directions parallel to the longest tracks in the UV plane.

For each averaged output visibility, the uvaver command combines multiple visibilities from neighboring times. These visibilities are also close to each other in the UV plane, such that each averaged visibility comes from a spread of UV coordinates. This essentially smooths the visibilities in the UV plane, which has the effect of scaling the image plane with the Fourier transform of the smoothing

window. The scaling function is unity at the origin of the map, but decreases further from the origin, such that sources further from the map center are scaled down more in flux. Unfortunately, the scaling effect differs from one averaged visibility to the next, so the scaling depends on spatial frequency. This is why sources appear smeared rather than just being weaker than they should be.

The `check_uvaver` command finds the averaged visibility that suffers most from the above effect, and it reports the flux weakening due to this visibility. This makes it a pessimistic estimate of how much a source would actually be weakened by `uvaver`.

Since the estimated flux reduction is derived from the worst affected visibility, it should be safe to time-average the visibilities if the reported flux loss is no more than a few percent.

Since the time-averaging degradation varies from one part of the map to another, the `check_uvaver` command requires you to specify a position in the map. If the UV coverage of the observation (see `uvplot`) has a similar width in all directions, and you just wish to know the worst case effect on any source within the primary beam, then you can just set the east argument to the radius of the primary beam, and the north argument to zero. On the other hand, if the UV coverage is very asymmetric, it will be necessary to explore the affect at different sources of interest, by specifying their locations as eastward and northward offsets from the map center using the east and north arguments.

THEORY

Imagine a time sequence of visibilities from a single baseline. Now collect these visibilities into a succession of time bins, each of duration `aver_time`. In the UV plane the locations of the visibilities in each bin will trace out a short line, due to the rotation of the Earth, which slowly changes the projected length and orientation of the baseline. For the moment, imagine that each of these short lines of visibilities just happen to fit within boxes of width `du` along the U axis, and `dv` along the V axis. If we take the visibilities within each of these boxes and compute their averages to obtain averaged visibilities, then the effect is as though we had applied a two dimensional boxcar smoothing function to the visibilities in the UV plane, or size `du x dv`. In other words the UV plane has effectively been convolved with a top-hat function of width `du` along the U axis, followed by a top-hat function of height `dv` along the V axis. This convolution in the Fourier plane results in the image plane being multiplied by the Fourier transforms of these top-hat functions, which are sinc functions. These sinc functions are unity at the origin, but decrease with increasing distance from the map center, so they mostly affect sources far from the map center.

A top-hat function of width du in the UV plane can be written as:

$$f(u) = 1/du \text{ where } |u| < du/2, \text{ and } 0 \text{ elsewhere.}$$

Note that the u coordinate here is a unitless number of wavelengths, whereas some published works use light-seconds, which need to be scaled by the observing frequency to obtain the u coordinate given here.

The Fourier transform of the above top-hat function is:

$$F(l) = \sin(\pi \cdot du \cdot l) / (\pi \cdot du \cdot l)$$

Here, l is an eastward offset from the map center. The combined effect of the two dimensional smoothing by du and dv is:

$$R(l,m) = \sin(\pi \cdot du \cdot l) / (\pi \cdot du \cdot l) * \sin(\pi \cdot dv \cdot m) / (\pi \cdot dv \cdot m)$$

The above treatment assumes that all time bins contain visibilities bounded by identically sized boxes in the UV plane. In practice this is rarely true. Bins further from the origin of the UV plane tend to cover larger areas of the UV plane, which means that the actual flux reductions due to time averaging vary, generally increasing with spatial frequency. This is why sources tend to be smeared by time averaging, rather than just being scaled down in flux, which is what the above treatment would otherwise imply.

Since the regions covered by the bins in the UV plane vary in size, the `check_uvaver` command searches for the averaged visibility that originates from the widest area in the UV plane. This yields the worst case flux reduction. This reduction may only apply to a few other averaged visibilities, so if the reported worst-case reduction is small, then the user can be confident that sources at the specified position will not be significantly affected.

A similar treatment of this effect, with the same result, can be found in VLA Scientific Memorandum No. 137. Currently this can be obtained from:

https://library.nrao.edu/public/memos/vla/sci/VLAS_137.pdf

Beware that in the above work, the U and V coordinates are distances in units of light-seconds, which is why their equation scales these coordinates by the frequency of observation, whereas the equations given above do not.

Another treatment can be found in the following book. However beware that this suffers from some errors. Specifically, equation 18-30 scales the U and V coordinates by the observing frequency, which implies that U and V are in light-second units, whereas equation 18-31

omits the frequency, which implies that U and V are actually numbers of wavelengths. Also 18-31 should be scaled by the averaging time, T_a , which is mentioned in the text just before it, but omitted from the equation.

Authors: Bridle, A. H. & Schwab, F. R.

Journal: Synthesis Imaging in Radio Astronomy II, A Collection of Lectures from the Sixth NRAO

clean *niter, gain, cutoff*

Clean a residual dirty map with the dirty beam.

EXAMPLE - (there is a more general example later in this help topic)

```
0>clean 100,0.05,0.0
Inverting map and beam
Estimated beam: bmin=1.8350 mas, bmaj=6.1080 mas, bpa=27.4 degrees
Clean: niter=100 gain=0.05 cutoff=0
Component: 050 - total flux cleaned = 3.218951 Jy
Component: 100 - total flux cleaned = 4.508268 Jy
Total flux subtracted in 100 components = 4.508268 Jy
Clean residual min=-0.184183 max=0.394841 Jy/beam
Clean residual mean=0.024779 rms=0.092559 Jy/beam
0>
```

All but the first line are responses from 'invert' and 'clean'.

PARAMETERS

- niter - The Maximum number of CLEAN iterations to be attempted.
If this is -ve then CLEANing will stop on the component before the first negative component.
- gain - The CLEAN loop gain (normally 0 -> 1). This sets the fraction of the peak (absolute) residual that is subtracted on each CLEAN iteration. The choice of a gain is a trade-off between the ideal (as small as possible) and the speed at which the map is reduced to the noise. It is common to start with a few iterations at high clean gain - say 0.1 - to clean away any bright point-like features such as a Galaxy core, and then to reduce it to clean the lower surface brightness extended structures.
- cutoff - Cleaning will stop if the maximum absolute residual falls to this value or lower (Jy). This may be used to prevent CLEAN from cleaning below the flux level at which (due to artifacts or noise) structures in the residuals can be believed.

'clean' remembers the parameters that were given to it the previous time it was invoked. Any omitted arguments will retain these values.

WINDOWS

In addition to the above explicit command line parameters, CLEAN can be supplied with a list of rectangular windows, such that CLEAN components are only taken from within these windows. If no windows are supplied then CLEAN will use the whole map. Clean windows can be set

directly via the 'mapwin' command or set with the cursor in the 'mapplot' command. The latter is the easiest to use.

The registration between map pixels and clean window edges is such that all pixels whose centers lie inside the windows are used. Where the window is so thin as to not enclose any pixel centers, the pixels with the nearest centers are used. Windows may be set interactively with the cursor (see help on the 'mapplot' command).

CONTEXT

Clean takes the residual dirty map from the previous clean and iteratively subtracts the dirty beam pattern convolved with the loop-gain times the brightest residual pixel. The end result is a new residual dirty map and a set of delta-function model components representing the flux subtracted from the selected pixels.

Errors in the gains and phases at individual telescopes set the level down to which it is safe to CLEAN. To enable further CLEANing it is necessary to follow CLEAN by self-calibration, which compares the observed visibilities to the CLEAN model to determine self-consistent solutions to the telescope based gain and phase errors. Having done this it should be possible to CLEAN to a lower residual flux than before.

A typical mapping sequence is:

```
0> observe myfile.uvf  (Read the FITS UV-data file)
0> vplot               (edit the visibilities to remove bad points)
0> mapsize 256,0.5     (set the map grid and cell-size)
0> mapplot             (Display the dirty map and set CLEAN windows)
0> clean -100,0.1,0.2  (Clean up to 100 components at a gain of 0.2,
                       stopping before the first negative component or
                       when the peak residual is less than 0.1 Jy)
0> selfcal             (Apply phase-only self-cal)
0> mapplot             (Display corrected residuals and change the
                       clean windows if necessary)
0> clean 100,0.05,0.05 (Use a lower clean gain and allow negatives,
                       while cleaning down to a lower residual flux)
0> selfcal true,true,900 (Amplitude+phase self-cal)
0> vplot              (Edit discrepant points - compare with model)
0> mapplot
0> clean
```

etc.. until the residual flux won't further decrease or the self-cal fit starts to diverge.

```
0> mapplot cln        (Look at the clean map)
0> save mynewfile      (Save the UV data, the map, the model, the
                       clean windows and the difmap parameters)
```

0> exit (Finished)

SIDE EFFECTS

If the beam and/or residual map need to be re-calculated following data editing, re-weighting etc.. then 'clean' will call 'invert'.

LIMITATIONS

Clean only cleans the inner half of the actual map array that is allocated. (mapplot only displays this area when plotting the map). Thus 'mapsize 256,0.5' will result in a map area 128x128 pixels wide.

RELATED COMMANDS

selfcal - Uses the clean model in self-calibration.
gscal - Uses the clean model and self-calibration to find overall gain corrections for each telescope.
invert - Converts residual UV data to the residual dirty map.
restore - Generate the clean map.
vplot - Interactive display and editing of visibilities.
mapplot - Display residual or clean map, or dirty beam.

clear_markers

Delete the current list of map markers.

EXAMPLE

```
0>clear_markers
Marker list cleared.
0>
```

CONTEXT

The 'mark_xy' and 'mark_radec' commands add markers to a list that is passed to mapplot. The clear_markers command, empties this list.

RELATED COMMANDS

mapplot	- Draw a map.
mark_xy	- Specify a map marker relative to the map center.
mark_radec	- Specify a map marker to be drawn at a given RA,Dec.
wmarkers	- List map marker commands to stdout or a file.
delmarker	- Delete a given map marker.

`clear_models` *filename*

Discard the current set of channel-range/polarization specific models.

EXAMPLE

```
0>clear_models 3c84.mtab
```

CONTEXT

In multi-model mode (see `help multi_model`), different models are associated with different channel-range/polarizations selections. Previously made models for a given channel-range/polarization selection are switched in by the 'select' command when the same selection is made again. To start afresh, the 'clear_models' command allows one to discard all of these models.

RELATED COMMANDS

```
multi_model  - Switch to and from multi-model mode.
read_models  - Restore a set of multi-model models from a file.
write_models - Save the current set of multi-model models to a file.
```

clrhist

Clear the history information recorded with the observation.

EXAMPLE

0> clrhist

CONTEXT

It is normal for UV FITS files to contain an extensive processing history, such as that recorded by AIPS. While Difmap adds very little to this Difmap does preserve it and passes it on to output UV FITS and map image FITS files, and while in some cases it might be useful, in others it may significantly enlarge data files for no good reason.

The 'clrhist' command clears the currently recorded history.

RELATED COMMANDS

addhist - Add a new line of history.

showhist - Display the currently recorded history.

clrmod *doold, donew, docont*

Clear one or more of the established, tentative and continuum models.

EXAMPLES

0>clrmod

clrmod: Cleared the tentative model.

0>

0>clrmod true

clrmod: Cleared the established and tentative models.

0>

0>clrmod true,true,true

clrmod: Cleared the established, tentative and continuum models.

0>

PARAMETER

doold - Default=false

If specified as true, then the established model will be cleared.

donew - Default=true

If this argument is omitted, or explicitly specified as true, then the tentative model will be cleared.

docont - Default=false

If specified as true, then the continuum model will be cleared.

CONTEXT

In difmap the term 'model' refers to a mixed list of delta, gaussian and other simple components, which when superposed in the image plane, form a facsimile of the source being mapped. This model is split into three parts, an established model, a tentative model and a continuum model. The function of the 'clrmod' command is to clear one or more of these models.

When the 'clrmod' command is invoked with no arguments, it clears just the tentative model. This is most useful to discard the latest results of the 'clean' command, after cleaning too deeply.

When the 'clrmod' command is invoked as 'clrmod true', it deletes both the established and tentative model, but leaves any existing continuum model untouched. This is commonly invoked after the 'select' command has been used to select a new data stream, when which the model for the previous selection is a bad match to the new selection.

For more information about models type 'help models'.

clroff

Undo all resoff calibration-offsets applied during this difmap session.

CONTEXT

If you have cold-feet about having used resoff to make your model and observed data look better, clroff will remove the resoff corrections.

corplot *tel_name, IF_index*

Display accumulated telescope based self-cal corrections versus UT.

EXAMPLES

1. Display the corrections for the BONN telescope in sub-array 1.

```
0>corplot BONN
```

2. Display corrections for telescope 3 in sub-array 1.

```
0>corplot 'telname(3)'
```

3. Display corrections interactively for any telescopes.

```
0>corplot
```

4. Display a telescope from sub-array 2, and its corrections in IF 3.

```
0>corplot 2:BOLOGNA, 3
```

PARAMETER

tel_name - Optional - default="".

A telescope name specification. In interactive mode this specifies the first telescope to be plotted. When plotting to a non-interactive device it specifies a range of telescopes to be displayed. See the `antenna_names` help topic for a description of the syntax and meaning of valid telescope specification arguments.

IF_index - Default = The first sampled IF.

The index of the IF to start displaying from. One may also move between IFs with the '[' and ']' keys - see below.

CONTEXT

This command gives one the opportunity to see what corrections have been applied by selfcal and which integrations couldn't be corrected due to lack of data. It also gives you the opportunity to flag or unflag corrections, and to reset individual corrections.

Flagged corrections are plotted in blue. Good corrections are plotted in green. Each page shows the corrections of a single telescope from a single sub-array, in a given IF.

INTERACTIVE MODE

If the plot device has a cursor, then an interactive mode will be

invoked. In this mode, initially corrections for the first telescope of the observation, or the requested telescope, will be displayed. Thereafter keyboard keys, in conjunction with the cursor allow one to peruse the corrections of other telescopes in any order.

To get a list of the currently defined keys and what they do, place the cursor within the plot window and then hit the 'H' key on the keyboard.

EDITING

Unless the selfflag command has been used to suppress flagging, the selfcal comand automatically flags corrections that it has insufficient information to determine. These flags can either all be removed with the 'uncalib' command, or toggled individually in 'corplot'. To toggle the flag status of a correction in corplot, click the left-mouse-button, or keyboard key 'A' near the point that you wish to flag or unflag. The flags will be applied both to the correction, and to all visibilities affected by that correction.

RELATED COMMANDS

scangap	-	Change the time gap used to delimit neighboring scans.
uncalib	-	Remove selected parts of corrections.
radplot	-	Display visibility amplitude vs. UV radius.
projplot	-	Display visibility amplitude/phase vs. projected UV distance.
uvplot	-	Display the sampling of the UV plane.
vplot	-	Display observed and model visibilities.
tplot	-	Display the time-sampling of each telescope.
cpplot	-	Display observed and model closure phases interactively.

cpplot *nplot, triangle, IF_index, npage*

Interactive display and editing of observed and model closure phases.

EXAMPLE

1. Plot closure phases of triangles of telescope WSRT, using 10 plots per page.

```
0>cpplot 10, wsrt
Using default options string "efbm"
For help move the cursor into the plot window and press 'H'.
0>
```

2. Plot the closure phases of all triangles that include the baseline WSRT-BONN, using 10 plots per page.

```
0>cpplot 10, wsrt-bonn
Using default options string "efbm"
For help move the cursor into the plot window and press 'H'.
0>
```

PARAMETERS

- nplot** - (Default = 5 plots per page).
The maximum number of plots per page. The actual number of plots per page used will be restricted to the number of possible triangles that depend on the telescopes described in the 'triangle' argument.
- triangle** - (Default = "")
A triangle specification comprised of an optional sub-array index plus colon, followed by between one and three telescope names each separated by a hyphen. Read the antenna_names help topic for more details
1. If only one telescope is given, then all independent closure triangles with that telescope will be displayed at 'nplot' plots per page. There are up to $ntel*(ntel-1)/2 - (ntel - 1)$ of these.
 2. If two telescopes are given then all independent closure triangles with the resulting baseline are plotted at 'nplot' plots per page. There are up to $ntel-2$ of these.
 3. If three telescopes are provided then this completely defines a single triangle to be plotted. Thus only one plot per page will be displayed.

- IF_index - (Optional) Default = The first sampled IF.
The index of the IF to start plotting visibilities from.
- npage - (Optional) Default = 0 (with requests no page limit).
The maximum number of pages to plot when plotting to a non-interactive device.

CONTEXT

In the absence of noise and correlator errors, the only remaining sources of measurement errors are at the individual telescopes. In terms of phases, where the absolute phase at any telescope is irrelevant, there are thus only $N-1$ telescope based phase errors compared to $N(N-1)/2$ baseline visibility measurements. There are thus $N(N-1)/2 - (N-1)$ phase quantities that are un-corrupted by telescope based phase errors. Such quantities can be formed by summing visibility phases around a closed loop of three telescopes to cancel out telescope phase errors, and are known as closure phases.

Closure phases are thus independent of the phase errors caused by local atmospheric fluctuations over individual telescopes. They are also independent of phase-center position errors, because such errors translate to antenna dependant time-variable phase shifts.

In reality, noise, mismatched telescope bandpasses, correlator errors and limited precision mean that closure phases are less than perfect, but they are often still useful indicators of source structure if telescope based errors dominate visibility phases.

Closure phase is implicitly used in self-calibration to solve for and remove telescope based phase errors. This command may be profitably used to determine whether the closure phases are sufficiently well behaved that phase self-cal is possible.

CONFIGURATION VARIABLES

- vflags - This is a string variable (shared with vplot) that over-rides the default display attributes. It is particularly useful when displaying to a hard-copy device, where interactive changes to the default attributes can not be made.

By default vflags="". In this case cplot substitutes the equivalent of vflags="efmb", which means plot error bars, flagged data, the model, and break the plot into scans.

When vflags is not "", all toggled attributes are initially set to off, and the contents of the vflags string determine which of the attributes get turned back on.

HARD COPY MODE

If this command is invoked when the PGPLOT output device is a hard-copy device, such as a laser printer, then all triangles corresponding to the given triangle specification will be plotted. For example:

```

0>dev /ps
Attempting to open device: '/ps'
0>cpplot 5,ovro
Using default options string "efbm"
Page 01: Triangles of 1:OVRO
Page 02: Triangles of 1:OVRO
Page 03: Triangles of 1:OVRO
Page 04: Triangles of 1:OVRO
Page 05: Triangles of 1:OVRO
Page 06: Triangles of 1:OVRO
Page 07: Triangles of 1:OVRO
Page 08: Triangles of 1:OVRO
Page 09: Triangles of 1:OVRO
0>

```

This plotted all closure triangles of OVRO at 5 plots per page.

INTERACTIVE MODE

If the display device has a cursor, then an interactive session will be invoked. In this mode, one page of baselines is initially plotted and mouse-keys and keyboard keys are used to guide the command through plotting and data editing.

Once cpplot has been invoked, move the cursor into the display window and press the 'H' key on your keyboard. Whenever this is pressed during the cpplot session a list of key bindings will be displayed in the window from which cpplot was invoked.

Most keys are not case-sensitive, such that both 'H' and 'h' have the same effect.

On standard PGPLOT conforming devices with a mouse or other cursor control device, mouse buttons are equivalent to the following keyboard keys:

```

Left button   = 'A'   (This is the normal key for selecting positions).
Middle button = 'D'   (This is used to cancel incomplete select ranges).
Right button  = 'X'   (Use this to exit cpplot).

```

Note that on a windowing system, the cursor must be within the display window before pressing a keyboard key or mouse button will be noticed.

However, after pressing a key that needs more complex answers, such as the entry of the number of baselines to display or the entry of a new triangle specification, the cursor should be moved to the original text window where a prompt will be displayed to read your selection.

MOVING ABOUT BETWEEN TRIANGLES

- N - Display next set of triangles.
- P - Display preceding set of triangles.

Pressing 'N' in cplot causes the Next set of closure triangles that match the last triangle specification, to be plotted on a new page. The 'P' key works identically but shows the Previous set of triangles.

- T - Specify a new triangle specification from the keyboard (text screen)

When you press this, a prompt will be displayed on the text screen and you will be expected to enter a new triangle description using the same format as described above for the 'triangle' command-line argument. See the antenna_names help topic for a description of triangle specification arguments.

MOVING BETWEEN IFs

Visibilities are only displayed from one IF at a time. The identity of this IF is displayed above the collection of plots. To move onto the next or previous IF in which any channels were selected with the 'select' command, use the following keys:

-] - Move to the next sampled IF.
- [- Move to the previous sampled IF.

The title above the plots will change to indicate which IF is being displayed.

CHANGING HOW CLOSURE PHASES ARE DISPLAYED

The following keys toggle attributes on or off. The display will not be re-plotted immediately whenever one of these is toggled, so you can press a sequence of toggle keys before re-displaying with the new attributes. When the sequence is complete, simply press any key that is not in the list below, such as the RETURN key and the display will be updated with the new attributes.

- M - Toggle display of model visibilities
- F - Toggle display of flagged visibilities
- E - Toggle display of error bars
- B - Toggle breaking the plot into scans (where present)

By default, a scan is taken to be any set of integrations which is separated by more than an hour from its nearest neighbors. This can be changed with the 'scangap' command.

CHANGING THE NUMBER OF PLOTS PER PAGE.

S - Select number of sub-plots per page

You will be prompted for a number on the text screen. Your answer will be used to change the number of plots that appear per page on the screen.

CHANGING THE DISPLAYED UT RANGE

U - select UT range to be displayed (hit U twice for full range)

Press 'U' to request a change in the displayed UT range. You will then be expected either to press 'U' again to display the whole UT range, or to move the cursor to the start UT of the required range and press 'A' (left mouse button), then move the cursor to the end UT of the required range and press 'A' again. The plot will then be re-displayed over the new range and all subsequent plots will be displayed over this range. At any point before the UT range selection has been completed, pressing the 'D' key (middle mouse button) will quietly abort the selection.

CHANGING THE DISPLAYED CLOSURE-PHASE RANGE

Z - Select closure-phase range to be displayed (hit Z twice for full range)

Press 'Z' to zoom in on a selected closure-phase range. (Press it twice to revert to the full -180 to 180 degree range). You will then be expected either to press 'Z' again to display the whole closure-phase range, or to move the cursor to the first (top or bottom) closure-phase of the required vertical range and press 'A' (left mouse button), then move the cursor up or down to the end closure-phase of the required range and press 'A' again. The plot will then be re-displayed over the new range and all subsequent plots will be displayed over this range. At any point before the closure-phase range selection has been completed, pressing the 'D' key (middle mouse button) will quietly abort the selection.

CHANGING WHETHER REDUNDANT BASELINES ARE DISPLAYED

By default, for each reference telescope specification (eg. each reference baseline or telescope) that it plots triangles for, cplot

displays all unique triangles that include that spec. This is easy to understand and helps to highlight errors that are specific to the reference specification. However, it does result in you seeing each triangle more than once if you step through all reference specifications of a given type. This can be annoying if you have a very large number of triangles. For this reason cpplot also provides the option to only show each triangle once, by only showing triangles with telescopes that are in increasing sequential order with respect to the last antenna of the reference specification. Thus if your reference specification is a baseline, the final telescope is only allowed to be taken from the set of telescopes that follow the last telescope of the reference baseline, in the list of all telescopes.

0 - Toggle telescope ordering to only show non-redundant triangles.

EDITING DATA

Cpplot gives you the option to edit closure phases interactively. You can edit visibilities of a specific station, a specific baseline, or all three baselines of a selected triangle.

When displayed, flagged data appear as red '+' symbols, un-flagged data as green points and the model appears as a continuous light-blue line. In addition, if a visibility is flagged by a selfcal correction flag, but not by a normal visibility flag, it is displayed as a blue 'x' symbol.

Station editing:

If you wish edits to be applied to all visibilities of a specific station have cpplot display closure phases for just that station. This is done by specifying just that station name when you invoke cpplot, or via the 'T' key. The rationale behind this is that if a single station is responsible for closure phase errors (surprisingly this IS possible) you will probably only see this when plotting many closure triangles of that station.

Baseline editing:

If you wish edits to be only applied to visibilities of a specific baseline, have cpplot display closure phases for just that baseline. This is done by specifying just that baseline name when you invoke cpplot, or via the 'T' key. The rationale behind this is that if a single baseline is responsible for closure phase errors, you will probably only see this when simultaneously plotting many closure triangles of that baseline.

Triangle editing:

If you ask cpplot to plot closure phases of a single triangle, then

all edits are applied to each of the three baselines that make up the triangle. This mode of editing can also be selected when displaying closure phases of a specific station or baseline, by toggling between the default editing mode and triangle editing with the space bar.

' ' - (SPACE BAR) Toggles to and from triangle based editing.
 When this toggle is active, the mode title above the plot will change between "Triangle editing..." and the default editing mode.

Note that cplot displays a closure phase as flagged if any of the three visibilities that form it are flagged. Thus if you try to restore a flagged closure phase in baseline editing mode, then the closure phase will not appear to change state unless only that baseline was flagged.

It is also possible to further specify the scope of edits.

I - Toggle IF editing scope.
 W - Toggle spectral-line editing scope.

By default, edits are applied to all polarizations, spectral-line channels and IFs related to the selected visibility. To specify that edits only be applied to the displayed IF, press the 'I' key. To specify that edits be only applied to the channels that were specified with the 'select' command, press the 'W' key. The mode line above the collection of plots, describes which of these editing modes is in effect.

A - Flag or unflag the nearest closure sample to the cursor.
 (Left mouse button)

Pressing 'A' on the keyboard or the left mouse button near a displayed closure sample causes the status of the visibilities of that closure triangle to from un-flagged to flagged or vice versa. If flagged data are displayed, then toggling an un-flagged closure-phase will produce a red cross signifying its new flagged status. Otherwise the point will magically disappear, but can be seen subsequently if the display of flagged data is enabled.

C - Flag all data inside select box
 R - Restore data inside select box

If you wish to flag or un-flag an extended region of data, point by point editing is tedious. The Clip and Restore keys, 'C' and 'R' enable you to edit all points inside a square box selected with the cursor. To use this facility, press the respective key and then move the cursor to one corner of the required region and press 'A' (left mouse button). Then move the cursor to the opposite corner of

the required region and again press 'A'. The selected points will be re-displayed with their changed statuses.

ENDING A CPLOT SESSION

To end a cplot session, press 'X' or the right mouse button.
If substantial editing was performed in cplot then it is advisable to use the wmerge command to save the modified file before continuing to map the data.

RELATED COMMANDS

scangap	-	Change the time gap used to delimit neighboring scans.
radplot	-	Display visibility amplitude vs. UV radius.
projplot	-	Display visibility amplitude/phase vs. projected UV distance.
uvplot	-	Display the sampling of the UV plane.
vplot	-	Display observed and model visibilities.
tplot	-	Display the time-sampling of each telescope.
corplot	-	Display accumulated self-calibration corrections.

delmarker *ra, dec*

Delete the closest map marker to a given Right Ascension and Declination.

EXAMPLES

1. The following example finds the map marker that is nearest, Right Ascension 12:29:06.7 and declination 2:03:08.60, and deletes it.

```
O>delmarker 12:29:06.7, 2:03:08.60
```

PARAMETERS

- ra - The target Right Ascension expressed in hours, minutes and seconds of time.
- dec - The target Declination expressed in degrees, minutes and seconds of arc.

CONTEXT

Although markers can be created both with `mark_radec` and `mark_xy`, the locations of all markers are stored in Right Ascension and Declination. Thus to delete a marker you must identify it by this position. The current list of markers, with their equatorial positions, can be seen with the `wmarkers` command.

ASSOCIATED COMMANDS

- `mapplot` - Draw a map.
- `mark_xy` - Specify a map marker relative to the map center.
- `clear_markers` - Delete the current list of markers.
- `wmarkers` - List map marker commands to stdout or a file.

delwin

Delete all clean windows.

EXAMPLE

```
0> delwin
All clean windows deleted
0>
```

CONTEXT

Presently, the only way to delete specific clean windows is in mapplot. Delwin provides an alternate way to remove all windows without having to run mapplot.

RELATED COMMANDS

mapplot - Allows interactive adding and deletion of clean windows.
addwin - Allows command line addition of windows.
wwins - Save CLEAN windows to a text file.

edmodel *dovar*

Modify the established and tentative models with an external editor.

EXAMPLE

To edit all components of the established and tentative models type:

```
0>edmod
[... After exiting the editor]
A total of 2 model components were read from file edmod.scr
0>
```

The following demonstrates what happens if you inadvertently introduce a syntax error into the model file while editing:

```
0>edmod
rmodel: Error at field 2 on line 3 of file: edmod.scr
0>
```

To only edit model components with variable parameters:

```
0>edmod true
Partitioning the model into established and variable parts.
Adding 1 model components to the UV plane model.
The fixed established model contains 1 components (1.38117 Jy).
The variable part of the model contains 1 components (2 Jy).
A total of 3 model components were read from file edmod.scr
0>
```

PARAMETERS

```
dovar    -    Default = false
             If this argument is omitted or given as FALSE, then all
             all components of the tentative and established models
             will be presented to be edited. Alternatively, if it is
             given as TRUE, then only those components that have at
             least one component marked as variable will be presented.
```

CONTEXT

During model fitting the edmodel command can be used to add,delete or change the components of the established and tentative models using an external editor. This is usually used to change values of component parameters or to change which parameters are to be held constant.

Note that a 'v' post-fixed to a parameter makes it a free-variable for model fitting. Unfortunately, since the coordinate system used to fit the components to the data is not the same as that displayed in model

files it is not possible to fix many of the parameters individually. The following collective sets of parameters may be constrained or allowed to vary:

1. The absence or presence of a postfixed 'v' on the Flux parameter effects only the Flux parameter.
2. Placing a 'v' on either or both Radius and Theta implies that both are variable. It is not possible to individually fix them.
3. If the axial ratio is exactly 1.0 and there is only a 'v' postfixed to the Major axis parameter, then axial-ratio and phi will remain fixed during model fitting.
4. In general if the axial ratio is not exactly 1.0 then postfixing a 'v' to any of Major, Ratio or Phi, implies that they are all variable.

DEFAULT EDITOR

On UNIX systems the edmodel command defaults to invoking the UNIX 'vi' editor. If this is not the editor that you are used to, then you should override the default by assigning the EDITOR environment variable with the path of your editor. This must be done before entering difmap, preferably in your .cshrc or .login files.

At the shell level or in your .cshrc or .login files you would type one of the following lines to set emacs as your default editor:

Under Bourne compatible shells (eg. sh, bash, ksh):

```
EDITOR="emacs";export EDITOR
```

Under csh compatible shells (eg. csh, tcsh):

```
setenv EDITOR emacs
```

DETAILS

If 'dovar' is omitted or presented as FALSE, then the established model is first merged with the tentative model, then the tentative model is written to a temporary file, presented to your editor and on exit from the editor, read back into the tentative model. Since the established model is cleared by this operation, the next operation that requires a UV representation of the model will establish the model first.

If 'dovar' is given as TRUE, then Edmod first partitions the established and tentative models into two parts, a model established in the UV plane (this can take significant time), formed only from

fixed components, and a tentative variable model to be edited. It then writes the variable components of the model into a scratch file and invokes an external screen editor on it. When the user exits from the editor, the scratch file is read and deleted. If no errors were found in the file then the edited model supersedes the original tentative model.

RELATED COMMANDS

modelfit - Fit model components to UV data.

`fix_weights` *dofix*

Set whether amplitude corrections affect the visibility weights.

EXAMPLE

```
0>fix_weights
```

Allow amplitude corrections to visibility weights.

```
0>
```

```
0>fix_weights true
```

Do not allow amplitude corrections to visibility weights.

```
0>
```

```
0>fix_weights false
```

Allow amplitude corrections to visibility weights.

```
0>
```

PARAMETERS

`dofix` - (Default=false)

If false then amplitude corrections from `selfcal` and `resoff` will be applied to the visibility weights, treating them as $1/(\text{amplitude variance})$.

If false, then amplitude corrections will have not change the visibility weights.

CONTEXT

`Difmap` assumes that the weight associated with each visibility is equal to $1 / (\text{amplitude variance})$. If the variances used to calculate these weights were determined by looking at the scatter of the calibrated amplitudes, then they will have been affected by any miscalibration of the amplitudes. In this case, subsequent corrections to the amplitudes by self-calibration or the `resoff` command, should also be used to correct the weights to correct for the corresponding miscalibration of the scatter based variances.

On the other hand, if the amplitude variances used to calculate the weights were determined theoretically from the expected thermal noise, then they should will not be affected by any miscalibration of the amplitudes, so they should not have amplitude corrections applied to them by self-calibration or the `resoff` command.

RELATED COMMANDS

`selfcal` - Self calibrate for telescope phase (and amplitude) corrections.

`uncalib` - Remove selected amplitude or phase corrections.

`resoff` - Determine and apply a baseline based amplitude and phase correction.

clroff - Undo the effects of all resoff commands in this difmap session.
wtscale - Change the scale factor that mutiplies the raw visibility weights.

flag *baselines, allchan, start_time, end_time*

Flag a set of baselines over a selectable time range.

EXAMPLES

The following example flags all visibilities of the currently selected channels.

```
0>flag *
Flagging all baselines in the currently selected channels.
```

The next example flags all visibilities of sub-array 2 of the currently selected channels.

```
0>flag 2:
Flagging baselines of subarray 2 in the currently selected channels.
```

The next example flags just the visibilities of baselines that are associated with an antenna called rx3, (in the default subarray, which is 1). If rx3 were in sub-array 2, you would type 2:rx3.

```
0>flag rx3
Flagging baselines of antenna 1:RX3 in the currently selected channels.
```

The next example flags just a single baseline, but this time for all channels, irrespective of the current selection.

```
0>flag rx3-rx5, true
Flagging baseline 1:RX3-RX5 in all channels.
```

The final example flags all baselines in just the selected channels, between 10:45 and 10:50 on the 20th of November 2001.

```
0>flag *, false, 20-nov-2001:10:45, 20-nov-2001:10:50
Flagging all baselines in the currently selected channels.
```

PARAMETERS

baselines	-	The set of baselines to flag, specified by one of the following forms:
*	-	All baselines in all sub-arrays.
sub:	-	All baselines in the sub-array specified by the number 'sub'.
sub:ta	-	All baselines of the telescope named 'ta', in sub-array 'sub'.

sub:ta-tb - The single baseline of the
telescopes named 'ta' and 'tb'.

In the above, if sub: is omitted from the last
two specifications, 1: is quietly substituted.

allchan - Default = false
If true, flag visibilities of the specified
baselines over all channels and IFs. Otherwise
only flag visibilities in the channels that
were last selected with the 'select' command.

start_time - Default = ""
If not omitted, or specified as an empty string,
visibility editing will start from the start of
the observation. Otherwise this argument should
specify the date and time at which editing
should start, specified in the following format.

dd-mmm-yyyy-mm:hh:mm:ss

Where dd is the day of the month 1-31, mmm is
a standard 3-letter abbreviation of a month name,
and yyyy is a fully qualified year (ie. 2003, not
03). Trailing components of the time that follows
the date can be omitted. The hour part, hh, is
an hour number 0-23. The minute part, mm and the
seconds part, ss go from 0-59.

end_time - Default = ""
If omitted, or specified as an empty string,
visibilities from the start time to the end of
the observation will be edited. Otherwise this
argument should specify the date and time at
which editing should end, using the same format
as described for the start_time argument.

CONTEXT

In most cases it is best to edit visibilities visually, using a
visibility inspection command like vplot, tplot or radplot. However
sometimes it is desirable to be able to perform editing
from a script, without any user interaction. The 'flag' command
facilitates this.

RELATED COMMANDS

unflag - Unflag a set of baselines over a selectable time range.

get prefix_name

Restore UV data, models and windows from files.

EXAMPLE

0>get 3c345

Attempting to open FITS file: 3c345.uvf

... details of UV FITS file contents

A total of 94 model components were read from file 3c345.mod

rwins: Read 3 windows from 3c345.win

0>

PARAMETERS

prefix_name - All the required files must have this as there prefix followed by the standard suffixes listed below.

STANDARD SUFFIXES

To the prefix name that you enter the following extensions are appended to indicate the file type:

.uvf - The suffix of the UV-FITS data file.

.mod - The suffix of any model file.

.win - The suffix of any windows file.

CONTEXT

Before leaving difmap it is important to save all processed data unless the data is to be discarded. The 'save' command provides a convenient method for saving all relevant data with a single command.

The 'get' command provides the matching facility for reading the files back into difmap.

Note that it doesn't read the FITS restored map file.

IMPORTANT

The get command does not restore the running parameters (such as the UV weighting and map size) to the states that they held when the save command was called - it only restores the data files. To do this don't use the 'get' command directly. Instead execute the .par file command file that was produced with the save command (in the above example this file would be called 3c345.par). This file contains all the commands necessary to restore both the data files and the state of all running parameters. See help on the 'save' command for more details.

RELATED COMMANDS

save - Save UV data, models, windows, the restored map and a command
file to use to restore the running parameters.

rwins - Read previously saved CLEAN windows from a text file.

rmodel - Read a CLEAN model from a VLBI model file.

gscale dofloat

Self calibrate to determine overall telescope amplitude corrections.

EXAMPLES

0>gscale

Computing UV representation of latest clean model (147 components)

The established model now contains 147 components and 7.0511 Jy

Performing overall amplitude self-cal

Correcting IF 1.

Amplitude normalization factor in sub-array 1: 1.01723

Telescope amplitude corrections in sub-array 1:

BONN	0.97	WSRT	0.95	JBNK1	0.99	BOLOGNA	1.01
TORUN	1.00*	HART	1.01	HSTK	1.01	NRAO	1.02
VLBA_KP	0.96	VLBA_PT	1.00*	VLBA_LA	0.96	VLA27	0.98
OVRO	0.96						

Fit before self-cal, rms=0.498946Jy sigma=3.396564

Fit after self-cal, rms=0.438911Jy sigma=3.160276

Reinstating the default IF.

0>

All but the first line are responses from the 'gscale' command.

Some of the reported corrections are post-fixed with a '*' symbol.

This means that no solution was possible for that telescope.

These corrections are not 1.0, only because of amplitude normalization.

Without amplitude normalization (dofloat=true) the same data-set gave:

0>gscale true

Performing overall amplitude self-cal

Correcting IF 1.

Telescope amplitude corrections:

BONN	0.97	WSRT	0.95	JBNK1	0.98	BOLOGNA	1.01
TORUN	1.00*	HART	1.02	HSTK	1.02	NRAO	1.01
VLBA_KP	0.96	VLBA_PT	1.00*	VLBA_LA	0.96	VLA27	0.98
OVRO	0.96						

Fit before self-cal, rms=0.500856Jy sigma=3.412605

Fit after self-cal, rms=0.365222Jy sigma=2.861346

Reinstating the default IF.

0>

PARAMETER

dofloat - If this parameter is omitted or given as false, the telescope corrections are normalized to maintain the current total flux. Alternatively, if dofloat=true then the telescope amplitudes will be allowed to float and the observed flux may be changed if not sufficiently constrained.

CONTEXT

When one or more telescopes are miscalibrated by a constant factor, this command may be used to solve for the correct telescope amplitudes. In order to do this a reasonable model of the source structure must have been provided. Such a model may be provided using CLEAN or using a model from the VLBI modelfit program read into this program via the 'rmodel' command.

This command should be used with caution. For instance, comparison of the current model, in 'vplot', and the observed visibilities may appear to indicate a number of consistently high baselines. This often indicates data that hasn't been cleaned deeply enough. In particular, check to see if just the shortest baselines are too high by examining the plot of visibility amplitude versus UV radius given by the radplot command. Such missing flux implies the existence of un-cleaned low surface brightness extended emission that hasn't been cleaned yet. Try switching to natural weighting (see help uvweight) and/or apply a UV-taper before cleaning (see help uvtaper). This may reveal such emission, which may then be cleaned.

If it turns out that artifacts due to poor amplitude calibration, prevent you from cleaning low surface-brightness extended emission then you should use the selftaper command to weight down the significance of the short baseline visibilities before calling gscale (see help selftaper). Failure to do this will make gscale attempt to force the visibility data to fit the incomplete model at short UV radii.

Amplitude self-calibration relies implicitly on the existence of closure amplitudes. To form such a closure amplitude requires that there be un-flagged visibilities on that lie baselines in closed arrays of at least 4 telescopes. Gscale attempts to detect visibilities that do not satisfy these conditions and exclude them from the fit. How it does this, and whether these visibilities are then flagged, is controlled by the selfflag command. See help selfflag for more details.

NB. Observations that contain multiple sub-arrays and/or multiple IFs will produce nif * nsub lists of telescope corrections, instead of the single list given the examples above.

RELATED COMMANDS

selfflag - Used to control the fate of un-correctable data.
selfcal - General self-calibration command.
selftaper - Used to down-weight short baselines during selfcal.
selflims - Used to set limits to amp and phase corrections in selfcal.
startmod - Phase selfcal to a starting model then discard the model.

header

List useful parts of the observation header on one's terminal.

imstat *statistic*

Return image plane map statistics.

EXAMPLE

```
0>print imstat(rms), imstat(mean), imstat(noise)
0.237115 0.00853315 0.0056867
0>
0>print imstat(bmin),imstat(bmaj),imstat(bpa)
2.62014 20.9194 -10.0713
0>
```

PARAMETERS

statistic - The name of the statistic wanted. Recognised names are:

- rms** - The RMS flux density in the map (Jy/beam).
- mean** - The mean flux density in the map (Jy/beam).
- noise** - The noise level in the map estimated from the visibility and gridding weights. (Jy/beam). Note that this number is meaningless unless the visibility weights are the reciprocal of the amplitude variance. See help wtscale for details.
- bmin** - The minor axis FWHM of the beam size estimated from the data during the last invert (map units, see help mapunits).
- bmaj** - The major axis FWHM of the beam size estimated from the data during the last invert (map units, see help mapunits).
- bpa** - The position angle of the beam major axis, estimated from the data during the last invert (degrees North -> East).
- dx** - The width of a map pixel, expressed in the currently selected mapunits.
- dy** - The height of a map pixel, expressed in the currently selected mapunits.
- du** - The U-axis width of a pixel in the UV plane, expressed in wavelengths.
- dv** - The V-axis height of a pixel in the UV plane, expressed in wavelengths.
- nx** - The number of pixels along the X axis. Note that, because of the way that CLEAN works, mapplot only displays the central nx/2 of these pixels.
- ny** - The number of pixels along the Y axis. Note that, because of the way that CLEAN works, mapplot only displays the central ny/2

of these pixels.

CONTEXT

This function provides access to statistics relating to the latest map. The return value may be used in expressions, and may thus be used in automatic mapping procedures.

SIDE EFFECTS

The map will be re-inverted if not currently up to date.

RELATED COMMANDS

uvstat - Return UV plane visibility statistics.
peak - Return details of the min,max or absolute max flux in a map.
mapvalue - Return the value of the map pixel nearest to a given coordinate.

invert

Grid selected UV data and Fourier invert it to a (residual) dirty map

EXAMPLE 1

```
0>invert
Inverting map and beam
Estimated beam: bmin=0.7503 mas, bmaj=1.8830 mas, bpa=-10.9 degrees
0>
```

The two lines without prompts are the response from the invert command. The second of these is an estimate of the restoring beam size.

EXAMPLE 2

```
0>mapsize 256,2
0>invert
Inverting map and beam
Your choice of large map pixels excluded 49.3% of the data.
The x-axis pixel size should ideally be below 1.025 milli-arcsec
The y-axis pixel size should ideally be below 3.466 milli-arcsec
Estimated beam: bmin=9.189 mas, bmaj=23.87 mas, bpa=-11.54 degrees
Estimated noise=6.17991 mJy/beam.
0>
```

In this case 49% of the visibilities represented higher spatial frequencies than could be Nyquist sampled by the coarse pixel spacing of the chosen map grid. These visibilities were thus excluded from the map produced by invert. The third and fourth lines of output show the maximum pixel spacings that can be accommodated in RA and Dec without lossing data. One could satisfy both sizes by using the mapsize command to select a new square map pixel size of 1 arcsecond, or by using the mapsize command to select a rectangular pixel size of 1.0 x 3.0 arcseconds.

CONTEXT

Invert is the command that takes UV data and transforms it to a dirty map and a dirty beam. Normally 'invert' is run for you by any commands that operate on the residual map, such as:

```
clean    - CLEAN a residual map.
restore  - Restore a residual map.
mapplot  - Display a residual map or beam.
```

It is only run by these commands when the current dirty map and/or beam has been invalidated by modifications to the UV data. The beam often doesn't need to be recomputed under these circumstances. When it

does need to be re-computed then 'invert' will also report a corresponding estimate of the new beam dimensions.

RESIDUALS

If an established clean model exists then this will be subtracted from the observed visibilities while gridding. The result is to produce a dirty residual map approximately the same as the one left after clean, but devoid of any aliasing from the removed components. If a non-residual dirty map is desired, then discard the established clean model by typing 'clrmod true' and re-invert.

RUNNING PARAMETERS

Although the command itself takes no arguments, the procedure that it runs takes a number of parameters. These are set and changed through a number of related difmap commands. Reasonable defaults are set for most of these, but you will be required to at least read a UV data file, using the 'observe' command and to specify a map grid, via the 'mapsize' command before first using invert. The rest of the commands offer a choice of visibility weighting and selection schemes which effect how the data are gridded on the UV plane. These commands are listed in the RELATED COMMANDS section below, and each command is discussed separately in its own help file.

GRIDDING

Visibilities for each IF of an observation are combined in the UV plane during gridding. If each IF has a different frequency then this implements a crude form of multi-frequency-synthesis, however currently no provision is made to correct for spectral-index effects, so if you don't think that the source structure and flux are constant between the multiple frequencies then you are advised to split the IFs in AIPS and map them separately.

Also note that if a range of spectral-line channels is selected via the difmap 'select' command, that a vector average of visibilities over those channels is formed before gridding. Thus different channels are not gridded separately and if the UV coordinates of visibilities change appreciably over the selected range then some degree of bandwidth smearing will be apparent.

UV SAMPLING CONSTRAINTS

In order to avoid aliasing and to avoid undersampling the image plane, the UV grid must be no more than half filled along either the U or the V axes. If the current map cell-size is too large to accomodate the range of U or V coordinates along the respective axes of the UV grid, then 'invert' will exclude visibilities that lie outside the Nyquist

UV limits from the final map. The accompanying warning messages provide directions for remedying this situation with the `mapsize` command. Note that these are no more than warnings. There are times when using large pixels is the only practicle way to view a large enough field.

If the Nyquist limits are exceeded along both the RA (x) axis and the Dec (y) axis, then the `invert` command will report corresponding limiting pixel sizes for the two axes separately. In such cases you could either take the smallest of the reported sizes and use the `mapsize` command to select square pixels, or you could use the `mapsize` command to select rectangular pixels to match both constraints. See 'help mapsize' for details.

SIDE EFFECTS

The latest CLEAN model will be added to the established model component list and model visibilities. If this is undesirable then use the '`clrmod`' command to clear the latest CLEAN model before inverting.

RELATED COMMANDS

`mapsize` - Specify the dimensions of subsequent maps.
`uvweight` - Selects combinations of natural, uniform, uncertainty, and radial weighting.
`uvtaper` - Apply a gaussian taper to the weights.
`uvrange` - Constrain the range of UV radii within which to select visibilities for gridding.
`uvzero` - Set a zero-UV-spacing flux.

keep

Establish the latest CLEAN model and compute its UV representation.

EXAMPLE

```
0>keep
```

```
Computing UV representation of latest clean model (91 components)
```

```
The established model now contains 182 components and 8.18052 Jy
```

```
0>
```

All but the first line are responses from the 'keep' command.

CONTEXT

During mapping, two separate CLEAN models are kept. These are the "established model" and the "latest clean model". The former is subtracted from the visibilities during invert and is kept both as a model component list AND as model visibilities in the UV plane. The latter - the latest clean model - is the model that has been accumulated by CLEANing since the last time that 'keep' was run. Also if a model is read from an external file using the 'rmodel' command then it replaces the latest clean model and deletes the established model. The latest clean model is added in to the established model and converted to its UV representation every time that the 'keep' command is called. There is rarely a need to do this explicitly, since commands that require the UV representation of the complete model, call 'keep' for you. Such commands include 'invert', 'selfcal' and 'vplot'. However, if you just keep cleaning the residual map without calling any of these other commands then to remove aliased artifacts it may be desirable to call the 'keep' command explicitly once in a while.

A separate UV plane representation of the established model is kept for each IF, in the uvmodel.scr scratch file, and automatically paged in and out by difmap as required.

RELATED COMMANDS

clrmod - Used to clear one or more of the established, tentative and continuum models.

`loglevs min, max, base`

Set logarithmic contour levels for `mapplot`.

EXAMPLE

```
0>loglevs 0.5
The new contour levels are:
-0.5 0.5 1 2 4 8 16 32 64
0>
```

The second and third lines are output lines from `loglevs` and show the resulting levels assigned to the `levs` variable.

PARAMETERS

```
min - The lowest (positive) level to include. The levels list starts
      with -min,min,... ie. The first positive level is also used as
      the only negative value.

max - (Optional: Default=100)
      This is the highest (positive) level to include in the levels
      listing. Thus: loglevs 0.5,2 produces -0.5,0.5,1,2 .

base - (Optional: Default=2)
       The scale factor between subsequent levels in the listing.
       Thus: loglevs 0.5,2,1.5 produces -0.5 0.5 0.75 1.125 1.6875 .
```

CONTEXT

The full dynamic range of a contour plot is often most cleanly represented by an exponentially increasing set of contour levels (conventionally called logarithmic levels).

The contour levels used by subsequent calls to the `mapplot` command are defined in a user-variable called `levs`. One way to set these levels is to directly assign them:

For example:

```
0> cmul=0
0> levs=-1,1,2,4,8,16,32,64
```

Defines a set of percentage contour levels starting at -1% and increasing exponentially up to 64%.

It is tedious to have to type out the levels explicitly in this way. The `loglevs` command provides a more convenient means to assigning "logarithmic" levels to the `levs` variable. Thus the

above example could be achieved by typing:

```
0> cmul=0
0> loglevs -1
```

RELATED COMMANDS

```
mapplot - Display map and beam images.
xyrange - Set the boundaries of the sub-image to be displayed in mapplot.
loglevs - Set logarithmic levels.
mapcolor - Select mapplot colormap type and transfer function.
mapfunc - Change the transfer function used in mapplot.
beamloc - Change when and where mapplot plots the clean beam ellipse.
```

`map_to_rad` *distance*

Convert an image plane distance from map units to radians.

EXAMPLE

The following example converts 10 arcminutes to radians.

```
0>mapunits arcmin
Distances in the map plane now have units: arcmin.
Distances in the UV plane now have units: kilo-wavelengths.
0>print map_to_rad(10)
0.00290888
0>
```

PARAMETERS

`distance` - A distance in the map plane, expressed in the currently selected map units.

CONTEXT

The 'mapunits' command allows the user to specify which units should be used both for specifying distances, such as the cell sizes, of maps, and for printing such distances.

The 'map_to_rad()' function allows the user to convert a distance in these units, into radians.

RELATED COMMANDS

`rad_to_map` - Convert an image plane distance from radians to map units.
`uv_to_wav` - Convert a UV plane distance from UV units to wavelengths.
`wav_to_uv` - Convert a UV plane distance from wavelengths to UV units.

mapcolor *type, contrast, brightness*

Select the colormap type, brightness and contrast to use in mapplot.

EXAMPLES

1. To disable psuedo-color plotting.

```
0>mapcolor none
Mapplot colormap: none, contrast: 1 brightness: 0.5.
0>
```

2. To select a grey-scale color map.

```
0>mapcol grey
Mapplot colormap: grey, contrast: 1 brightness: 0.5.
0>
```

3. To select the current default color colormap:

```
0>mapcolor color
Mapplot colormap: rainbow, contrast: 1 brightness: 0.5.
0>
```

4. To change the default colormap to the AIPS TVFIDDLE colormap and select it for the next mapplot:

```
0>mapcolor aips
Mapplot colormap: aips, contrast: 1 brightness: 0.5.
0>
```

PARAMETERS

```
type - The type of colormap to use, selected from:
      none - Turn off psuedo-color plotting.
      grey - Use a grey-scale ramp [initial default].
      color - Use the default RGB colormap.
      aips - The AIPS TVFIDDLE colormap.
      heat - Flame colored colormap.
      ramp - This is a three ramp colormap, based on
              one in Iraf's ximtool.
      red - Red ramp.
      green - Green ramp.
      blue - Blue ramp.
```

```
brightness - The brightness of the color bar between 0 and 1.
              With the gray-scale colormap, the image will be
              completely black for brightness <= 0, and completely
              white at brightness >= 1.
```

The initial default is 0.5, which places the center of the colormap at the center of the color bar.

contrast - The contrast of the colormap, expressed as the reciprocal of the width of the colormap on the displayed color bar. The initial default of 1.0 spreads the colormap over the length of the color bar. Higher contrasts compress the colormap, while lower contrasts expand it.

CONTEXT

Mapplot can display maps using contours, and/or grey-scale or psuedo-color. The colormap command provides a means of selecting these options prior to running mapplot. There are three classes of colormap that you can choose:

none - This turns off grey-scale and color plotting entirely.

grey - This selects the default grey-scale colormap.

color - All colormaps except those named above, are of class 'color'. Specifying 'color' to the colormap command selects the last colormap of class 'color' that was previously selected. Thus after selecting 'aips' once, either interactively in 'mapplot', or via the mapcolor command, and then subsequent requests for 'color' install the 'aips' colormap. Other 'color' color-maps are listed above under the explanation of the 'type' argument.

You also have the option of specifying how the colormap will be mapped onto the mapplot image intensity. By default, the contrast is set to 1 and the brightness to 0.5. This fits the colormap exactly within the color bar plotted under the mapplot image. The center of the colormap can be moved up or down the color bar by changing the colormap brightness. This is best done interactively in mapplot, but can also be set on the command line with the mapcolor command. The width of the colormap compared to that of the color bar can also be changed by altering the contrast. High contrast equates to a colormap that occupies only a small fraction of the color bar. Low contrast equates to a colormap extent that exceeds the width of the color bar, such that only a small range of its colors are visible. Negative contrasts act the same as positive contrasts except that the colormap is reversed. Again, the contrast is best set interactively in mapplot, but can be preset with the 'mapcolor' command.

Note that to completely flip the sense of a color map you should first run the mapcolor command without arguments, to see what the current brightness and contrast are. Then run the mapcolor command and

substitute -contrast for the original contrast, and 1-brightness for the original brightness.

To see what the current colormap settings are, invoke the colormap command with no arguments.

Note that changes to the colormap selection, brightness and contrast within mapplot remain in place after you exit mapplot. It is thus possible to set these parameters interactively in mapplot for subsequent mapplot plots. In particular, this is invaluable for making hard copies in color after setting the desired colors on an interactive device.

Also see 'help mapfunc' for details on how to specify the transfer function and data range to plot.

Limitations

When using this command note that:

1. Some devices are not capable of displaying color. In such cases RGB colormap types will be translated into grey-levels. This is device dependant and the results may not be what you expect.
2. Some devices can't even display different levels of grey. On such devices a random-stipple pattern is used to simulate different grey-levels.
3. If you are using an old version of PGPLOT that does not have the PGIMAG call, then only grey-scale colormaps are generally available on hard-copy devices.

RELATED COMMANDS

```
mapplot  -  Display map and beam images.
xyrange  -  Set the boundaries of the sub-image to be displayed in mapplot.
loglevs  -  Set logarithmic contour levels for use in 'mapplot'.
mapfunc  -  Change the transfer function used in mapplot.
mapplot  -  Display map and beam images.
beamloc  -  Change when and where mapplot plots the clean beam ellipse.
```

mapfunc *function, vmin, vmax*

Change the transfer function and data range displayed in mapplot.

EXAMPLES

To select a linear transfer function:

```
0>mapfunc linear
Mapplot transfer-function = linear, Data range = data min -> data max.
0>
```

To select a logarithmic transfer function, and limit the range of fluxes displayed:

```
0>mapfunc log, peak(flux,min), peak(flux,max)/2
Mapplot transfer-function = log, Data range = -0.103472 -> 0.162955 Jy.
0>
```

To select the whole available data range:

```
0>mapfunc linear,0,0
Mapplot transfer-function = linear, Data range = data min -> data max.
0>
```

PARAMETERS

```
function - The transfer function to use, selected from:
            linear - Linear transfer function.
            log    - Logarithmic transfer function.
            sqrt   - Square-root transfer function.

vmin      - The minimum map intensity to be plotted (Jy).

vmax      - The maximum map intensity to be plotted (Jy).
            To select the whole data range, set vmin and vmax
            both to zero.
```

CONTEXT

Mapplot displays images using contours and/or grey-scale/psuedo-color plots. When grey-scale or psuedo-color are selected (see help mapcolor), mapplot defaults to mapping the whole range of data in the map or beam, linearly onto the current colormap. In high dynamic range images with bright features this has the tendency of making low brightness features difficult to see. In such cases it is often advantageous to take the log or square root of the image values, and/or reduce the maximum intensity plotted. The 'mapfunc' command allows one to do this.

Note that mapplot also allows one to interactively change the transfer function (but not the data range).

Also note that older versions of PGPLOT did not allow one to change the transfer function, and if you are using such a version, all transfer functions will appear linear.

RELATED COMMANDS

mapplot - Display map and beam images.
mapcolor - Change the colormap, brightness and contrast used in mapplot.
xyrange - Set the boundaries of the sub-image to be displayed in mapplot.
loglevs - Set logarithmic contour levels for use in 'mapplot'.
beamloc - Change when and where mapplot plots the clean beam ellipse.

maplot *beam_map_cln, domodel*

Alternative name for the mapplot command - see help mapplot.

mapplot *target, domodel, usecursor*

Display the dirty beam, residual or restored map and modify CLEAN windows.

EXAMPLE 1

```
O>mapplot beam
Graphics device/type (? to see list, default /NULL): /xw
Move the cursor into the plot window and press 'H' for help
O>
```

In the above example, no PGPLOT display device had previously been opened so the second line requests the PGPLOT display device. Once this has been entered, future uses of mapplot will not require the device name to be entered. The plot device can be changed with the "device" command.

```
O> mapplot map,true
```

This plots the residual map overlayed with symbols for each model component.

PARAMETERS

- | | |
|-----------|---|
| target | - (Default = map)
The type of map or beam to display, from:
map - displays the map,
beam - displays the dirty beam,
cln - displays the restored map (after
restoring when necessary).
pmap - Display a contour/grey-scale residual map
of the currently selected polarization,
overlayed by polarization vectors formed
from the residual Q and U maps.
pcln - Display a restored contour/grey-scale map
of the currently selected polarization,
overlayed by polarization vectors formed
from restored Q and U maps. |
| domodel | - (Default = false)
If true then any model components that have been accumulated will be plotted symbolically. Delta components are denoted by '+' symbols, and others by their elliptical outline. Positive components are drawn in green and negative components in red. Variable components are plotted in yellow (orange if negative). |
| usecursor | - (Default = true)
If this argument is true, and the plot device has a |

cursor, then the cursor will be displayed for the user to use to interact with the image.

If this argument is false, or the plot device does not have a cursor, then as soon as the image has been plotted, new commands can be entered at the difmap prompt.

RUNNING PARAMETERS

There are a number of auxiliary parameters that don't appear in the argument list. Some of these parameters exist as variables, while others are commands. All are listed below.

False color plotting

=====

Commands:

- mapcolor - The 'mapcolor' command allows one to pre-select the color table, brightness and contrast under which mapplot maps will be plotted.
See 'help mapcolor' for further details.
- mapfunc - The 'mapfunc' command allows one to select the transfer function with which the maps in mapplot will be plotted and to specify the range of intensities to be displayed.
See 'help mapfunc' for further details.

contour plotting

=====

Variables:

- docont - If docont==true then future use of 'mapplot' to plot the restored map will include a contour representation of the clean map. This parameter is ignored when plotting the beam and residual maps if false-color has also been selected. Docont is initially defined as docont=true.
- levs - A 1-D array of contour levels. If this has not been set then the default percentage levels of -1,1,2,4,8,16,32,64 are used for the map and -64,-16,-4,4,16,32,64 for the beam. If valid contour levels are given, they are treated as percentage levels if 'cmul' is <= 0, or as absolute levels pre-scaled by 'cmul' otherwise.

Commands:

- loglevs - This command provides a short cut to setting the 'levs' variable described above. It sets the contour levels to

a logarithmic ramp. See 'help loglevs' for further details.

- polvec - This command configures the attributes used when displaying polarization vectors.
- mark_radec - Add an optionally labelled marker to be displayed on subsequent maps at a given Right Ascension and Declination.
- mark_xy - Add an optionally labelled marker to be displayed on subsequent maps at a given X,Y position relative to the map center.

Note that only levels that appear in the displayed area are presented in the list of levels that is written beneath the plot.

Restricting the plotted display range

=====

By default, the whole available map or beam area is displayed when you invoke mapplot. To change this behavior use the xyrange command (see help xyrange) to select the required sub-image for subsequent calls to mapplot. Alternatively, the displayed area can be changed interactively with the cursor and optionally saved for later calls to mapplot - see below.

USE OF THE CURSOR

If your display device has the capability to display a graphics cursor, then this will be displayed and you will be able to (among other things) add or remove CLEAN windows. The options are selected by pressing any of the keyboard or mouse keys given below (the cursor must be within the display window). In particular, to get a list of key bindings while running mapplot, hit the 'H' key on your keyboard and the list will be displayed in your terminal window.

NB. If you have a mouse, the following keys are bound to the mouse buttons.

- A - Left mouse button.
- D - Middle mouse button.
- X - Right mouse button.

Mapplot cursor key Bindings

=====

- Key 'A' (left-button) Set a window vertex. (ie move the cursor to where the start vertex is required and press 'A', then move the cursor to the desired position of the opposite corner and again press 'A' and the new window will be drawn.
- Key 'D' (middle-button) Delete the CLEAN window which has one of its corners closer to the cursor than any other

	window. This key may also be used to cancel a partially complete window selection.
Key 'C'	This switches from gray-scale to a pseudo-color plot.
Key 'G'	Re-plots gray-scale after 'C' has been pressed.
Key 'S'	Display statistics of the region of the map inside a CLEAN window. This includes the total flux only if the displayed map is a restored map. The window used is the one that has one of its corners closer to the cursor than any other window.
Key 'V'	Report value and coordinates of the pixel that lies immediately under the cursor. When polarization vectors are being displayed, this also shows the polarization angle and intensity of the pixel.
Key 'Z'	Select a sub-image to be displayed by pressing the 'A' key at two opposite corners of the required area. To cancel sub-image selection, press 'D' at any point during the selection. To zoom out back to the full display area, press 'Z' a second time.
Key 'K'	Pressing this key will tell mapplot to remember the current sub-image boundaries on the next call to mapplot.
Key 'f' (lower case)	Pressing this key over a particular position in the plot changes the brightness and contrast of the displayed colormap. The X-axis position of the cursor controls the brightness, increasing from left to right. The Y-axis position controls the contrast, with 0 contrast in the middle of the plot, positive contrasts below this and negative contrasts above.
Key 'F'	This resets the colormap brightness and contrast to the defaults: brightness=1.0, contrast=0.5.
Key 'c' (lower case)	This changes the colormap from grey-scale to the default color colormap.
Key 'C'	This allows you to select a colormap by name. At the prompt, type in the name of the required colormap, or enter nothing to leave things as they are.
Key 'T'	Re-display the plot with a specified transfer function (log,linear,sqrt). At the prompt, enter the name of the transfer function required.
Key 'L'	Re-display the current plot.
Key 'm' (lower case)	This toggles whether to overlay a symbolic representation of the model on the map.
Key 'M' (upper case)	This toggles whether to overlay a symbolic representation of just the variable components of the model on the map.

Key 'N'	Initiate creation of a new model component - see later.
Key 'R'	Remove the component that has the closest center to the cursor. This deletes the component from the established or tentative model of the observation. Note that if you plan to delete many components a faster way is to use the difmap 'winmod' command.
Key 'U'	Remove the marker nearest the cursor.
Key 'H'	Display list of key bindings.
Key 'X' (right-button)	Exit from 'mapplot'.

DISPLAYING RESTORED MAPS

There are two ways of having the map restored and plotted. You can either call the 'restore' command explicitly then type 'mapplot cln', or simply type 'mapplot cln'. For example:

```
0>restore 1.2,2.5,-32
0>mapplot cln
```

or

```
0> mapplot cln
```

In the first case a non-default restoring beam size was required so restore was invoked explicitly. In the second case restore is called by mapplot, with no arguments.

Note that if you typed:

```
0>restore 1.2,2.5,-32
0>mapplot
```

then since mapplot defaults to plotting the residual map when given no arguments, it will re-invert the data and display the residual map - take care!

DISPLAYING POLARIZATION VECTORS

If you have a observation that either contains the Q and U stokes parameters, or the LR and RL polarizations, mapplot is capable of displaying linear polarization as short vectors overlaid on the grey-scale/contour image of the current map. The length of the vectors reflects the polarized intensity in the underlying pixel, and the angle of the vector indicates the associated polarization angle. A detailed example of how to plot polarization maps can be read in the polarization help page. Basically you need to switch to multi-model

mode, so that models associated with the maps that you make of Q and U are kept separately by difmap, and used by the mapplot command to remake these maps behind the scenes. One makes individual maps of Q, U and the polarization that you want to display under the polarization vectors, configure the properties of the polarization vectors using the 'polvec' command, and type:

```
mapplot pcln
```

This displays the polarization vectors overlaid over the map of the currently selected polarization.

Note that a polarization angle of zero points towards the top of the map, and increases as the vector rotates from North through East (ie. anticlockwise in mapplot). You can examine the polarization angle and intensity of any given pixel in the displayed map by pressing the 'v' key while the cursor is over the pixel of interest.

MARKING AND LABELLING FEATURES IN MAPS

Difmap maintains a list of map markers that can be displayed on maps. Markers are added to this list with the 'mark_radec' and 'mark_xy' commands. The former command is useful for labeling known sources by their Right Ascension and Declination, whereas the latter, which allows one to mark a given X,Y position in a map, is useful for labeling newly discovered features, first seen in difmap maps.

Individual markers can be removed from the list, either interactively in mapplot, with the 'u' key, or non-interactively using the 'delmarker' command. The whole list of markers can also be deleted using the 'clear_markers' command.

Note that although mapplot doesn't provide an interactive way to add markers (there are too many parameter for this to be automated with a single mouse-button click), when defining a new X,Y marker from the keyboard, the position of the feature to be labelled can first be found interactively in mapplot using the 'v' key.

SIDE EFFECTS

Note that changes to the colormap selection, brightness and contrast within mapplot remain in place after you exit mapplot. It is thus possible to set these parameters interactively in mapplot for subsequent mapplot plots. In particular, this is invaluable for making hard copies in color after setting the desired colors on an interactive device.

When displaying a residual map or dirty beam the UV data will be re-inverted if recent changes to the grid weighting or modifications

to the data itself, eg. editing, make the current residual map out of date.

When plotting the restored map (using the `cln` option) the map will be restored if an up-to-date restored map has not already been created with the `'restore'` command. In this case `'restore'` will be called with no arguments.

If new components are interactively added, or deleted then the map will be marked as out of date and will be re-inverted the next time that it is required.

MODEL COMPONENT CREATION AND DELETION

Maplot can be used to interactively add and/or remove variable model components pertinent to subsequent model fitting. Other forms of variable model component editing should be performed with the `difmap` `'edmod'` command. The changes to the map itself are NOT displayed as model components are added and removed. This would take too long. Only changes to the symbolic outlines of components are displayed.

The syntax used to create a component is as follows:

Press `'N'` to initiate description of a new component.

Press `'A'` (left-mouse-button) between 0 and 3 times, to describe the position and dimensions of the new component.

Press `'N'` to install the component, or press `'D'` (middle-mouse-button) to cancel the component.

Each press of the `'A'` key over a point in the map describes one more variable feature of the component, starting from a delta component at the center of the map. The optional steps (0 selections to 3 selections) are as follows:

0. If `'N'` is pressed without having described any parameters then a delta component will be placed at the center of the map.
1. The first click of the `'A'` key over a selected point in the map defines the position of the center of the new component. A dot is plotted to indicate its position.

Pressing `'N'` at this point installs a delta-component at this position.

2. A second click of the `'A'` key over another selected position in the map defines the axis of a circular gaussian, as the line joining the selected component center to the newly selected position. This line is drawn so that one can judge its veracity.

Pressing 'N' at this point installs a circular gaussian with the axial ratio and axial position angle as fixed parameters.

3. A third click of the 'A' key describes the second axis of an elliptical gaussian component via its distance from the selected component center. The position angle of the first axis selected (in step 2 above) defines the major axis position angle. A line is drawn at right angles to the first axis so that one can check its veracity.

Pressing 'N' at this point installs an elliptical gaussian with all parameters variable. Pressing 'D' cancels the component.

Once the component is installed, its symbol representation is displayed.

If the component turns out to be poorly described, you can then delete it with the 'R' (remove component) key.

Note that the flux given to the component is simply the flux of the pixel under its center. This is not a very good choice, but it is hard to see what would be a good choice, especially given that the map is not re-inverted and re-displayed in maplot each time a new component is added or removed. If you have a better idea of what the component flux should be, use the difmap 'edmod' command to change it. Alternatively let the difmap 'modelfit' command find it for you.

EXTENDED EXAMPLE

To plot a restored map with non-default percentage contours and a reduced grey-scale saturation flux:

```
mapfunc linear, -1, 0.5
levs=-0.1,0.1,0.25,0.5,1,2,4,8,16,32,64
cmul=0
mapplot cln
```

In the above case, the levs= assignment could be avoided by using the loglevs command.

```
loglevs 0.1
```

See help loglevs for more details.

RELATED COMMANDS

```
xyrange - Set the boundaries of the sub-image to be displayed in mapplot.
loglevs  - Set logarithmic levels.
mapcolor - Select mapplot colormap type and transfer function.
mapfunc  - Change the transfer function used in mapplot.
```

beamloc - Change when and where to plot the clean beam ellipse.
polvec - Configure the attributes used when plotting polarization vectors.

mapsize *nx, dx, ny, dy*

Request a new map grid of given width and cell size

EXAMPLES

1. To request a square map with square pixels:

```
0>mapsize 256,0.5
Map grid = 256x256 pixels with 0.5x0.5 milli-arcsec cellsize.
0>
```

2. To request a rectangular grid with rectangular pixels.

```
0>maps 256,0.5,128,1.0
New map grid = 256x128 pixels with 0.5x1.0 milli-arcsec cellsize.
0>
```

3. To see what the current map size is, omit all arguments:

```
0>maps
Map grid = 256x128 pixels with 0.500x1.00 milli-arcsec cellsize.
0>
```

The second line of each example shows the response from the 'mapsize' command, confirming what was typed.

PARAMETERS

- nx** - The number of pixels along the X-axis map and beam grids.
This number must be positive and be an integer power of 2, such as 256 or 512 or 1024.
- dx** - This is the width of each pixel along the X-axis. Unless changed with the mapunits command, the units are milli-arcsec. If this is omitted, or specified as zero, the maximum pixel size that can be used without cutting off any visibilities will be substituted.
- ny** - Optional: If not provided, ny is given the value of nx. The number of pixels along the Y-axis map and beam grids. This number must be positive and be an integer power of 2, such as 256 or 512 or 1024.
- dy** - Optional: If not provided, dy is given the value of dx. This is the width of each pixel along the Y-axis. Unless changed with the mapunits command, the units are milli-arcsec. If this is specified as zero, the maximum pixel size that can be used without cutting off any visibilities will be substituted.

SIDE EFFECTS

Any existing map will be destroyed, and the new map will be initialized to contain 0.0 in each element.

CONTEXT

This command specifies the size of the grid to use to sample the map and beam in the image plane. All commands that operate on maps, such as 'clean', 'restore', 'mapplot' and 'invert', require that such a grid be created before they are invoked. 'mapsize' may be called at any time thereafter to change the size of the map or beam arrays.

The maximum size of the map and beam grids is limited only by the amount of memory available on your computer. Note that the amount of memory required by the two arrays is $2.(nx.ny).sizeof(float)$ bytes. If $sizeof(float)$ is 4 bytes, then the following numbers apply for square grids:

<code>nx=ny</code>	Memory
-----	-----
128	131 Kb
256	524 Kb
512	2.0 Mb
1024	8.4 Mb
2048	34.0 Mb

Clearly, large grids require large amounts of memory, take a much longer time to process and may cause your computer to page heavily. For this reason it is beneficial to use the smallest grid that can accommodate the region of interest. In particular, if there are many sources to be mapped, scattered over a wide area, it may be faster to partially map each source separately, using the 'shift' command to temporarily shift each source into the center of a smaller map area.

CHOOSING THE CELL SIZE

The appropriate choice of cellsize along the X and Y axes of a map grid depends on the sampling of the UV plane by visibilities. To sample the finer detail in the image-plane implied by larger U or V coordinates, requires smaller cell sizes in the map. In particular, to avoid undersampling the image plane, the highest spatial frequencies sampled by the UV data must have a period of at least two pixels. This constraint applies separately to spatial frequencies along the X and Y axes of the map.

The maximum cell sizes that can be selected, can either be implicitly selected by specifying them as 0.0, to the mapsize command, or they can be pre-calculated and displayed by entering the following

commands.

```
print rad_to_map((0.25 - 2/nx) / uv_to_wav(vis_stats(umag)(6))
print rad_to_map((0.25 - 2/ny) / uv_to_wav(vis_stats(vmag)(6))
```

where nx and ny are the number of pixels that are to be requested along the RA and Dec axes of the map.

In the above example, note that `vis_stats(umag)` returns an array of 6 different statistics for the U coordinate magnitudes of all of the visibilities that are included by the current UV range, and that the sixth element of this returned array holds the maximum value of this magnitude. Therefore `vis_stats(umag)(6)` returns the maximum sampled U coordinate that is sampled by any visibility. Since this is returned in the UV units that were last specified via the `'mapunits'` command, the `'uv_to_wav()'` function is then used to convert this to wavelengths. Similarly, the `'rad_to_map()'` function is used to convert the calculated cell size from radians to the units that will be expected by the `'mapsize'` command.

The `" + 2/nx "` part of the above equation, is due to the 2 pixel fuzz that is introduced by the gridding convolution function that is used to interpolate visibilities onto a regular grid.

The maximum cell-size described above is enforced by the `difmap` `'invert'` command, whose job it is to create a dirty map and beam from the visibility data. Visibilities that represent higher spatial frequencies than can be Nyquist sampled by the chosen pixel size are excluded from the maps that it creates. When it does this, the `'invert'` command tells you what percentage of the visibilities had to be excluded and displays the above limiting values of dx and dy so that you can, if you want, change the map grid to avoid omitting any data from subsequent maps. Note that it is not necessarily an error to choose a cell size that is too big to sample all of the visibilities. Sometimes this is the only practicle way to create a map that is large enough to view one's field of interest.

Also note that because the standard CLEAN algorithm only cleans the inner `nx/2` by `ny/2` pixels of the map, the maximum map area cleanable by `'clean'` and displayed by `'mapplot'` is actually the area:

```
X range = -nx/(4.umax+2)/2 -> +nx/(4.umax+2)/2
Y range = -ny/(4.umax+2)/2 -> +ny/(4.umax+2)/2
```

RELATED COMMANDS

```
-----
invert - Invert the UV visibility residuals to form a residual map
        and dirty beam.
restore - Combine model components and a residual map to produce a
        clean map.
```

mapplot - Display the residual or clean map, or dirty beam.

mapunits *unit*

Change the units used to specify and label map and UV plane distances.

EXAMPLES

0>mapu uas

Distances in the map plane now have units: micro-arcsec.

Distances in the UV plane now have units: giga-wavelengths.

0>

0>mapunit mas

Distances in the map plane now have units: milli-arcsec.

Distances in the UV plane now have units: mega-wavelengths.

0>

0>mapunit arcsec

Distances in the map plane now have units: arcsec.

Distances in the UV plane now have units: kilo-wavelengths.

0>

0>mapunit

Distances in the map plane now have units: arcsec.

Distances in the UV plane now have units: kilo-wavelengths.

0>

ARGUMENTS

unit - Optional.

One of the following distance units:

 mas - Milli-arcseconds and mega-wavelengths.

 arcsec - Arcseconds and kilo-wavelengths.

 arcmin - Arcminutes and kilo-wavelengths.

If no name is provided, then the current settings are reported without modification, as in the third example above.

CONTEXT

VLBI antennas are generally separated by a few million wavelengths and are sensitive to source structure at the milli-arcsecond scale. The natural units to express these characteristics are thus milli-arcseconds in the map plane, and mega-lambda in the UV plane.

Other more compact arrays such as the VLA, generally have baseline lengths that are more naturally expressed in kilo-wavelengths, and are sensitive to source structure at the arcsecond scale.

Since there is no one set of units that satisfies all observations, difmap gives you the choice of a few units. The chosen units refer

both to the units of the arguments to difmap commands, and to how distances are displayed and labelled by difmap.

Unless the 'mapunits' command is invoked to change the units, difmap sets the map-plane distance units to milli-arcseconds and the baseline length units to mega-wavelengths. This reflects difmap's VLBI heritage.

mapvalue *x, y*

Return the value of the map pixel nearest to a given coordinate.

EXAMPLE

1. The following use of mapval() displays the flux in the map at a position 3.5 arcsec to the east and 0.7 arcsec to the north of the center of the map.

```
0>mapunits arcsec
Distances in the map plane now have units: arcsec.
Distances in the UV plane now have units: kilo-wavelengths.
0>print mapval(3.5,0.7)
0.293557
0>
```

2. The flux at the center of the map can be displayed with:

```
0>print mapval(0,0)
3.44326
0>
```

PARAMETERS

- x - The X offset with respect to the center of the map, as displayed by maplot.
- y - The Y offset with respect to the center of the map, as displayed by maplot.

SIDE-EFFECTS

If the map is out of date with the UV data then it will be re-inverted before returning a value.

RELATED FUNCTIONS

- peak - Return details of the min,max or absolute max flux in a map.
- imstat - Return image plane map statistics.

mark_radec *ra, dec, symbol, color, size, text, just, x_offset, y_offset*

Add an optionally annotated map marker to be drawn at a given RA, Dec.

EXAMPLES

1. The following example labels the source 3c273, placing the text 3 characters to the right of, and 4 characters above the source position, and drawing an arrow from this point to the source position. The arrow and the text are drawn with pgplot color 10, which matches the green color of the mapplot axes.

```
0>mark_radec 12:29:06.7, 2:03:08.60, arrow, 10, 1.0, "3c273", 0.0, 3, 4
```

2. The following example draws a "+" symbol at the location of 3c273, accompanied by a label, which, since positional specifications haven't been given, is drawn to just to the right of the symbol.

```
0>mark_radec 12:29:06.7, 2:03:08.60, plus, 10, 1.0, "3c273"
```

PARAMETERS

ra - The target Right Ascension expressed in hours, minutes and seconds of time.

dec - The target Declination expressed in degrees, minutes and seconds of arc.

symbol - (Default = dot)
The symbol to draw. If this is the word

arrow

then an arrow is drawn between the annotation text and the specified source. Otherwise it must be one of the following names of symbols to draw at the source position.

filled_diamond	- An opaque diamond.
filled_triangle	- An opaque triangle.
open_square	- A square outline.
dot	- A small dot.
plus	- A + symbol.
asterisk	- An * symbol.
cross	- An x symbol.
open_triangle	- Triangular outline.
circle_plus	- A plus symbol inside a circle.
circle_dot	- A dot inside a circle.
knotted_hanky	- The PGLOT symbol that looks like a handkerchief with knots in its corners!

- open_diamond - The outline of a diamond shape.
 - open_star - A line drawing of a star.
 - maltese_cross - A line drawing of a maltese cross.
 - star_of_david - A line drawing of a star of david
 - filled_square - An opaque square drawn.
 - filled_circle - An opaque circle.
 - filled_star - An opaque star shape.
 - circle1 - A small circle.
 - circle2 - A slightly larger circle.
 - circle3 - Another slightly larger circle.
 - circle4 - Yet another slightly larger circle.
- color - (Default = 11)
The PGPLOT color index with which to draw the marker and its annotation text. The standard pgplot colors are:
 - 0 - Black
 - 1 - White
 - 2 - Red
 - 3 - Green
 - 4 - Blue
 - 5 - Cyan
 - 6 - Magenta
 - 7 - Yellow
 - 8 - Orange
 - 9 - Green + Yellow
 - 10 - Green + Cyan
 - 11 - Blue + Cyan
 - 12 - Blue + Magenta
 - 13 - Red + Magenta
 - 14 - Dark Grey
 - 15 - Light Grey
- size - (Default = 1.0)
The size of characters and plot symbols, relative to the normal character size.
- text - (Default = "")
The annotation string.
- just - (Default = 0.0)
The justification string, where 0.0 means left justify, 0.5 means center, and 1.0 means right justify.
- x_offset - (Default = 1.0)
The horizontal offset of the justification point of the annotation string relative to the source location. The offset is specified in character widths, with positive values moving the string to the right, and negative to the left.

`y_offset` - (Default = 0.5)

The vertical offset of the center of the annotation string relative to the source location. The offset is specified in character heights, with positive values moving the string towards the top of the plot, and negative values towards the bottom of the plot.

CONTEXT

This command allows one to label known sources within a map. The epoch of the Right Ascension and Declination are assumed to be the same as that of the current observation. This can be seen by running the 'header' command and looking at the number in parentheses which follows the RA listed under "Source parameters:".

Markers added using this command are added to the list of markers which are passed to subsequent invocations of the `mapplot` command.

The 'save' command writes 'mark_radec' commands to the output parameter file, so that the list can be preserved between `difmap` sessions.

ASSOCIATED COMMANDS

<code>mapplot</code>	- Draw a map.
<code>mark_xy</code>	- Specify a map marker relative to the map center.
<code>clear_markers</code>	- Delete the current list of markers.
<code>wmarkers</code>	- List map marker commands to stdout or a file.
<code>delmarker</code>	- Delete a given map marker.

mark_xy *x, y, symbol, color, size, text, just, x_offset, y_offset*

Add an optionally annotated map marker to be drawn at a given X,Y offset.

EXAMPLES

1. The following example labels a feature at 10 arcminutes east, and 5 arcminutes north of the map center, placing the text 3 characters to the right of, and 4 characters above the marked position, and drawing an arrow from this point to the marked position. The arrow and the text are drawn with pgplot color 10, which matches the green color of the mapplot axes.

```
0>mapunits arcmin
....
0>mark_xy 10.0, 5.0, arrow, 10, 1.0, "Interesting huh?", 0.0, 3, 4
```

2. The following command just draws a red asterix at 2 arcseconds east and 4 arcseconds north of the map center. No annotation is drawn.

```
0>mapunits arcmin
....
0>mark_xy 2.0, 4.0, asterix, 2
```

PARAMETERS

- | | |
|--------|--|
| x | - The eastward offset of the position to be marked, specified in the units that were last selected with the 'mapunits' command. |
| y | - The northward offset of the position to be marked, specified in the units that were last selected with the 'mapunits' command. |
| symbol | - (Default = dot)
The symbol to draw. If this is the word |

arrow

then an arrow is drawn between the annotation text and the specified source. Otherwise it must be one of the following names of symbols to draw at the source position.

filled_diamond	- An opaque diamond.
filled_triangle	- An opaque triangle.
open_square	- A square outline.
dot	- A small dot.
plus	- A + symbol.
asterisk	- An * symbol.

- cross - An x symbol.
 - open_triangle - Triangular outline.
 - circle_plus - A plus symbol inside a circle.
 - circle_dot - A dot inside a circle.
 - knotted_hanky - The PGPLOT symbol that looks like a handkerchief with knots in its corners!
 - open_diamond - The outline of a diamond shape.
 - open_star - A line drawing of a star.
 - maltese_cross - A line drawing of a maltese cross.
 - star_of_david - A line drawing of a star of david
 - filled_square - An opaque square drawn.
 - filled_circle - An opaque circle.
 - filled_star - An opaque star shape.
 - circle1 - A small circle.
 - circle2 - A slightly larger circle.
 - circle3 - Another slightly larger circle.
 - circle4 - Yet another slightly larger circle.
- color - (Default = 11)
The PGPLOT color index with which to draw the marker and its annotation text. The standard pgplot colors are:
- 0 - Black
 - 1 - White
 - 2 - Red
 - 3 - Green
 - 4 - Blue
 - 5 - Cyan
 - 6 - Magenta
 - 7 - Yellow
 - 8 - Orange
 - 9 - Green + Yellow
 - 10 - Green + Cyan
 - 11 - Blue + Cyan
 - 12 - Blue + Magenta
 - 13 - Red + Magenta
 - 14 - Dark Grey
 - 15 - Light Grey
- size - (Default = 1.0)
The size of characters and plot symbols, relative to the normal character size.
- text - (Default = "")
The annotation string.
- just - (Default = 0.0)
The justification string, where 0.0 means left justify, 0.5 means center, and 1.0 means right justify.

`x_offset` - (Default = 1.0)
 The horizontal offset of the justification point of the annotation string relative to the source location. The offset is specified in character widths, with positive values moving the string to the right, and negative to the left.

`y_offset` - (Default = 0.5)
 The vertical offset of the center of the annotation string relative to the source location. The offset is specified in character heights, with positive values moving the string towards the top of the plot, and negative values towards the bottom of the plot.

CONTEXT

This command allows one to label visible features in a map. Note that to interactively determine the location of such a feature, invoke 'mapplot', and hit the 'v' key while the mouse cursor is centered over the feature. The returned numbers can then be used in your call to the 'mark_xy' command. Internally the marked position is recorded as the corresponding Right Ascension and Declination. As a result, if you shift the map, the marker will move with any features therein.

Markers added using this command are added to the list of markers which are passed to subsequent invocations of the mapplot command.

The 'save' command writes 'mark_radec' commands to the output parameter file, so that the list can be preserved between difmap sessions.

ASSOCIATED COMMANDS

<code>mapplot</code>	- Draw a map.
<code>mark_radec</code>	- Specify a map marker to be drawn at a given RA,Dec.
<code>clear_markers</code>	- Delete the current list of markers.
<code>wmarkers</code>	- List map marker commands to stdout or a file.
<code>delmarker</code>	- Delete a given map marker.

mjd *date_and_time*

Return the Modified Julian Date corresponding to a given calendar date.

EXAMPLES

The following example, returns the Modified Julian Date that corresponds to 13-jun-2001 at 10:30am UTC.

```
0>print mjd(13-jun-2001:10:30)
52073.4
0>
```

PARAMETERS

`date_and_time` - This must be a Gregorian calendar date and time, formatted like `dd-mmm-yyyy:hh:mm`. Where `dd` is the date within the month, `mmm` is a three character month name, `yyyy` is a fully qualified year, `hh` is the hour in the day, and `mm` is the number of minutes into the hour. Trailing time components can be omitted.

RETURN VALUE

The return value is a Modified Julian Date, expressed in floating point. This unfortunately only provides a resolution of about 5 minutes.

model_flux *do_continuum*

Return the flux in either the normal model or the continuum model.

EXAMPLES

```
0>print model_flux()
```

```
1.37916
```

```
0>
```

```
0>print "Flux in normal model =", model_flux(false)
```

```
Flux in normal model = 0.830256
```

```
0>print "Flux in continuum model =", model_flux(true)
```

```
Flux in continuum model = 1.37916
```

```
0>
```

PARAMETERS

do_continuum - (Default=false)

If this argument is omitted or given as FALSE, the total flux in the normal established and tentative models is returned. If it is given as TRUE, then the flux of just the continuum model (if any) will be returned.

CONTEXT

This command is used to return the total flux in either the normal or the continuum model. This is the same flux that is reported at the end of the clean and modelfit commands.

RELATED COMMANDS

showmodel - Report detailed information about the current models.

modelfit *niter*

Fit image-plane model components to visibilities in the UV plane.

EXAMPLE

0>modelfit 4

Partitioning the model into established and variable parts.

The fixed established model contains 0 components (0 Jy).

The variable part of the model contains 2 components (0.97 Jy).

There are 10 variables and 4510 usable visibilities.

This gives $2 \times 4510 - 10 = 9010$ degrees of freedom.

Reduced Chi-squared = Chi-squared / 9010.

Iteration 00: Reduced Chi-squared=46.315838

! Flux (Jy)	Radius (mas)	Theta (deg)	Major (mas)	Axial ratio	Phi (deg)	T \
! Freq (Hz)	SpecIndex					
0.950000v	0.00000	0.00000	1.00000v	1.00000v	0.00000v	1 \
3.10000e+10	0.0					
0.0200000v	1.00000v	-130.000v	1.00000v	1.00000v	0.00000v	1 \
3.10000e+10	0.0					

Iteration 01: Reduced Chi-squared=4.3881679

! Flux (Jy)	Radius (mas)	Theta (deg)	Major (mas)	Axial ratio	Phi (deg)	T \
! Freq (Hz)	SpecIndex					
0.416386v	0.00000	0.00000	0.524179v	0.437841v	-6.49258v	1 \
3.10000e+10	0.0					
0.0826878v	1.00780v	-137.060v	3.31603v	0.732082v	-56.6298v	1 \
3.10000e+10	0.0					

Iteration 02: Reduced Chi-squared=0.89815858

! Flux (Jy)	Radius (mas)	Theta (deg)	Major (mas)	Axial ratio	Phi (deg)	T \
! Freq (Hz)	SpecIndex					
0.493566v	0.00000	0.00000	0.371532v	0.167305v	-7.53078v	1 \
3.10000e+10	0.0					
0.0209711v	1.20514v	-165.679v	3.93011v	0.331629v	-49.3905v	1 \
3.10000e+10	0.0					

Iteration 03: Reduced Chi-squared=1.0050499 (Increased)

Iteration 04: Reduced Chi-squared=1.0088361 (Increased)

0>

ARGUMENTS

niter - The number of iterations to try, or zero to just see the current fit. 10 and 20 are good numbers to try, before the first selfcal.

CONTEXT

 Model fitting directly to the visibilities of interferometric data has two common uses.

1. It is employed to find good starting models for the initial phase self-calibration of phase-unstable data, especially when a point source starting model has proved insufficient.
2. It is commonly used to parameterise the general characteristics of sources with as few variables as possible.

The difmap modelfit program fits aggregates of various forms of model components, fitting directly to the real and imaginary parts of the observed visibilities using the powerful Levenberg-Marquardt non-linear least squares minimization technique. This makes it substantially faster and a lesser consumer of memory than the Caltech VLBI package modelfit program, that fitted to an ad hoc combination of closure-phase and amplitude.

The disadvantage of the newer algorithm is that it assumes that the phases are well calibrated. The older algorithm was insensitive to this through its use of closure phases. As a result, using modelfit in difmap may require a number of interleaved self-calibration steps to converge on the model that agrees well with the closure phases.

Note that in version 2.3d the scope of the uvrage command was extended to include modelfitting. Thus you can now restrict modelfit to only fit to visibilities that lie within a given range of distances from the center of the UV plane. This can be useful if you only have believable flux in the short baselines, or if you want to start by fitting the brighter larger scale features before complicating the model with weaker fine structures.

SPECIFYING MODELS FOR MODELFIT

There are three ways to specify model components to modelfit.

1. In model files, components with variable parameters are denoted by a 'v' character post-fixing each of the variable parameters.

Thus in the following model the first component is an elliptical gaussian component with all parameters variable except its position, while the second component has all parameters variable.

! Flux (Jy)	Radius (mas)	Theta (deg)	Major (mas)	Axial ratio	Phi (deg)	T
0.140490v	0.00000	0.00000	0.704297v	0.147144v	-48.4654v	1
0.118984v	1.01917v	137.776v	1.10527v	0.544262v	-57.5779v	1

The meaning of each of the above parameters (as originally defined for the Caltech VLBI package), is as follows:

- Flux - The integrated flux in the component (Jy).
- Radius - The radial distance of the component center from the center of the map (milli-arcsec).
- Theta - The position angle of the center of the component (degrees North -> East) wrt an imaginary line drawn vertically through the map center.

The following components may be omitted for delta components, except when you want to specify spectral indexes, in which case they should be specified as zero.

- Major - The FWHM major axis of the elliptically stretched component (milli-arcsec).
- Ratio - The ratio of the minor axis to the major axis (0 -> 1).
- Phi - The Position angle of the major axis (degrees North -> East).
- T - The type of component. Recognised types are:
 - 0 - Delta function.
 - 1 - Gaussian.
 - 2 - Uniformly bright disk.
 - 3 - Optically thin sphere.
 - 4 - Ring.
 - 5 - Un-supported component type - kept for compatibility.
 - 6 - Sunyaev-Zel'dovich.

The following two parameters are used to optionally give components spectral indexes.

- Freq - The frequency for which the specified Flux value is defined, specified in Hz.
- SpecInd - The spectral index.

When modelfitting multiple IFs the above two parameters are used to compute different fluxes for each IF, such that given an IF frequency of f , the corresponding model component flux is:

$$\text{Flux} * (f/\text{Freq})^{\text{SpecInd}}$$

where Flux, Freq and SpecInd are model component parameters described above. Note that when multiple spectral line channels are selected per IF, the frequency used for each IF is the mean of the selected spectral line channel frequencies within that IF. It is assumed that the flux doesn't change significantly between spectral-line channels within an IF.

Internally the positions of components are recorded by their X and Y axis positions wrt the map center and fitting is wrt these parameters. For this reason if either Radius or Theta is made variable, both will be taken as variable. It is not possible to fix one or the other

individually. Similarly, except for the special case of a circular component where only Major is marked as variable and the axial ratio is exactly 1.0, it is not possible to individually mark Major, Ratio and Phi to be fixed. Marking one as variable sets them all as variable.

More obviously, the model component type T can not be marked to be variable.

2. While model components can be specified via a model file in the above format, it is usually more convenient to incrementally add and remove components interactively via the difmap 'mapplot' command. See help mapplot for further details.
3. The final method is really directed towards changing the trial parameters of an existing variable model, although new components may also be created. This method uses the difmap 'edmodel' command, which invokes an editor on an internally generated model scratch file, initialized with the existing variable model components, appropriately shifted if the 'shift' command has previously been used. See help edmodel for further details.

RELATED COMMANDS

```
-----
showmodel - Report detailed information about the current models.
edmodel   - Edit the current models with an external editor.
rmodel    - Read a new model for modelfit.
mapplot   - Mapplot allows interactive addition and removal of
             components.
addcmp    - Add a model component by hand to the tentative model.
uvrange   - Set the min and max UV radii of usable visibilities.
```

multi_model *enable*

Specify whether to keep one model per channel-range/polarization selection.

EXAMPLE

```
O>multi_model true
Maintain separate models for each channel/polarization selection.
```

```
O>multi_model false
Use one model for all channel/polarization selections.
```

PARAMETERS

```
enable - Default=false.
        If true, different models are associated with each
        channel-range/polarization selection.
        If false one model is used for all selections.
```

CONTEXT

When making polarization or spectral line maps, each spectral-line channel and polarization samples a different aspect of the source, and should thus be represented by a different model. By default difmap doesn't maintain separate models for different channel-range/polarization selections, so in such cases, it is necessary to tell it to do so via the 'multi_model' command. Having done so, the 'select' command maintains a table of models for each channel-range/polarization selection that the user has mapped, saving and restoring models from this table whenever a new selection is made. These models can be cleared with the clear_models command, written to a file with the write_models command, and read from a file with the read_models command.

Conversely, when making single-polarization continuum maps, it only makes sense to have one model, since only one map will be produced at the end of the day. However such datasets commonly include multiple IFs, and it isn't uncommon for people to switch between selections of IFs while making maps, as a sanity check. In such cases one doesn't want the select command to keep switching in different models whenever a new selection of IFs is chosen, so by default the same model is maintained for all selections. To maintain backwards compatibility with older versions of difmap, this is the default mode.

Note that you can change the effective default by placing the appropriate 'multi_model' command in your difmap startup script (ie. the file specified by the DIFMAP_LOGIN environment variable).

RELATED COMMANDS

read_models - Restore a set of multi-model models from a file.
write_models - Save the current set of multi-model models to a file.
clear_models - Discard all multi-model models.
mapplot - Display map and beam images.

nbase *subarray*

Return the number of baselines in a given sub-array.

EXAMPLE

Example 1 - Write a message saying how many baselines are in sub-array 1.

```
0> print "There are", nbase, "baselines in sub-array 1"
There are 10 baselines in sub-array 1.
```

Example 2 - List the names of all 'nbase' baselines in sub-array 2.

```
0> do i=1,nbase(2)
1>   print basename(i,2)
1> end do
WSRT-BOLOGNA
WSRT-HSTK
BOLOGNA-HSTK
WSRT-VLBA_KP
BOLOGNA-VLBA_KP
HSTK-VLBA_KP
WSRT-VLBA_LA
BOLOGNA-VLBA_LA
HSTK-VLBA_LA
VLBA_KP-VLBA_LA
0>
```

PARAMETERS

subarray - Optional: Defaults to 1.
This is the number of the antenna sub-array to return the baseline count from. The 'header' command may be used to examine what sub-arrays are available.

RELATED FUNCTIONS

basename - Return the name of the n'th baseline of a given sub-array.

nchan

Return the count of the number of spectral-line channels per IF.

EXAMPLE

```
O>print "There are", nchan, "spectral-line channels."
There are 96 spectral-line channels.
O>
```

CONTEXT

'nchan()' is a function that returns the integer count of the number of spectral-line channels per IF of the current observation.

RELATED FUNCTIONS AND COMMANDS

nif	-	Return the number of IFs in the current observation.
nsub	-	Return the number of sub-arrays in the current observation.
nbase	-	Return the number of baselines in a given sub-array.
ntel	-	Return the number of telescopes in a given sub-array.
header	-	Displays the characteristics of the current observation.

nif

Return the number of IFs in the current observation.

EXAMPLE

```
O>print "There are", nif, "IFs in the current observation."
There are 2 IFs in the current observation.
O>
```

CONTEXT

nif() is a function that returns the integer count of the number of IFs in the current observation.

RELATED FUNCTIONS AND COMMANDS

nchan	-	Return the number of spectral-line channels per IF.
nsub	-	Return the number of sub-arrays in the current observation.
nbase	-	Return the number of baselines in a given sub-array.
ntel	-	Return the number of telescopes in a given sub-array.
header	-	Displays the characteristics of the current observation.

nsub

Return the number of sub-arrays in the current observation.

EXAMPLE

```
0>print "There are", nsub, "sub-arrays in the current observation."
There are 4 sub-arrays in the current observation.
0>
```

CONTEXT

nsub() is a function that returns the integer count of the number of telescope sub-arrays per IF of the current observation.

RELATED FUNCTIONS AND COMMANDS

nif	-	Return the number of IFs in the current observation.
nchan	-	Return the number of spectral-line channels per IF.
nbase	-	Return the number of baselines in a given sub-array.
ntel	-	Return the number of telescopes in a given sub-array.
header	-	Displays the characteristics of the current observation.

ntel *subarray*

Return the number of telescopes in a given sub-array.

EXAMPLE

Example 1 - Write a message saying how many telescopes are in the first sub-array.

```
0> print "There are", ntel, "telescopes in sub-array 1"
```

```
There are 13 telescopes in sub-array 1
```

Example 2 - List the names of all 'ntel' telescopes in sub-array 2.

```
0> do i=1,ntel(2)
```

```
1>   print telname(i,2)
```

```
1> end do
```

```
BONN
```

```
WSRT
```

```
JBNK1
```

```
BOLOGNA
```

```
VLBA_KP
```

```
VLBA_PT
```

```
VLBA_LA
```

```
VLA27
```

```
OVR0
```

```
0>
```

PARAMETERS

subarray - Optional: Defaults to 1.

This is the number of the antenna sub-array to return the telescope count from. The 'header' command may be used to examine what sub-arrays are available.

RELATED FUNCTIONS

telname - Return the name of the n'th telescope of a given sub-array.

observe *file_name, bin_width, do_scatter*

Read UV data from a random-groups UV FITS file.

EXAMPLE

```
O>observe /scr/mcs/fits/gbt_spec.uvf
Reading UV FITS file: /scr/mcs/fits/gbt_spec.uvf
AN table 1: 7 integrations on 45 of 45 possible baselines.
Apparent sampling: 1 visibilities/baseline/integration-bin.
Found source: 1641+399
```

There are 2 IFs, and a total of 128 channels:

IF	Channel origin	Frequency at origin	Freq offset per channel	Number of channels	Overall IF bandwidth	
						(Hz)
01	1	1.53575e+10	125000	64	8e+06	
02	65	1.53655e+10	125000	64	8e+06	

Polarization(s): RR LL RL LR

Read 353 lines of history.

Reading 161280 visibilities.

O>

All lines following the 'observe' command line are informational output messages from the observe command.

PARAMETERS

file_name - The name of the UV FITS file to be read.

bin_width - Default=0.0 (seconds)

This argument is used to specify how visibilities are to be collected into integrations. This is discussed further in the CONTEXT section below. There are two alternatives:

1. If `binwid<1.0` seconds then 'observe' will assign an integration to every time-stamp in the file. This is ok if visibilities on different baselines within the same integration have the same time-stamps, but disastrous otherwise.
2. If `binwid>=1.0` seconds then 'observe' will bin visibilities onto a regular integration grid of spacing 'binwid' seconds anchored at 0 UTC on the first day of the observation. Where multiple

visibilities fall on the same baseline of a given integration, they will be vector averaged using a weighted mean using the same algorithm as uvaver.

do_scatter - Default=false
 If do_scatter=true the weights from the FITS file will be supplanted by weights estimated from the scatter of the data within each integration bin. There must be at least two visibilities per baseline within each integration bin to do this. Where this is not the case the output visibility will be flagged, and be assigned the input visibility weight.

CONTEXT

Before interferometer data can be processed, it must be read into difmap. The observe command reads interferometer data in the form of single source UV FITS files. The older Caltech VLBI merge format files are no longer acceptable. The caltech VLBI package program 'MERGEFITS' may be used to convert merge files to FITS.

UV FITS files do not provide any means to map visibilities on different baselines into integrations. Each visibility has its own time-stamp, which need not agree with those on other baselines within the same logical integration. In fact UV FITS files do not even limit visibilities on different baselines to have the same integration time. Difmap, on the other hand does require that visibilities be grouped into integrations. This is the reason for the 'binwid' argument of the 'observe' command. If the visibilities do not lie on an integration grid then you must specify a suitable integration time into which visibilities should be binned into integrations. Depending on how the FITS file has been processed, it may already have visibilities grouped into integrations with identical time-stamps assigned to each grouped visibility, in which case no 'binwid' argument will be required. This is the case for files written by difmap. If you do not know what state your file is in, then try to read it with the observe command without specifying an integration time. Then if 'observe' reports something along the following lines, then either run the 'uvaver' command to re-grid the data or equivalently re-run the observe command with a suitable integration time.

```
observe poor_sampling.uvf
...
Apparent sampling: 0.245305 visibilities/baseline/integration.
*** This seems a bit low - see "help observe" on the binwid argument.
...
```

Other symptoms of in-completely binned integrations are that selfcal flags all of your data due to the lack of closure quantities, and that station based editing in vplot behaves like baseline based editing.

While the 'binwid' and 'scatter' options are provided primarily as a mechanism to group data into integrations, they may also be employed to average ones data to longer integration times for efficiency reasons. The effect is identically equivalent to reading the data with observe and then running the 'uvaver' command with the same arguments, except that in the latter case you might not have sufficient memory to read the data before averaging and reading in the larger file would take longer.

LIMITATIONS

Only single source, time-ordered UV FITS files will be accepted by difmap. Multi-source files must be split into one or more single source files before presenting them to difmap. AIPS provides the SPLIT task to do this. Similarly, AIPS UVSRT may be used to re-sort files into time order (eg. TB order).

Difmap accepts multiple sub-arrays within a file, multiple polarizations, spectral-line channels and multiple IFs.

While difmap does accept multiple IFs, it does not make any allowances for spectral-index effects over the range of frequencies covered by the IFs. If the source structure changes significantly between IFs, you should SPLIT the IFs into separate files and process them independantly within difmap. Also note that difmap requires that the frequencies assigned to each IF do not change over the duration of the file. To enforce this difmap rejects files containing multiple frequency IDs.

SIDE EFFECTS

Any previously read UV data will be discarded before the new file is read. Any existing models and CLEAN windows will also be discarded. All weight options are restored to their defaults.

The current map-grid will be cleared, but its dimensions and cell-size will not be changed.

RELATED COMMANDS

wobs - Write out a modified UV FITS file.
 select - Select a polarization and channels to be processed.
 uvaver - Used to re-grid UV data.

peak *attribute, pixel_type*

Return details of the min,max or absolute max flux in a map.

EXAMPLE

1. Display the min,max and absolute max flux in the current map.

```
0>pr peak(flux,min),peak(flux,max),peak(flux,abs)
-0.543233 3.44326 3.44326
0>
```

2. Shift the phase-center to that of the map pixel that contains the max absolute flux to the center of the map.

```
0>shift -peak(x),-peak(y)
shift: Shifting UV data, clean models and windows by: 20 (mas) East
shift: Shifting UV data, clean models and windows by: -20 (mas) North
0>
```

PARAMETERS

- attribute** - The name of the attribute of the selected pixel to be returned by the function.
Possible values are:
- flux** - The flux of the selected pixel.
 - x** - The X-axis position of the pixel.
 - y** - The Y-axis position of the pixel.
 - ra** - The Right Ascension of the pixel (degrees).
Beware that this is returned in a single precision number which only has 6 significant figures of precision.
 - dec** - The declination of the pixel (degrees).
Beware that this is returned in a single precision number which only has 6 significant figures of precision.
- pixel_type** - Default = abs
The name of the type of pixel to return the details of.
Possible values are:
- abs** - The pixel with the max absolute flux in the map.
 - min** - The pixel with the min flux in the map.
 - max** - The pixel with the max flux in the map.

CONTEXT

peak() is a function that returns details about one of two specific pixels of the current residual or clean map. The two pixels are, the pixel having the minimum flux in the map, and the pixel having the maximum flux in the map. In addition, one of these pixels has a higher absolute flux than the other, and requesting the pixel with the max

absolute flux will return details about this pixel.

The selectable attributes of the requested pixel are its (signed) flux, and its X and Y axis positions with respect to the center of the map, all measured in map units.

SIDE EFFECTS

If the map is not up to date, then it will be inverted before a value is returned.

RELATED FUNCTIONS

mapvalue - Return the value of the map pixel closest to a given position.

imstat - Return image plane map statistics.

peakwin *size, doabs*

Place a CLEAN window around the brightest pixel in the residual map.

EXAMPLE

0>peakwin

Added new window around map position (2, -2).

0>

PARAMETERS

size - Default = 1.0

The relative size of the clean window with respect to the aspect of the clean beam.

doabs - Default = false

By default the clean window will be placed around the pixel with the maximum flux. However, if doabs=true then the clean window will be placed around the pixel with the maximum absolute value in the map. The latter could be useful for spectral line mapping of absorption features.

CONTEXT

This command is provided as a primitive for automatic mapping scripts, where CLEAN windows need to be set non-interactively. The command looks at the position of the brightest pixel in the residual map, and if it is not already enclosed in a CLEAN window, adds a new CLEAN window centered upon it. The added window has the dimensions of the rectangular area that just encloses the ellipse of the CLEAN beam last estimated by the 'invert' command, scaled by the optional 'size' argument.

SIDE EFFECTS

If the residual map and/or beam is not up to date when this function is called, the 'invert' command will be invoked.

`planet_geometry planet, epoch`

Return the geometry of a given planet.

EXAMPLES

1. In the first example, the 'observe' command is first used to read in an observation of Jupiter. This enables the 'planet_geometry' command to get the planet name and the epoch of interest from the observation.

```
0>obs jupiter.uvf
...
0>planet_geometry
jupiter: Equatorial diameter=35.4563 arcsec
Flattening=0.06487
Geometric diameter=34.287 arcsec
Solid angle=923.314 arcsec^2
0>
```

or in its functional form:

```
0>obs jupiter.uvf
...
0>print planet_geometry()
35.4563 0.06487 34.287 923.314
0>
```

2. If no planet observation has been read in, the planet name and epoch must be specified.

```
0>planet_geometry jupiter, mjd(13-jun-2001:7:12)
jupiter: Equatorial diameter=32.2457 arcsec
Flattening=0.06487
Geometric diameter=31.1823 arcsec
Solid angle=763.672 arcsec^2
0>
```

or in its functional form:

```
0>print planet_geom(jupiter, mjd(13-jun-2001:7:12))
32.2457 0.06487 31.1823 763.672
0>
```

Note that the `mjd()` function has been used specify the Modified Julian Date of 2001-jun-13 at 7:12am UTC.

PARAMETERS

`planet` - Default=""

The name of the planet of interest.

If this argument is omitted, or specified as "", the source name of the current observation is used. This must be a simple un-embellished planet name, specified in uppercase, lowercase or mixed case letters. Only the planets Mercury, Venus, Mars, Jupiter, Saturn, Uranus and Neptune are recognized. Pluto isn't well enough characterized to be usable.

epoch - Default=0.0
 The UTC date of the observation, expressed as a Modified Julian Date. If this argument is omitted, or specified as 0.0, the mid date of the current observation is substituted.

RETURN ARRAY

If the command is used as a function, it returns an array of values, these being the following parameters, in the following order:

- (1) - The equatorial diameter (a) in arc-seconds.
- (2) - The flattening $(a-b)/a$.
- (3) - The geometric diameter in arc-seconds.
- (4) - The solid angle subtended by the planet,
 in arc-seconds squared.

`planet_temp flux, planet, epoch, freq`

Find the brightness temperature implied by the measured flux of a planet.

EXAMPLES

1. In the following example, the 'observe' command was first used to read in an observation of Jupiter, then a range of channels was selected using the 'select' command. Then, modeling it as a point source with spectral index of 2.0, modelfit returned a best fit flux of 181.454 Jy at the center frequency of the observation.

Presenting the planet_temp command with this flux, caused it to deduce that Jupiter was being observed by looking at the source name that was read from the UV FITS file. Since no frequency or epoch were specified, it substituted the middle frequency and the date of the mid point of the observation.

```
0>planet_temp 181.454
Apparent brightness temperature = 150.83 Kelvin
```

2. The second example is the same as the first, except that instead of just displaying the result to the user, we capture the returned temperature in a variable.

```
0>float temp
0>temp = planet_temp(181.454)
```

3. If no observation has been read in, using the planet_temp command is more cumbersome, in that since the planet name, the epoch of interest, and the frequency can't be taken from the observation, they have to be specified explicitly. For example,

```
0>planet_temp 1.0, jupiter, mjd(13-jun-2001:7:12), 30e9
Apparent brightness temperature = 2.01477 Kelvin
0>
```

equivalently using the functional form,

```
0>print planet_temp(1.0,jupiter,mjd(13-jun-2001:7:12),30e9)
2.01477
0>
```

Note that the mjd() function has been used specify the Modified Julian Date of 2001-jun-13 at 7:12am UTC.

ARGUMENTS

flux - The flux to be converted to a brightness temperature.

- planet - Default=""
 The name of the planet which returned the above flux.
 If this argument is omitted, or specified as "", the
 source name of the current observation is used. This
 must be a simple un-embellished planet name, specified
 in uppercase, lowercase or mixed case letters. Only
 the planets Mercury, Venus, Mars, Jupiter, Saturn,
 Uranus and Neptune are recognized. Pluto isn't well
 enough characterized to be usable.
- epoch - Default=0.0
 The UTC date of the observation, expressed as a
 Modified Julian Date. If this argument is omitted,
 or specified as 0.0, the mid date of the current
 observation is substituted.
- freq - Default=0.0
 The frequency of the observation (Hz). If this
 argument is omitted, or specified as 0.0, the mean
 frequency of the currently selected channels is
 used.

CONTEXT

To compute the brightness temperature of a planet, given its measured flux in Jy, one needs the solid angle, w , subtended by the planet in steradians and the frequency, f , at which the flux was measured. One can then use the Rayleigh-Jeans approximation to compute a brightness temperature defined by the following equation:

$$T = c^2 * \text{flux} * 1.0e-26 / (w * 2 * k * f^2)$$

where c is the speed of light, k is Boltzmann's constant, and the factor $1.0e-26/w$ converts Jy to $W/m^2/Hz/sr$.

In order to compute the solid angle subtended by the planet, an approximate algorithm implemented in `slalib` is used to compute the distance of the target planet at the specified epoch, and this is combined with the equatorial diameter and flatnesses tabulated in the Explanatory Supplement to the Astronomical Almanac.

If you are planning to use the `'modelfit'` command to fit for the flux of the planet, and you are doing this over a wide range of frequency channels, I suggest that you create the initial model component interactively using the `'n'` key in the `'mapplot'`, then use the `'edmod'` command to change the spectral index to 2.0 before preceding to run the `modelfit` command (you could also use the `'addcmp'` command to do both of these steps non-interactively). Model fitting will then provide you with a flux that reflects the thermal nature of the

radiation, as implicitly assumed by the 'planet_temp' command. The frequency that you present to the 'planet_temp' command should be the same frequency as was associated with the spectral index in the model component.

pointing_center *ra, dec*

Specify the original pointing center of the observation.

EXAMPLE

```
0>pointing_center 10:45:23.123, -02:12:56.234
Pointing center: RA=10:45:23.123 Dec=-02:12:56.234 (2000.0)
0>
```

PARAMETERS

- ra - The Right Ascension of the pointing center, written as a sexagesimal number of hours, minutes and seconds.
- dec - The Declination of the pointing center, written as a sexagesimal number of degrees, minutes and seconds.

CONTEXT

If the visibilities in a UV FITS file have been shifted to a new pointing center, and there are no OBSRA and OBSDEC keywords to indicate the original pointing center, then any primary beam that the user specifies will end up being applied to the wrong location. To correct this, this command can be used to specify the Right Ascension and Declination of the original pointing center of the observation.

polvec *scale, icut, pcut, dx, dy*

Configure how polarization vectors are displayed.

EXAMPLE

```
0>polvec 5.0, 0.3, 0.0, 2, 2
Give polarization vectors lengths of 5 milli-arcsec/Jy.
Draw vectors where unpolarized flux > 0.3 Jy and polarized flux > 0 Jy.
Draw polarization vectors in every 2nd X pixel and in every 2nd Y pixel.
```

All but the first of the above lines are responses from the polvec command.

PARAMETERS

scale - Specify the length per Jy of each vector on the map. This length is specified in the units specified by the last call to the 'mapunits' command.

The initial default is 0.0, which results in an error message being emitted by the mapplot command when it is asked to draw a vector plot.

icut - Default = 0.0 Jy.
The minimum flux of the underlying single polarization map for which vectors should be drawn.

pcut - Default = 0.0 Jy.
The minimum polarized intensity for which vectors should be drawn.

dx,dy - Default = 1,1.
Polarization vectors will be drawn at most every this number of pixels horizontally and vertically. Thus by default vectors will be drawn in every pixel that satisfies the icut and pcut flux constraints.

CONTEXT

Given an observation with Q and U polarizations, the 'mapplot' command has the capability to overlay polarization vectors over the map of the currently selected polarization. Normally one selects I polarization before running mapplot in this mode, so that the total intensity map is displayed overlaid with polarization vectors.

The 'polvec' command configures the general attributes of these polarization vectors, allowing specification of their length per Jy of polarized flux density, how bright the underlying map or the polarized flux must be before a vector should be drawn in a given pixel, and

whether to subsample the available pixels to avoid overcrowding.

RELATED COMMANDS

mapplot - Display map and beam images.
multi_model - Enable or disable multi-model mode.

primary_beam *samples, resolution, frequency*

Specify a primary beam to use on all baselines.

EXAMPLE

To install a simple gaussian approximation of a primary beam, you can use the `gauss_array()` function to provide the array of primary beam samples.

```
0>mapunits arcmin
0>primary_beam gauss_array(45.1, 4.0, 0.5), 0.5, 30.0e9
```

Where 45.1 is the FWHM of the primary beam (in arcminutes in this case), 4.0 specifies how far out to sample the primary beam, expressed as the number of sigma, and 0.5 is the resolution along the x-axis of the gaussian. 30.0e9 is the frequency at which the primary beam is being specified, (30GHz in this case).

For non-gaussian primary beams, you must specify the array of samples explicitly. For example, to install the above gaussian beam without using the `gauss_array()` function, you could do the following:

```
0>mapunits arcmin
! Convert the FWHM of the gaussian to the equivalent standard deviation.
0>float sigma
0>sigma = 0.4246609001 * 45.1
0>primary_beam exp(-0.5 * (ramp(0, sigma*4, 0.5)/sigma)^2), 0.5, 30.0e9
0>...
```

Breaking this down, note that the `ramp()` function,

```
ramp(0, sigma*4, 0.5)
```

returns an array of values which linearly increment by 0.5 from 0 to `sigma*4`. The resulting array is then divided by `sigma`, by writing:

```
ramp(0, sigma*4, 0.5) / sigma
```

This returns another array, which is then squared to produce another array:

```
(ramp(0, sigma*4, 0.5) / sigma)^2
```

Finally this is multiplied by -0.5 and passed to the `exp()` function:

```
exp(-0.5 * (ramp(0, sigma*4, 0.5)/sigma)^2)
```

This returns another array, which is the required gaussian array sampled out to 4 standard deviations.

PARAMETERS

- samples - An array of values which sample the primary beam.

- resolution - The angular distance on the sky between samples.
 The units can be changed with the 'mapunits'
 command. If the mapunits command hasn't been used,
 the units are milli-arcseconds.

- frequency - The frequency at which the beam in the samples
 array is being specified, measured in Hz. Note
 that the primary beam for other frequencies is
 computed under the assumption that the width of
 the primary beam scales inversely proportional
 to frequency.

CONTEXT

Where the antennas that make up an interferometer have diameters that are comparable to the distance between them, the field of view of the interferometer is limited by the diameters of the point spread functions of the individual antennas, which we will refer to as voltage beams. The field of view is limited to the diameter of the product of the voltage beams of the individual antennas of each baseline. This product is known as the primary beam. In the case where all of the antennas are identical and aren't shadowed by neighboring antennas, all baselines have the same primary beam, and the resulting map is effectively multiplied by this beam, centered at the original pointing center. If one cleans or modelfits a dataset from such an interferometer, without taking account of the primary beam, model components away from the center of the primary beam have lower fluxes than their true values. If alternatively, one tells difmap about the primary beam, difmap will thereafter appropriately scale up the fluxes of new model components according to their distance from the pointing center.

projplot *pa, station, uvmin, uvmax, ampmin, ampmax, phsmin, phsmax, docursor, deltapa*

Plot visibility amplitudes and/or phases versus projected UV distance.

EXAMPLE

```
0> projplot 45
```

```
0>
```

This example plots baseline amplitude and phase versus UV distance along a radial line 45 degrees North through East. Model data if present is plotted in red. If the display device has a cursor then interactive mode will be invoked after the initial display has been completed.

PARAMETERS

- pa** - (Optional: default = 0.0)
 The position angle of the radial line along which the
 UV distance is to be measured, given in degrees North
 through East.
- station** - (Optional) The name of a station to highlight in the
 plot. If not given or given as an empty string "", no
 station will initially be highlighted. (Stations
 can be specified subsequently with the cursor).
- uvmin** - (Optional) Default=0.
 The minimum UV radius to display.
 Unless changed with the mapunits command, the units are
 millions of wavelengths.
- uvmax** - (Optional) Default=0. (0 specifies the whole range).
 The maximum UV radius to display.
 Unless changed with the mapunits command, the units are
 millions of wavelengths.
- ampmin** - (Optional) Default=0.
 The minimum amplitude to display.
- ampmax** - (Optional) Default=0.
 The maximum amplitude to display.
 If you want to set later arguments while keeping the
 default amplitude range set both ampmin and ampmax
 to 0.
- phsmin** - (Optional) Default=-180.
 The minimum phase to be displayed (degrees).

- phsmax - (Optional) Default=+180.
 The maximum phase to be displayed (degrees).
 If you want to set later arguments while keeping the
 default amplitude range set both phsmin and phsmax
 to 0.
- docursor - (Optional) Default=TRUE.
 If the display device has a cursor but you don't want
 to enter the interactive cursor mode, make this FALSE.
- deltapa - (Optional) Default=-1
 When a positive angle that is less than 90 degrees is specified,
 this tells projplot to only plot visibilities that lie within
 deltapa degrees of the current projection angle, pa, in the
 UV plane.

CONTEXT

This command is almost identical to the radplot command, except that rather than displaying data versus radial distance from the origin, projplot displays the same data versus the UV distance parallel to a given position angle. This is useful for the class of sources that are dominated by structure along a given position angle. Such sources include double sources and sources with linear jets. When projplot is used to plot such data at the same position angle as the dominant source structure then its effectively 1-dimensional structure is revealed. In particular, for a double source, when the projection angle in projplot matches the position angle of the line connecting the two sources, you should see decaying cos-squared fringes in the amplitude plot and a linear ramp in phase. Setting the projection angle accurately is essential to seeing such structure, so projplot provides interactive angle increment and decrement keys to help search for the best angle (see below). Apart from the obvious educational and essthetic uses of such a display, when used with linearly oriented structures, projplot has the advantage over radplot of displaying data in a less messy form in which bad data can more easily be identified.

Note that by default, projplot projects all of the available visibilities along the specified direction. This is great for a source that is dominated, for example by a jet in one direction. However it can lead to a plot that is hard to interpret when there are multiple significant features in diverse directions. For cases such as these, the deltapa argument can be used to limit the plotted visibilities to those that fall within a few degrees of the chosen projection direction.

INTERACTIVE MODE

If your display device has a cursor, then there are a number of interactive options that you can use. These are selected by moving the

cursor into the plot window, and pressing the following keys:

- X - Quit the projplot session, and return to the difmap prompt.
- L - Re-display the plot.
- . - Toggle the marker symbol used to display the data between the smallest available and the next largest. By default the smallest possible marker symbol is used for speed, but if the data are sparse then the larger marker symbol may be more convenient.
- H - (Help) Briefly list all keyboard options.
- n - Highlight visibilities from the next telescope in the currently highlighted sub-array. Initially no telescope is highlighted. If this key then pressed, the visibilities of the first telescope of the first sub-array will be highlighted. Pressing this key again highlights the second telescope, etc.. When the last telescope of the current sub-array has been plotted, the first telescope of the next sub-array will be plotted. So by repeatedly pressing this key you will eventually have seen each telescope of all sub-arrays individually highlighted.
- p - This has the reverse the effect to the 'n' key, highlighting the visibilities of the previous telescope in the current sub-array, and wrapping back to the last telescope in the previous sub-array, after the first telescope of the current sub-array is highlighted.
- N - [Upper-case n] Similar to the lower-case 'n' key this key results in the first telescope of the next sub-array in the observation to be highlighted. In this way you can jump between sub-arrays without having to see all the telescopes of the intermediate sub-arrays highlighted one at a time.
- P - [Upper-case p] This has the reverse effect of the 'N' key, allowing one to directly step backwards through sub-arrays.
- T - To have a specific telescope of a specific sub-array highlighted without having to step through all the intermediate telescopes and sub-arrays, press this key. On the text window you will then be prompted for the name of the telescope to be highlighted. Telescopes are specified by an optional sub-array number delimited by a colon, followed by an unambiguous telescope name prefix. If the sub-array number and colon are omitted then the sub-array from which telescopes are currently being highlighted will be assumed. For instance, both 3:BONN and BON refer to a telescope named BONN in sub-array 3

if another telescope of sub-array 3 is currently being highlighted. Also, provided that sub-array 2 does not contain more than one telescope starting with the prefix JBN, 2:JBN is sufficient to refer to a telescope called JBNK1 in sub-array 2.

- S - This key invokes a report on the baseline and time of the nearest point to the cursor. The result is displayed on your terminal. The actual data point is unaffected.
- A - This key (left-mouse-button) flags the nearest data-point to the cursor. Note that the preferred method for editing points is through the vplot command in station editing mode. Individual points should only be flagged in projplot if they are clearly singled out as being extremely bad, as opposed to being part of a group of bad points manifesting a bad telescope. If you delete too many points singly, rather than by station you may find that subsequent self-calibration will produce wild solutions due to incomplete or ambiguous un-connected closure triangles.
- C - This initiates cut area selection, through which a rectangular area may be selected within which to flag data. After pressing 'C' you will be expected to enter the locations of two opposite corners of the area, by pressing 'A' (left-mouse-button) over the corresponding points of the plot. To cancel a half selected area press 'D' (central-mouse-button). Once two corners have been selected, all visibilities displayed in that area will be flagged and will then disappear from the plot.
- W - Toggle spectral-line based editing. By default, edits are applied to all polarizations over the range of spectral-line channels requested in the last use of the 'select' command. The 'W' key toggles between applying edits to the currently selected spectral-line channels, and to all spectral-channels. The mode line above the plot shows the spectral-line channel range being edited.
- U - Select a new UV-distance display range. Having pressed this key, you will be expected to either press it a second time to select the full available UV-distance range (and enable future autoscaling), or to select a new range by pressing the 'A' key twice, once at each of the two UV-radii that enclose the desired range. To abort the selection press the 'D' key (middle mouse button) and the display will remain unchanged.
- Z - Select a new amplitude or phase display range. Having pressed this key, you will be expected to either press it a second time to select the full available amplitude and phase ranges (and enable future autoscaling), or to select a new range by pressing the 'A' key twice, once at each of the two amplitudes

or phases that enclose the desired range. To abort the selection press the 'D' key (middle mouse button) and the display will remain unchanged.

Keys used to interactively change the projection angle:

- ? - If you press this key then in the terminal window that you invoked projplot from, you will be prompted to enter a new projection angle in degrees. Having done so the display will be updated for the new projectio angle.
- < - This key decrements the currently selected projection angle by 5 degrees and then updates the display for the new angle.
- > - This key decrements the currently selected projection angle by 5 degrees and then updates the display for the new angle.

Display mode options. The following keys each change how the data are displayed. In order that you can change more than one attribute at a time, they may be grouped by pressing one key after the other. When you have specified all the attributes that you want to, press the RETURN key and the display will be refreshed.

- M - This key toggles whether the model is plotted. Model amplitudes/phases are plotted as red dots.
- 1 - Only display amplitudes versus UV radius.
- 2 - Only display phases versus UV radius.
- 3 - Display both amplitudes and phases in vertically adjacent plots.
- E - In addition to the amplitude and/or phase plots that are currently displayed, display a plot of the statistical amplitude errors versus projected UV radius. Note that if the weight scale factor isn't unity (see the "help wtscale"), then these errors will be proportional but not equal to the statistical errors.
- + - This key toggles whether the cursor is shown as a small cross or as a cross-hair that extends the full width and height of the display. Currently only the /xserve and /xwindow PGPLOT devices support this feature.
- - Toggle between the default of displaying absolute amplitudes and/or phases, and the option of displaying the amplitudes and/or phases of the vector difference between the data and the current model.

CONFIGURATION VARIABLES

 The display-mode option keys listed above may be preset in the variable, 'pflags'. If pflags is an empty string, "", then the default option string "m3" is substituted. This requests that the otherwise optional plotting of the model be selected by default and that both amplitudes and phases be displayed. To override this default and have projplot start up by displaying just amplitudes, on the command line one would type:

```
0> pflags="m1"
```

If you always want to override the default, then place such an assignment line in your difmap login file (the file named in the optional DIFMAP_LOGIN environment variable).

OTHER DISPLAY COMMANDS

```
radplot -   Display visibilities amp/phase vs. UV radius.
uvplot  -   Display the sampling of the UV plane.
vplot   -   Display observed and model visibilities from the current IF.
tplot   -   Display the time-sampling of each telescope of the current IF.
corplot -   Display the self-cal corrections of the current IF.
cpplot  -   Display observed and model closure phases interactively.
```

rad_to_map *distance*

Convert an image plane distance from radians to map units.

EXAMPLE

The following example converts 0.003 radians to arcminutes.

```
0>mapunits arcmin
Distances in the map plane now have units: arcmin.
Distances in the UV plane now have units: kilo-wavelengths.
0>print rad_to_map(0.003)
10.3132
0>
```

PARAMETERS

distance - A distance in the map plane, expressed in radians.

CONTEXT

The 'mapunits' command allows the user to specify which units should be used both for specifying distances, such as the cell sizes, of maps, and for printing such distances.

The 'rad_to_map()' function allows the user to convert a distance in radians, to the currently selected map units.

RELATED COMMANDS

```
map_to_rad - Convert an image plane distance from map units to radians.
uv_to_wav  - Convert a UV plane distance from UV units to wavelengths.
wav_to_uv  - Convert a UV plane distance from wavelengths to UV units.
```

radplot *station, uvmin, uvmax, ampmin, ampmax, phsmin, phsmax, docursor*

Plot visibility amplitudes and/or phases versus UV radius.

EXAMPLE

```
0> radpl "BONN"
```

```
0>
```

This example plots baseline amplitude against UV radius, plotting all but the data in the first sub-array associated with BONN in green. BONN data is plotted in a different color to highlight it. Model data if present is plotted in red. If the display device has a cursor then interactive mode will be invoked after the initial display has been completed.

PARAMETERS

station - (Optional) The name of a station to highlight in the plot. If not given or given as an empty string "", no station will initially be highlighted. (Stations can be specified subsequently with the cursor).

uvmin - (Optional) Default=0.
The minimum UV radius to display.
Unless changed with the mapunits command, the units are millions of wavelengths.

uvmax - (Optional) Default=0. (0 specifies the whole range).
The maximum UV radius to display.
Unless changed with the mapunits command, the units are millions of wavelengths.

ampmin - (Optional) Default=0.
The minimum amplitude to display.

ampmax - (Optional) Default=0.
The maximum amplitude to display.
If you want to set later arguments while keeping the default amplitude range set both ampmin and ampmax to 0.

phsmin - (Optional) Default=-180.
The minimum phase to be displayed (degrees).

phsmax - (Optional) Default=+180.
The maximum phase to be displayed (degrees).
If you want to set later arguments while keeping the

default amplitude range set both phsmin and phsmax to 0.

docursor - (Optional) Default=TRUE.
If the display device has a cursor but you don't want to enter the interactive cursor mode, make this FALSE.

CONTEXT

This task is useful for getting an over-all look at the whole data set in one plot. By default it plots the amplitude of visibilities of all IFs versus radial distance from the origin of the UV plane. Optionally it can also be used to display visibility phases and statistical errors instead or in addition to the plot of amplitudes. See below for details.

For each baseline, one point is plotted per IF. Where multiple spectral-line channels are selected from a given IF, the visibility is the average of those channels and is displayed at the corresponding mean UV radius.

It is often possible to get a vague idea of how the map will appear from looking at this plot. Unresolved point sources appear as flat amplitude distributions. Resolved sources display one or more gaussian fall-off distributions. Multi-component sources beat against each other, modulating the gaussian fall-off. In some cases you may be able to get a better impression by running the alternate 'projplot' command. See help projplot for details.

If the distribution is reasonably smooth, it is also possible to see the effects of poor calibration. The highlighting function is invaluable here. Highlighting each station in turn will eventually reveal the identity of a bad station or baseline. The 's' option key described below provides a more direct way of identifying the origin of discrepant points.

INTERACTIVE MODE

If your display device has a cursor, then there are a number of interactive options that you can use. These are selected by moving the cursor into the plot window, and pressing the following keys:

X - Quit the radplot session, and return to the difmap prompt.

L - Re-display the plot.

. - Toggle the marker symbol used to display the data between the smallest available and the next largest. By default

the smallest possible marker symbol is used for speed, but if the data are sparse then the larger marker symbol may be more convenient.

- H - (Help) Briefly list all keyboard options.

- n - Highlight visibilities from the next telescope in the currently highlighted sub-array. Initially no telescope is highlighted. If this key then pressed, the visibilities of the first telescope of the first sub-array will be highlighted. Pressing this key again highlights the second telescope, etc.. When the last telescope of the current sub-array has been plotted, the first telescope of the next sub-array will be plotted. So by repeatedly pressing this key you will eventually have seen each telescope of all sub-arrays individually highlighted.

- p - This has the reverse the effect to the 'n' key, highlighting the visibilities of the previous telescope in the current sub-array, and wrapping back to the last telescope in the previous sub-array, after the first telescope of the current sub-array is highlighted.

- N - [Upper-case n] Similar to the lower-case 'n' key this key results in the first telescope of the next sub-array in the observation to be highlighted. In this way you can jump between sub-arrays without having to see all the telescopes of the intermediate sub-arrays highlighted one at a time.

- P - [Upper-case p] This has the reverse effect of the 'N' key, allowing one to directly step backwards through sub-arrays.

- T - To have a specific telescope of a specific sub-array highlighted without having to step through all the intermediate telescopes and sub-arrays, press this key. On the text window you will then be prompted for the name of the telescope to be highlighted. Telescopes are specified by an optional sub-array number delimited by a colon, followed by an unambiguous telescope name prefix. If the sub-array number and colon are omitted then the sub-array from which telescopes are currently being highlighted will be assumed. For instance, both 3:BONN and BON refer to a telescope named BONN in sub-array 3 if another telescope of sub-array 3 is currently being highlighted. Also, provided that sub-array 2 does not contain more than one telescope starting with the prefix JBN, 2:JBN is sufficient to refer to a telescope called JBNK1 in sub-array 2.

- s - This key invokes a report on the baseline and time of the nearest point to the cursor. The result is displayed on your terminal. The actual data point is unaffected.

- S - This key initiates the selection of a set of visibilities for which to work out the scalar mean and scatter of the corresponding amplitudes and phases. After pressing 'S', you will be expected to enter the locations of two opposite corners of the area, by pressing 'A' (left-mouse-button) over the corresponding points of either the amplitude or phase plots. To cancel a half selected area press 'D' (central-mouse-button). Once two corners have been selected, the amplitude and statistics of the visibilities within that area will be displayed on your terminal. Note that amplitudes and phases are not Gaussian distributed, so the means will be biased. To get unbiased averages, use the 'V' key option instead, to compute vector averages.

- V - This key initiates the selection of a set of visibilities for which to work out the vector mean and scatter of the corresponding real and imaginary parts. After pressing 'V', you will be expected to enter the locations of two opposite corners of the area, by pressing 'A' (left-mouse-button) over the corresponding points of either the amplitude or phase plots. To cancel a half selected area press 'D' (central-mouse-button). Once two corners have been selected, the statistics of the real and imaginary parts of the visibilities within that area will be displayed on your terminal. The mean amplitude and phase, can of course be computed from these values.

- A - This key (left-mouse-button) flags the nearest data-point to the cursor. Note that the preferred method for editing points is through the vplot command in station editing mode. Individual points should only be flagged in radplot if they are clearly singled out as being extremely bad, as opposed to being part of a group of bad points manifesting a bad telescope. If you delete too many points singly, rather than by station you may find that subsequent self-calibration will produce wild solutions due to incomplete or ambiguous un-connected closure triangles.

- C - This initiates cut area selection, through which a rectangular area may be selected within which to flag data. After pressing 'C' you will be expected to enter the locations of two opposite corners of the area, by pressing 'A' (left-mouse-button) over the corresponding points of the plot. To cancel a half selected area press 'D' (central-mouse-button). Once two corners have been selected, all visibilities displayed in that area will be flagged and will then disappear from the plot.

- W - Toggle the scope of edits with respect to spectral-line channels. Each point in the plot comes from a single IF, and its UV radius in the plot is determined by the mean frequency of the channels currently selected from that IF. By default, if

you select such a point to be flagged, then all spectral-line channels of the source IF will be flagged, irrespective of the channels which actually went into making the plot. The alternative is to edit just the channels that went into making the plot. The 'W' key toggles which of the two options is used, and a mode line above the plot describes which option is currently in effect.

- U - Select a new UV-radius display range. Having pressed this key, you will be expected to either press it a second time to select the full available UV-radius range (and enable future autoscaling), or to select a new range by pressing the 'A' key twice, once at each of the two UV-radii that enclose the desired range. To abort the selection press the 'D' key (middle mouse button) and the display will remain unchanged.
- Z - Select a new amplitude display range. Having pressed this key, you will be expected to either press it a second time to select the full available amplitude range (and enable future autoscaling), or to select a new range by pressing the 'A' key twice, once at each of the two amplitudes that enclose the desired range. To abort the selection press the 'D' key (middle mouse button) and the display will remain unchanged.

Display mode options. The following keys each change how the data are displayed. In order that you can change more than one attribute at a time, they may be grouped by pressing one key after the other. When you have specified all the attributes that you want to, press the RETURN key and the display will be refreshed.

- M - This key toggles whether the model is plotted. Model amplitudes/phases are plotted as red dots.
- 1 - Only display amplitudes versus UV radius.
- 2 - Only display phases versus UV radius.
- 3 - Display both amplitudes and phases in vertically adjacent plots.
- E - In addition to the amplitude and/or phase plots that are currently displayed, display a plot of the statistical amplitude errors versus UV radius. Note that if the weight scale factor isn't unity (see the "help wtscale"), then these errors will be proportional but not equal to the statistical errors.
- + - This key toggles whether the cursor is shown as a small cross or as a cross-hair that extends the full width and height of the display. Currently only the /xserve and /xwindow PGPLOT devices support this feature.

- - Toggle between the default of displaying absolute amplitudes and/or phases, and the option of displaying the amplitudes and/or phases of the vector difference between the data and the current model.

CONFIGURATION VARIABLES

The display-mode option keys listed above may be preset in the variable, 'rflags'. If rflags is an empty string, "", then the default option string "m1" is substituted. This requests that the otherwise optional plotting of the model be selected by default and that only amplitudes be displayed. To override this default and have radplot start up by displaying both amplitudes and phases, on the command line one would type:

```
0> rflags="m3"
```

If you always want to override the default, then place such an assignment line in your difmap login file (the file named in the optional DIFMAP_LOGIN environment variable).

OTHER DISPLAY COMMANDS

```
projplot -   Display visibilities amp/phase vs. projected UV distance.
uvplot    -   Display the sampling of the UV plane.
vplot     -   Display observed and model visibilities from the current IF.
tplot     -   Display the time-sampling of each telescope of the current IF.
corplot   -   Display the self-cal corrections of the current IF.
cpplot    -   Display observed and model closure phases interactively.
```

`read_models` *filename*

Read a table of models, indexed by channel-range/polarization, from a file.

EXAMPLE

```
0>read_models 3c84.mtab
Read 212 model components for stokes LL, channels 1, 10
Read 176 model components for stokes U, channels 1, 10
```

PARAMETERS

`filename` - The name of the file to read the models from. This should be a file previously written by `write_models`.

CONTEXT

In multi-model mode (see `help multi_model`), different models are associated with different channel-range/polarizations selections. This command, in conjunction with the `write_models` command allow one to save and restore these models. In particular, they are used by the 'save' command for this purpose. Each model in the files written by `write_models`, is written in the same format as by the 'wmodel' command, with the exception that each model is preceded by a select line, specifying the polarization and channel range to which the model refers.

RELATED COMMANDS

`multi_model` - Switch to and from multi-model mode.
`write_models` - Save the current set of multi-model models to a file.
`clear_models` - Discard all multi-model models.

resoff *baseline_name*

Determine and apply a baseline based amplitude and phase correction.

EXAMPLE

To correct for correlator offsets on baseline BONN-WSRT of the first subarray.

```
0> resoff "BON-WSRT"
1:BONN-WSRT: 0.945151 amplitude change, -0.399408 degrees phase offset
0>
```

To correct for correlator offsets on baseline HSTK-OVRO of the 3rd sub-array.

```
0> resoff 3:HSTK-OVRO
3:HSTK-OVRO: 0.988347 amplitude change, -3.50419 degrees phase offset
0>
```

To correct all baselines of sub-array 2.

```
0> resoff 2
2:WSRT-JBNK: 0 amplitude change, -5.63816 degrees phase offset
2:WSRT-BOLOGNA: 0 amplitude change, 13.0693 degrees phase offset
2:WSRT-HSTK: 0 amplitude change, -0.687232 degrees phase offset
2:JBNK-BOLOGNA: 0 amplitude change, 12.97 degrees phase offset
2:JBNK-HSTK: 0 amplitude change, 2.05668 degrees phase offset
2:BOLOGNA-HSTK: 0 amplitude change, 2.33325 degrees phase offset
```

To correct all baselines.

```
0> resoff
1:BONN-WSRT: 0.945151 amplitude change, -0.399408 degrees phase offset
.....etc.....
```

PARAMETER

```
baseline_name - Default=""
                The name of the baseline(s) to be corrected. See the
                antenna_names help topic for a discussion of the
                syntax and interpretation of baseline specification
                arguments.
```

CONTEXT

The use of this command is NOT recommended except in exceptional circumstances. The only recommended use of this command is for inspection to determine whether a data-set contains closure offsets with respect to a given model - in which case it should be followed by

clroff to remove the closure offsets instated. In very high dynamic range mapping it can be used to fix baseline dependent amplitude and phase "non-closing" errors. On other data-sets it is just as likely to introduce new non-closing errors and should not be used.

resoff should only be used after all possible cleaning and self-calibration has been performed and then only when the data appear to fit very well on all but one or two baselines. Visual inspection of the data is essential before embarking on this coarse, so use the vplot and radplot commands first. If the miscreant baselines are short baselines, then the real cause of the errors is undoubtedly an incomplete model, possibly due to insufficient sampling at short UV spacings. In such cases use of resoff is not justified.

resoff determines and applies a single amplitude and phase correction for the requested baseline. The amplitude correction is determined as the ratio of the weighted means of the model and observed amplitudes on the requested baseline, over the whole observation. The phase correction is the weighted mean phase difference between the model and observed phases over the same range.

Be warned, unlike self-calibration, resoff is not constrained by any closure or other constraints. It will always make a map look more like the model but at the expense of reality if the model isn't correct.

SIDE EFFECTS

Resoff establishes the latest clean model when necessary. It also invalidates the current residual map, which will be re-calculated by the next command that needs it.

RELATED COMMANDS

clroff - Undo the effects of all resoff commands in this difmap session.
fix_weights - Set whether amplitude corrections affect the visibility weights.

restore *bmin, bmaj, bpa, omit_residuals, do_smooth*

Add the CLEAN model to the residual map to form a clean map.

EXAMPLE

```
O>restore 1.8, 6.1, 27
```

```
Restoring with beam: 1.8 x 6.1 at 27 degrees (North through East)
```

```
Clean map min=-0.050992 max=3.324691 Jy/beam
```

```
O>
```

All but the first line are responses from the 'restore' command and the client commands that it calls.

PARAMETERS

- | | |
|----------------|--|
| bmin | - The minor axis full-width-at-half-maximum of the restoring beam (clean beam). Unless changed with the mapunits command, the units are milli-arcsec. If this is the only parameter specified then a circular beam with bmaj=bmin and bpa=0 will be assumed. |
| bmaj | - The major axis full-width-at-half-maximum of the restoring beam (clean beam). Unless changed with the mapunits command, the units are milli-arcsec. |
| bpa | - The position angle of the beam measured from north going towards east. ie. a positive angle denotes an anti-clockwise rotation on the map when displayed by 'mapplot'. If this argument is omitted then 0 will be assumed. |
| omit_residuals | - Default = false
Normally the clean model is superposed on top of the residual map. Alternatively the residual map can be omitted by specifying a value of true for the omit_residuals argument. The result will be an image of the clean model. |
| do_smooth | - Default = true
By default the residual map is smoothed with a gaussian of width 3x3 pixels before the clean model is added to it. This is a common way of compensating for the fact that the model is convolved with a smooth gaussian beam, while the residual noise is convolved with the dirty beam. To prevent this behavior specify a value of false for the do_smooth argument. |

If no parameters are specified 'restore' will either use the values that were given to it the previous time it was invoked, or if the weighting used by invert has been changed since the last 'restore', 'restore' will use the estimated beam size from the last time that 'invert' inverted the beam. Commands that force 'restore' to discard the previous beam parameters and use the estimate from invert, include: observe, uvtaper, uvrange, uvzero, uvweight.

To force restore to fall back on the estimated beam size generated by the last invert, type

```
0> restore 0
```

Examples are:

```
0> restore 0    Always restores with the latest estimate from 'invert'.
```

```
0> restore      Uses whatever restore used last time unless the dirty
                  beam produced by invert has been changed by changes in
                  weighing etc..
```

```
0> restore 1.5,3.2,-10.5  Restores with an elliptical beam of
                           bmin=1.5, bmaj=3.2 at -10.5 degrees.
```

```
0> restore 1.5          Restores with a circular beam of size 1.5.
```

```
0> restore 1.5,3.2      Restores with an ellipse of major-axis position angle 0.
```

CONTEXT

This is usually the final step in mapping a source. It can also be done at any time while mapping, but since it over-writes the current residual map you will incur the overhead of re-inverting the UV data on the next clean.

Restore generates an image of the CLEAN-model components convolved with an elliptical gaussian beam and adds this to the current residual map. This is the final map.

After restoring the map, it may be displayed with mapplot (the first argument to mapplot must be cln). 'mapplot' allows contours to be plotted on restored maps and will also plot a shaded representation of the restoring beam in one corner.

For example:

```
0> mapplot cln
```

This plots the restored map. Note that if the map isn't restored when

you type this or the UV data has been changed since you restore'd, mapplot will call restore for you with no arguments. Also note that if you neglect to write cln as the first argument to mapplot, then mapplot will assume that you want to plot the residual map and will re-invert it for you - take care!

SIDE EFFECTS

Since the restored map over-writes the residual map array, the residual map is thereafter useless for continued cleaning and will be re-inverted by the next clean. This saves memory at the expense of CPU time.

rmodel *file_name, do_continuum*

Read a new CLEAN model from a VLBI model file.

EXAMPLE

```
O>rmodel test.mod
```

```
A total of 26 model components were read from file test.mod
```

```
O>
```

PARAMETERS

`file_name` - The name of an existing model file.

`do_continuum` - (Default=false)

If omitted or given as FALSE, the contents of the model file will supersede the established and tentative models. If it is given as TRUE, then the contents will be used to supersede the continuum model.

CONTEXT

There are at least two reasons why one might want to read in an external a model file. The first is to provide a starting model for self-calibration - this is best done using the 'startmod' command. The second is to let one re-start mapping where one left off at an earlier date.

A third reason that is relevant to spectral-line work, is to read in a model from a separate observation of the continuum underlying a spectral-line source. The model produced by the continuum observation can then be used as a model to be subtracted prior to spectral-line mapping. See the 'setcont' command for further details.

SIDE EFFECTS

The models read supersede any existing models of the same type. Thus, if `do_continuum` is FALSE, the current established and tentative models will be discarded before reading the new model. If `do_continuum` is TRUE, then the current continuum model will be discarded before reading the new model. So, if you wish to retain these models for later, remember to use the `wmodel` command before using `rmodel`.

RELATED COMMANDS

`startmod` - Self-calibrate an observation with a temporary point-source model.

`wmodel` - Write clean models to a VLBI model file.

`showmodel` - Report detailed information about the current models.

`save` - Save all data files and the difmap running parameters.

rwins *winfile*

Read windows previously saved CLEAN windows from a text file.

EXAMPLE

```
O>rwins "myfile.win"
rwins: Read 9 windows from myfile.win
O>
```

PARAMETERS

winfile - The name of a file containing previously saved windows.

CONTEXT

CLEAN windows are set interactively in mapplot and/or directly via the addwin command. The resulting list of windows may be saved in a file using the 'wwins' command and retrieved later using the 'rwins' command. This makes it possible to use the same windows for data at a different epoch or to continue cleaning a data-set at a later date using the same windows.

The window file is a text file. Each line in the text file corresponds to a single CLEAN window and should contain 4 numbers separated by spaces. The first two numbers specify the relative Right-Ascension of the two vertical window edges (either one first) wrt the center of any map. The second two specify the relative Declination of each horizontal window edge (either one first) wrt the center of any map. All positions are measured in milli-arcseconds and use the same sign conventions as displayed by the mapplot command.

SIDE EFFECTS

Any existing list of windows will be deleted before reading from the file.

RELATED COMMANDS

wwins - Write CLEAN windows to a text file.

save prefix_name

Save UV data, models, windows, the restored map and a command file.

EXAMPLE

```

O>save "3c345"
Writing UV FITS file: 3c345.uvf
Writing 94 model components to file: 3c345.mod
wwins: Wrote 3 windows to 3c345.win
Writing map to FITS file: 3c345.fits
Writing difmap environment to: 3c345.par
O>

```

PARAMETERS

prefix_name - The prefix to give all the output file names - see the example above.

STANDARD SUFFIXES

To the prefix name that you enter the following extensions are appended to indicate the file type:

```

.uvf - The suffix for the UV FITS data file.
.mod - The suffix for any model file.
.win - The suffix for any windows file.
.fits - The suffix for any restored map fits file.
.par - The command-file containing commands to restore the
       running parameters and data-files into difmap on a later
       occasion.

```

CONTEXT

Before leaving difmap it is important to save all processed data, unless the data is to be discarded. The 'save' command provides a convenient method for saving all relevant data with a single command. It also now saves the running parameters in a command file.

'save' calls the wobs, wmodel, wwins and wmap commands for you where appropriate, to save the UV data, any models, any windows and any restored map that has or can be created. Note that if necessary, when there is a model to restore with, the map will be restored for you.

The 'save' command also writes a command file that may be used to restore the files and difmap state at a later date.

There are now two methods for reading the data back into difmap at a

later date. Originally the 'get' command was provided for this purpose, but the get command could only restore the data files - it did not restore the running parameters. Now the save command has been modified to create an extra file - the .par file. This is a command file that contains commands to restore the data files into difmap AND to restore the running parameters.

Thus the preferred method for retrieving the state of difmap at a later date, is to execute this command file.

In the example given above a file 3c345.par was created. To restore the files and running parameters at a later date one would type:

```
O>@3c345.par
```

The response in this case would be something like:

```
Reading UV FITS file: 3c345.uvf
The integration time appears to be 1.31321 seconds.
There appear to be 485 integrations.
AN table 1: 485 integrations on 55 of 55 possible baselines.
Found source: 3C345
```

```
Number of spectral-line channels: 1
```

IF	Freq (Hz)	Chan	BW (Hz)	Full BW (Hz)	Sideband
01	1.66299e+09		2e+06	2e+06	USB

```
Polarization(s): RR
```

```
Read 192 lines of history.
```

```
Reading an estimated 26675 visibilities.
Selecting channels: 1 -> 1, polarization: RR.
A total of 94 model components were read from file 3c345.mod
rwins: Read 3 windows from 3c345.win
New map grid = 256x256 pixels with 0.5x0.5 milli-arcsec cellsize
Gaussian UV-taper cancelled
UV range cancelled
New uniform weighting binwidth=2 pixels
Amplitude error weighting cancelled
Radial weighting cancelled
Zero-baseline flux cancelled
Self-cal gaussian UV-taper cancelled
Amplitude corrections will not be limited
Phase corrections will not be limited
The mapplot display area will be limited to:
  Relative RA: (The whole available range)
  Relative DEC: (The whole available range)
```

- In phase-only self-cal, good data on baselines that are not in closed arrays of at least 3 telescopes will be flagged.
- In amplitude self-cal, good data on baselines that are not in closed arrays of at least 4 telescopes will be flagged.

0>

These are the output lines of the various commands that were run from the command file.

RELATED COMMANDS

get - Restore UV data, models and windows from files.
wobs - Write the UV data to a UV FITS file.
wmodel - Write the CLEAN model to a (text) VLBI model file.
wmap - Save the restored map as a FITS file.
wbeam - Save the dirty-beam as a FITS file.
wwins - Write CLEAN windows to a text file.
exit - Exit calls save if so requested.

scangap *gap, sub_array*

Change the time gap used to delimit neighboring scans.

EXAMPLES

1. To assign a 30 minute scan separator to all sub-arrays:

```
0>scangap 30*60
Delimiting interscan gap changed to 1800 seconds in all sub-arrays.
0>
```

2. To re-instate the default (1 hour) separator:

```
0>scangap 0
Delimiting interscan gap changed to 3600 seconds in all sub-arrays.
0>
```

3. To assign a 20 minute scan separator only to sub-array 2:

```
0>scangap 20*60, 2
Delimiting interscan gap changed to 1200 seconds in sub-array 2.
0>
```

4. To see what the current gaps are set to, omit all arguments:

```
0>scangap
The delimiting interscan gap is 3600 seconds in sub-array 1.
The delimiting interscan gap is 1200 seconds in sub-array 2.
0>
```

PARAMETERS

- | | |
|-----------|--|
| gap | - The new interscan gap to assume. This is the minimum time separation between neighboring integrations at which integrations are taken to occupy different scans. When an observation is first read into difmap, all sub-arrays are assigned a scan delimiter of 3600 seconds (1 hour). This default may be re-selected by giving 'gap' as 0, as in the second example above. |
| sub_array | - Optional: Default=0
If this argument is not provided, or is given as 0, then the new scan delimiter is assigned to all sub-arrays. Otherwise, the number given specifies the sub-array to which the scan separator should be assigned. |

CONTEXT

 All difmap plotting commands that plot anything versus time, offer the option of resolving the data into multiple isolated scans. These scans are then plotted side by side with the intervening gaps removed. The plotting commands also know not to extend model lines between scans, even when scan mode is not in effect.

In order to decide what constitutes a scan, difmap simply says that any two integrations that are separated by more than a given amount of time mark the end of one scan and the start of a new scan. By default this time interval is one hour, and is a good choice for most observations. Sometimes the default is not the correct choice, and the 'scangap' command gives you the opportunity to change the default, either on a sub-array by sub-array basis, or globally for all sub-arrays.

Note that no more than 20 scans per sub-array are allowed. This means that if too short an interval is requested, the command will emit an error message ignore the requested interval.

RELATED COMMANDS

 vplot - Display observed and model visibilities from the current IF.
 tplot - Display the time-sampling of each telescope of the current IF.
 corplot - Display the self-cal corrections of the current IF.
 cpplot - Display observed and model closure phases interactively.

select *polarization, ca, cb, ...*

Select a polarization and ranges of spectral-line channels to be processed.

EXAMPLES

1. Select visibilities of polarization I over two inclusive spectral-line channel ranges.

```
0>select I, 20, 25, 47, 65
Selecting polarization: I, channels: 20..25 47..65
Reading IF 1 channels: 20..25 47..65
0>
```

2. Select continuum channels either side of a spectral-line feature that occupies the range 21 to 60:

```
0>select I, 1, 20, 60, nif*nchan
Selecting polarization: I, channels: 1..20 60..112
Reading IF 1 channels: 1..20 60..112
0>
```

3. Select channel ranges that span more than one IF. The following example is based on an observation that has three IFs, each of which has 64 channels.

```
0>select i,50,70
Selecting polarization: I, channels: 50..70
Reading IF 1 channels: 50..64
Reading IF 2 channels: 65..70
Reading IF 3 channels: (none)
0>
```

4. Channel ranges can be given in any order and can overlap. The following example shows that the select command re-organizes ranges into non-overlapping ranges.

```
0>select I, 23,45, 60,70, 25,30, 51,53, 40,50
Selecting polarization: I, channels: 23..53 60..70
Reading IF 1 channels: 23..53 60..70
0>
```

5. If the last range given is unpaired, then this denotes a range consisting of a single channel.

```
0>select I,10
Selecting polarization: I, channels: 10..10
Reading IF 1 channels: 10..10
0>
```

ARGUMENTS

polarization - (Default = The polarization last selected, or on the first call, either I if available, or the first recorded polarization.
The name of a polarization that was recorded in the UV FITS file, or one that can be derived from other recorded polarizations. Use the 'header' command to see which polarizations were recorded.
Further details are provided below.

ca, cb, ... - Default = The ranges of a previous selection, or all channels on the first call.
One or more ranges of spectral-line channels from one or more IFs are specified by giving the indexes of the first and last channels of each range, using channel indexes taken from the set [1..nif*nchan].

If an odd number of channel indexes is specified, the last index is taken to represent a range that contains a single channel.

The select command sorts channel ranges into ascending order and coalesces overlapping and contiguous ranges.

CONTEXT

In Difmap, one combination of stokes parameters and spectral-line channels is mapped at a time. This combination is known as the "current processing stream". The 'select' command allows you to specify the scope of this stream in terms of the polarization to be processed and one or more ranges of spectral-line channels. Once such a selection has been made, most difmap commands operate on this data, rather than the raw data from the UV FITS file. However edits, and 'selfcal' and 'resoff' calibrations are not restricted to the selected data, and are applied both to the raw data set and to all future selections.

For example, to map a spectral-line data-set, one would typically select one or more channels containing strong line-emission from a single IF. This would then be mapped, and in the course of doing so, self-calibration would be applied. The corrections accumulated for the channels of the selected IF would thereafter be applied to all other channel selections from that IF, unless the corrections were cleared with the 'uncal' command. Furthermore, future corrections, based on other spectral-line channels of the same IF, would build upon the existing corrections. This ensures that the relative positions of features in different channels are preserved.

Note that the 'specplot' command is independent of the current processing stream selection. This makes it a convenient way to see what channels to select before using the 'select' command. See also the spectral_line help topic.

POLARIZATION SELECTION

The polarizations recorded in the originating UV FITS file can be displayed with the 'header' command.

The polarization argument may either be one of these polarizations, or be the name of a polarization that can be derived from them.

The names of recognised polarizations are:

I Q U V RR LL RL LR XX YY XY YX PI

The following is a list of polarizations that may be constructed from other polarizations:

Name	Derived from:	Derived as:
I	RR and LL	$(RR + LL)/2$
Q	RL and LR	$(RL + LR)/2$
U	RL and LR	i. $(LR + RL)/2$
V	RR and LL	$(RR - LL)/2$
PI	RR and/or LL	(Either RR, LL or the weighted mean of RR and LL, as explained below)

The PI polarization stands for pseudo-I polarization. It is intended for observations of non-circularly polarized sources, where one can then consider LL and RR as being independent measures of I polarization. This is particularly useful for observations made with interferometers with baselines that only sample one of LL or RR. In such cases, selecting 'I' would result in flagged visibilities on these baselines, whereas selecting PI results in the unmatched LL or RR visibility being used. Similarly, since PI considers both LL and RR to be independent measures of I, on baselines that do have both RR and LL, selecting PI results in the weighted mean of these visibilities being used, rather than $(RR+LL)/2$.

CHANNEL RANGES (and IF ranges)

The channel numbers that the 'select' command understands start at 1, and end with the total number of channels in the observation. When there are multiple IFs, the latter is the sum of the number of channels in all IFs. For specification purposes, channel numbers increase seamlessly from the first IF to the last IF, regardless of the relationship between neighboring IFs and the sign of their

inter-channel frequency increments. This is the same convention used by the 'specplot' command, so if you are confused by this, use the 'specplot' command to look at the assignments of channels to IFs.

For example, consider an observation that has two IFs, each with 64 channels. In the example, the 'header' command shows the following assignment of frequencies to the IFs:

There are 2 IFs, and a total of 128 channels:

IF	Channel origin	Frequency at origin	Freq offset per channel	Number of channels	Overall IF bandwidth	
01	1	1.53575e+10	125000	64	8e+06	
02	65	1.53655e+10	125000	64	8e+06	(Hz)

In this example channel number 1 refers to the first channel of IF 1 and has a center frequency of 1.53655e+10 Hz, whereas channel number 65 refers to the first channel of IF 2 and has a center frequency of 1.53575e+10 Hz.

Note that when no channels are selected from a given IF, that IF is said to be unsampled, and it will not be processed, apart from the application of global editing. This is essential for spectral-line observations, but can also be used in continuum observations if different IFs must be mapped separately.

Note that continuum observations are treated as spectral-line observations with only one channel per IF. In such cases, the channel numbers are equivalent to IF indexes, and one can then select ranges of IF indexes directly.

IMPLEMENTATION DETAILS

The 'select' command extracts the requested data from a scratch file called uvdata.scr that contains all the raw UV data from the last UV FITS file that was read. For each baseline of each integration, 'select' derives a single visibility per IF. This is formed by vector averaging visibilities of the given polarization, over the given range of spectral-line channels. These visibilities are then stored in another scratch file called ifdata.scr, from which one IF at a time is temporarily paged into memory for display or other processing.

RELATED HELP TOPICS

spectral_line - An overview of spectral-line mapping in Difmap.
multi_IF - An overview of multi-IF mapping in Difmap.

RELATED COMMANDS

- observe - Read a random-groups UV FITS file.
- header - Use this to see what polarizations, IFs and spectral-line channels are available.
- specplot - This plots visibility spectra, and provides a convenient way to see what channel ranges to select, based upon the location of spectral features.

selfant *antenna, dofix, weight, maxamp, maxphs*

Set antenna based constraints for subsequent self-calibration.

EXAMPLES

The following example shows how the selfant command lists the current parameters of all telescopes when it is invoked with no arguments.

```
0>selfant
1:BONN      self-cal  status=correctable  weight=1 maxamp=none maxphs=none
1:WSRT      self-cal  status=correctable  weight=1 maxamp=none maxphs=none
1:JBNK1     self-cal  status=correctable  weight=1 maxamp=none maxphs=none
1:BOLOGNA   self-cal  status=correctable  weight=1 maxamp=none maxphs=none
1:TORUN     self-cal  status=correctable  weight=1 maxamp=none maxphs=none
0>
```

The following example shows how the selfant command sets just one parameter for the one specified telescope, then reports the current parameters for that telescope, including any that were previously set, but not changed by this command.

```
0>selfant BOLOGNA, true
1:BOLOGNA   self-cal  status=fixed  weight=1 maxamp=none maxphs=none
0>
```

The following example demonstrates how the selfant command overrides the selflims command for a specified telescope or telescopes, but reports the global limits specified by the selflims command for other telescopes that have not been explicitly configured using this command.

```
0>selflims 2.0, 45.0
Selfcal amplitude corrections will be limited to 0.5 -> 2.0.
Selfcal phase corrections will be limited to -45 -> 45 degrees.
0>
0>selfant JBNK1, false, 1.0, 10.0, 90.0
1:JBNK1     self-cal  status=correctable  weight=1  maxamp=10  maxphs=90
0>selfant
1:BONN      self-cal  status=correctable  weight=1  maxamp=2.0  maxphs=45
1:WSRT      self-cal  status=correctable  weight=1  maxamp=2.0  maxphs=45
1:JBNK1     self-cal  status=correctable  weight=1  maxamp=10  maxphs=90
1:BOLOGNA   self-cal  status=correctable  weight=1  maxamp=2.0  maxphs=45
1:TORUN     self-cal  status=correctable  weight=1  maxamp=2.0  maxphs=45
0>
```

PARAMETERS

antenna - (Default="")

The name of the antenna or antennas to be addressed.
 If this argument is omitted or supplied as "", then
 all antennas will be addressed. See the antenna_names help
 topic for a discussion of the syntax and interpretation of
 antenna specifications.

dofix - Normally this should be false. If true then subsequent
 self-calibration will not be allowed to change the complex
 gain of the named antenna. If this argument is omitted no
 change is made to the existing value.

weight - The relative self-cal weight correction factor to give to
 the named antenna.
 Normally this should be 1.0. This factor expresses the
 degree to which the existing weights on the named antenna
 should be believed with respect to those of other antennas.
 If this argument is omitted no change is made to the
 existing value.

maxamp - If maxamp < 1 then no limit will be placed on amplitude
 corrections for this telescope unless the selflims command
 has been used to specify a global amplitude limit.

If maxamp >=1 then for each selfcal solution-time interval,
 selfcal checks whether the amplitude corrections of any
 telescopes fall outside the range 1/maxamp to maxamp, where
 maxamp is either the telescope specific amplitude limit, if
 specified by selfant command, or the global amplitude limit
 set by the selflims command. If the amplitude correction of
 one or more telescopes exceeds these limits, then the value
 of the doclip argument of the selflims command determines
 how selfcal applies the amplitude limit. See 'help selflims'
 for details.

maxpsh - If maxpsh <= 0 then no limit will be placed on phase
 corrections for this telescope unless the selflims command
 has been used to specify a global phase limit.

If maxpsh >=0 then for each selfcal solution-time interval,
 selfcal checks whether the phase corrections of any
 telescopes fall outside the range -maxpsh to maxpsh, where
 maxpsh is either the telescope specific phase limit, if
 specified by the selfant command, or the global phase limit
 set by the selflims command. If the phase correction one or
 more telescopes exceeds these limits, then the value
 of the doclip argument of the selflims command determines
 how selfcal applies the phase limit. See 'help selflims'
 for details.

After processing the above arguments the selfant command reports the

current parameter settings for each of the specified telescopes, or all of the telescopes if none were specified. For antennas where no telescope-specific limits have been specified, then the limits specified by the selflims command will be shown instead.

CONTEXT

This command affects the way that corrections for specific antennas of specified sub-arrays are treated during subsequent self-calibration. Effected commands are selfcal, gscale and startmod.

There are many reasons why preventing the gain of one or more antennas from being changed, or from changing too much is advantageous. On the one hand it constrains the freedom of solutions of other antennas. This is useful if one has very few antennas and the antenna that is fixed is an antenna that one believes to be well calibrated. On the other hand it can be used to fix the largely unconstrained gains of a very noisy antenna, to prevent them from going wild.

Each visibility of an observation has an associated weight assigned to it. Such weights may be indirectly inspected in vplot, where they are plotted as the equivalent $1/\sqrt{\text{weight}}$ error bars. The difmap self-calibration algorithm uses them to determine the weights to be used during fitting. If the real uncertainties on all baselines appear to be equally in error by a constant factor then nothing is lost. But if the uncertainties on all baselines of a given antenna appear to be over or under-estimated with respect to those of other antennas then selfant may be used to specify that appropriately scaled versions of the weights be used in subsequent self-calibration. This is the technically correct way to use the 'weight' argument. Sometimes however, one may decide that the correct weights of a given antenna are so big compared to other antennas that the resulting self-cal solutions are dominated by visibilities of this antenna. In such cases it may be advantageous to reduce the weight of the good antenna by specifying 'weight' with a value less than 1.0.

RELATED COMMANDS

selfcal - Performs self-calibration.
 selfflag - Used to control the fate of un-correctable data.
 selftaper - Used to down-weight short baselines during selfcal.
 selflims - Used to set global limits on corrections by selfcal.
 gscal - Calculate overall telescope amplitude corrections.
 startmod - Phase selfcal to a starting model then discard the model.

selfcal *doamp, dofloat, solint*

Self calibrate for telescope phase (and amplitude) corrections.

EXAMPLES

1. Phase-only self-calibration:

```
0>selfcal
Performing phase self-cal
Adding 49 model components to the UV plane model.
The established model now contains 49 components and 5.43211 Jy

Correcting IF 1.
A total of 2187 telescope corrections were flagged in sub-array 1.

Fit before self-cal, rms=0.333969Jy  sigma=1.037002
Fit after  self-cal, rms=0.325501Jy  sigma=1.012208

Reinstating the default IF.
0>
```

2. Amplitude and phase self-calibration with a solution interval.

```
0>selfcal true,true,1440
Performing amp+phase self-cal over 1440 minute time intervals

Correcting IF 1.
A total of 222 telescope corrections were flagged in sub-array 1.

Fit before self-cal, rms=0.325501Jy  sigma=1.012208
Fit after  self-cal, rms=0.255853Jy  sigma=0.747795

Reinstating the default IF.
0>
```

All but the first line of each example are responses from 'selfcal'.

PARAMETERS

doamp - (Default=false)

If this parameter is omitted or given as false, only phase corrections are calculated and applied. Alternatively, if doamp=true telescope amplitude corrections will also be calculated and applied.

dofloat - (Default=false)

If doamp=true, then dofloat specifies how the amplitude corrections are to be applied. Initially the corrections are applied verbatim to the visibilities. If dofloat is true

then this is all that occurs. However if `dofloat` is false then the total amplitude self-cal corrections so far applied to the data, by this and previous invocations of `selfcal` and `gscale`, are summed over all integrations and telescopes to form a mean, and then all corrections are divided through by this number to normalize them. This provides an established but hard to justify means of anchoring the flux scale, in the event that feedback in the CLEAN+`selfcal` loop becomes slightly positive. It is also often abused as a means to get amplitude self-cal to work before the model has been completed at short UV spacings. If you are tempted to do this please look at the help given for the `selftaper` command.

`solint` - The solution interval in minutes.
Solutions are calculated within top-hat solution bins of this width and then are smoothed with a gaussian filter. The half power point of the filter in the Fourier plane is at $1/(2.\text{solint})$. According to the Shannon sampling theorem this is the highest frequency that can be sampled with a sample grid spacing of `solint`.

CONTEXT

Self-calibration is a means to finding self-consistent telescope based amplitude and phase corrections, through comparison of the observed data to an approximate model of the source structure. The algorithm implemented here performs least-squares minimization. It was based on sample code provided by Tim Cornwell.

As explained in the 'clean' help topic, self-calibration with 'selfcal' is usually combined in a loop with 'clean' and a number of auxiliary data editing, display etc.. commands.

The initial model to use with `selfcal` may either be provided by a few iterations of CLEAN, or by an external model read in with the `rmodel` command. Such an external model could come from model fitting, or be the model from a different map of the same source.

It is strictly recommended that one initially avoid amplitude self-calibration, and only use phase self-calibration until the model flux from CLEAN reaches that of the data, as displayed in `radplot`. Only at this point should amplitude self-calibration be used to remove amplitude errors. If you have trouble getting all the flux at short UV spacings, try switching to natural weighting (see help `uvweight`) and/or to using a UV-taper (see help `uvtaper`) or UV range, (see help `uvrange`) before continuing CLEANing. This may reveal the extended emission implied by the extra flux at short spacings. If this still doesn't reveal any more believable flux then, and only then attempt amplitude self-cal, but be sure to use the `selftaper` command

to weight down the significance of the badly fitting short UV spacings. If you fail to use selftaper in this way then selfcal will happily attempt to drag the short UV spacing visibilities down to fit the erroneously low model.

When interleaved applications of clean and selfcal start to provide diminishing returns, the residual map (as displayed by 'mapplot map') should appear to contain only noise and be devoid of beam-shaped features. Depending on the quality of the data, this may not always be achievable. If artifacts remain after many cycles of clean and self-cal, then this may indicate, either low-surface brightness extended emission that will take some time to clean out, or real problems with the data. Both cases can be determined by examining the fit between the model and observed visibilities with the 'vplot' command and the radplot command. Judicious editing may then be used to remove any bad data.

In addition, self-calibration will fail to find phase-solutions for any telescope that appears on less than one un-flagged closure triangle at a given integration and similarly fail to find gain solutions for telescopes with no un-flagged closure amplitudes. The selfflag command may be used to specify how to deal with this. The default is to flag all corrections that can not be deduced. These correction flags are propagated to the visibilities that they effect when the corrections are applied to the data.

To assign extra weight to a given telescope during self-calibration, or to prevent the complex gain of a given telescope from being changed see help on the selfant command.

Application of selfcal to multiple sub-arrays, IFs and polarizations

Selfcal is applied separately to each sub-array of each IF. Corrections accumulated in this fashion are propagated to all spectral-line channels and polarizations in the same IFs and sub-arrays. Thus by careful mapping and self-calibration of a strong spectral-line channel you will also be correcting the phases and amplitudes of the weaker channels so that they can sub-sequently be cleaned.

Agreement factors

Selfcal displays two agreement factors.

The one denoted 'rms' is the root-mean-square difference between the model and observed visibilities, measured in Jy. This is formed as the mean of the squares of the amplitude difference vectors between the model and observed visibilities.

The second, which is denoted by the name 'sigma', is the root mean

square of the squared difference between the data and the model (taking the real and imaginary parts of visibilities as separate measurements) divided by the individual variances implied by the visibility weights. This would be equivalent to the square-root of the reduced chi-squared, except that no account is taken of the number of degrees of freedom implied by the number of gains being varied.

RELATED COMMANDS

selfflag	- Used to control the fate of un-correctable data.
selftaper	- Used to down-weight short baselines during selfcal.
selflims	- Used to set limits to amp and phase corrections in selfcal.
selfant	- Set antenna based constraints in selfcal.
gscal	- Calculate overall telescope amplitude corrections.
startmod	- Phase selfcal to a starting model then discard the model.
corplot	- Display and allow editing of corrections.
uncalib	- Remove selected parts of corrections.
uvrange	- Set the min and max UV radii of usable visibilities.
fix_weights	- Set whether amplitude corrections affect the visibility weights.

selfflag *doflag, n_phase, n_amp*

Control the identification and fate of un-correctable visibilities.

EXAMPLES

1. To specify that a minimum of 3 telescopes for phase, and 4 telescopes for amplitude selfcal must exist in a closed loop of a given solution bin, or the solution bin will be flagged, type:

```
0>selfflag true,3,4
```

- In phase-only self-cal, good data on baselines that are not in closed arrays of at least 3 telescopes will be flagged.
- In amplitude self-cal, good data on baselines that are not in closed arrays of at least 4 telescopes will be flagged.

```
0>
```

2. To specify the same as two, but instead of flagging the bad solution bins, just don't correct them, type:

```
0>selfflag false,3,4
```

- In phase-only self-cal, good data on baselines that are not in closed arrays of at least 3 telescopes will not be used.
- In amplitude self-cal, good data on baselines that are not in closed arrays of at least 4 telescopes will not be used.

```
0>
```

3. To specify that all un-flagged data should be used in selfcal, type:

```
0>selfflag false,0,0
```

- In phase-only self-cal, all un-flagged data will be used
- In amplitude self-cal, all un-flagged data will be used

```
0>
```

All but the first line of each example are responses from 'selfflag'.

PARAMETERS

doflag - Flag un-correctable visibilities?

This controls what happens to visibilities that are found to be uncorrectable in subsequent invocations of selfcal. If this argument is false, the visibilities are left un-corrected. If true they are flagged.

n_phase - The minimum number of telescopes required for a phase solution. In theory the minimum number of connected telescopes required to form a constrained phase self-cal solution is 3. See below for further discussion.

`n_amp` - The minimum number of telescopes required for an amplitude solution. In theory the minimum number of connected telescopes required to form a constrained amplitude phase self-cal solution is 4. See below for further discussion.

If no arguments are provided, the current settings are displayed.

Whenever the observe command is used, all the above values are reset, to the values that would be set by typing:

```
selfflag true ,3 , 4
```

CONTEXT

This command determines the behavior of subsequent invocations of the selfcal and gscale self-calibration commands when they encounter un-constrained visibilities.

Self-calibration implicitly relies on the existence of closure phases for phase solutions, and closure amplitudes for amplitude solutions. To form a closure phase requires at least 3 connected telescopes in a closed loop, and to form a closure amplitude requires at least four.

Before attempting to ascertain corrections for a given integration, the current self-calibration algorithms attempt to determine if there are sufficient un-flagged visibilities to constrain solutions for each telescope. They do this in the following manner.

First they identify which telescopes lie on closed baseline paths. Only visibilities from these telescopes can be used in self-calibration. If a prior call to selfflag set dogain to true, then the rest of the un-flagged visibilities are flagged at this point, since they are totally un-correctable by self-calibration and will come to dominate the goodness of fit and prevent the fit from converging.

The number of correctable telescopes is then counted and compared against `n_phase` if performing phase-only selfcal, or against `n_amp` if amplitudes are being corrected. If there are insufficient telescopes to attempt a solution and 'doflag' is set to true, then the correction is flagged, and when the corrections are applied to the data, such correction flags are propagated to all affected the visibilities. These flags can be globally removed with the 'uncalib' command, or individually applied or removed with the 'corplot' command.

Since it is impossible to have fewer than 3 telescopes forming a closed array, if the `p_mintel` (`a_mintel` for phase selfcal) is set to less than or equal to 2 then the above checks will be turned off, no flagging will be done and all visibilities will be used in selfcal.

Note that if `n_phase` and/or `n_amp` are lowered below their defaults (3,4) then because there are too few constraints in integrations that benefit from this, the resulting visibilities will fit too well to the model. Needless to say, this is not recommended.

RELATED COMMANDS

- `selfcal` - Self-calibrate data with the current model.
- `selftaper` - Used to down-weight short baselines during `selfcal`.
- `selflims` - Used to set limits to amp and phase corrections in `selfcal`.
- `selfant` - Set antenna based constraints in `selfcal`.
- `gscale` - Finds overall amplitude corrections for telescopes.
- `startmod` - Phase `selfcal` to a starting model then discard the model.
- `corplot` - Display and allow editing of corrections.
- `uncalib` - Remove selected parts of corrections.

selflims *maxamp, maxphs, doclip*

Set limits on acceptable amplitude and phase corrections in selfcal.

EXAMPLES

1. To prevent selfcal from making amplitude corrections of more than 1.1 (unitless scale factor), or phase corrections of more than 10 degrees, type:

```
0>selflims 1.1,10
Selfcal amplitude corrections will be limited to 0.909091 -> 1.1.
Selfcal phase corrections will be limited to -10 -> 10 degrees.
0>
```

All but the first line are the response from selflims confirming the new limits.

2. To display the currently selected limits omit all arguments:

```
0>selflims
Selfcal amplitude corrections will be limited to 0.909091 -> 1.1.
Selfcal phase corrections will be limited to -10 -> 10 degrees.
0>
```

3. To cancel all limits type:

```
0>selflims 0,0
Selfcal amplitude corrections will not be limited.
Selfcal phase corrections will not be limited.
0>
```

PARAMETERS

maxamp - If maxamp < 1 then no limit will be placed on amplitude corrections except where the selfant command has been used to specify amplitude limits for individual telescopes.

If maxamp >=1 then for each selfcal solution-time interval, selfcal checks whether the amplitude corrections of any telescopes fall outside the range 1/maxamp to maxamp, where maxamp is either the telescope specific amplitude limit, if specified by selfant, or the global amplitude limit set by the selflims command. If the amplitude correction of one or more telescopes exceeds these limits, then the value of the doclip argument determines how selfcal applies the amplitude limit. See the documentation of the doclip argument below for details.

maxphs - If maxphs <= 0 then no limit will be placed on phase

corrections except where the selfant command has been used to specify phase limits for individual telescopes.

If maxpchs ≥ 0 then for each selfcal solution-time interval, selfcal checks whether the phase corrections of any telescopes fall outside the range -maxpchs to maxpchs, where maxpchs is either the telescope specific phase limit, if specified by selfant, or the global phase limit set by the selflims command. If the phase correction of one or more telescopes exceeds these limits, then the value of the doclip argument determines how selfcal applies the amplitude limit. See the documentation of the doclip argument below for details.

doclip - Default = false

By default, when selfcal finds a correction that exceeds ranges set by the maxamp and maxpchs arguments of this command (selflims) or the selfant command, then the corrections of all telescopes in the associated solution interval are discarded and the visibilities are left uncorrected. However if the doclip argument is specified as true, then telescope corrections are clipped where necessary, to prevent them from exceeding the specified limits during the minimization procedure.

When a new data-set is read in using the 'observe' command the limits are reset to provide no limits on corrections.

CONTEXT

In some data sets where the data is sparse, noisy or not well constrained for other reasons, selfcal may produce ridiculously large aberrant corrections. The selflims command is used before calling selfcal to set limits on the allowed corrections.

Note that rejection of the solution for a given solution bin does NOT result in the effected visibilities being flagged - regardless of the flagging disposition set by any previous use of the selfflag command.

RELATED COMMANDS

selfcal - Perform self-calibration.
 selfflag - Used to control the fate of un-correctable data.
 selftaper - Used to down-weight short baselines during selfcal.
 selfant - Set antenna based constraints in selfcal.

selftaper *gaussian_value, gaussian_radius*

Sets a (1 - gaussian) taper to weight down SHORT baselines.

EXAMPLE

1. To set a gaussian taper.

```
0>selftaper 0.1,10
Selfcal gaussian taper: value 0.1 at UV radius = 10 mega-wavelengths.
0>
```

The second line is the response from the selftaper command, confirming the taper selection.

2. To see the currently selected taper value omit all arguments:

```
0>selftaper
Selfcal gaussian taper: value 0.1 at UV radius = 10 mega-wavelengths.
0>
```

3. To cancel a previously selected gaussian taper, type:

```
0>selftaper 0
No selfcal UV-taper is currently set.
0>
```

PARAMETERS

gaussian_value - The value of the (1 - gaussian) taper at 'gaussian_radius'. Valid values lie between 0 -> 1 exclusive. Other values cause the cancellation of any existing gaussian taper.

gaussian_radius - The UV radius at which (1 - gaussian) taper should have RISEN to 'gaussian_value'. (values <= 0 de-select gaussian tapering). Unless changed with the mapunits command, the units are million-wavelengths.

To cancel a previous gaussian taper you should type: selftaper 0
Whenever a new UV data set is read with the 'observe' command any existing gaussian taper is cancelled.

CONTEXT

When required, this command should be called before using the selfcal command.

Repeated cycles of clean tend to build up a model that first becomes complete on long baselines. If amplitude selfcal is applied at this

point it will probably find solutions that pull down the observed amplitudes at the shorter UV radii to match the incomplete model there. To circumvent this, you should get selfcal to heavily weight down the short baselines wrt the longer baselines by calling selftaper.

A good indication of where the model and observed visibilities diverge is given by the plot of UV amplitude vs UV radius in radplot.

Selftaper forces subsequent invocations of selfcal to weight down the visibilities as a function of radius. The form of this function is.

$$(1 - \text{gaussian}(\text{uv_radius}))$$

In detail:

$$(1 - \exp(\log(1 - \text{gaussian_value}) * (\text{uv_radius} / \text{gaussian_radius})^2))$$

This goes from 0 at zero UV radius, reaches value=gaussian_value at a UV radius=gaussian_radius and approaches 1 as UV radius tends to infinity.

RELATED COMMANDS

```
selfcal    - Performs self-calibration.
selflims   - Used to set limits to amp and phase corrections in selfcal.
selfflag   - Used to control the fate of un-correctable data.
selfant     - Set antenna based constraints in selfcal.
gscal      - Calculate overall telescope amplitude corrections.
startmod    - Phase selfcal to a starting model then discard the model.
```

setcont *doset*

Add the current model to the continuum model, or vice versa.

EXAMPLE

0>setcont

Moved 20 components to the continuum model.

0>

0>setcont false

Moved 20 continuum components to the normal model.

0>

PARAMETERS

doset - Default = true
 If this argument is omitted or given as true, then the
 current models are added to the continuum
 model. Otherwise, if doset=false, then the continuum
 model will be moved back into the normal model.

CONTEXT

In observations of spectral-line sources, background continuum is often present. If the spectral-line features of the source are of primary interest then the constant contribution from the continuum part of the source should be removed before mapping individual spectral-line channels.

The usual way that this is done is to map channels that do not contain any spectral line features. This results in a clean-component model of the continuum. This model is then subtracted from the channels that do have spectral-line emission, leaving only the spectral-line features to be mapped.

While, in theory the subtraction of the continuum model could be done either in the map plane, or in the UV plane, in general it is more accurate to do it in the UV plane, and this is what difmap allows you to do.

Difmap differs from other packages in the way that it does this, in that it employs residual mapping techniques to make continuum subtraction more flexible. In difmap the continuum model is never subtracted directly from the observed visibilities. Instead difmap maintains the continuum model as a part of the established model, (which is subtracted on-the-fly during gridding). When the residual map is restored to display the clean map, the continuum portion of the established model is not added into the map along with the rest of the

model, thus leaving just the spectral-line portion of the map.

The advantages of this approach are:

1. The continuum model can be adjusted or deleted at any time without affecting the observed UV data.
2. Since the model visibilities and the observed visibilities are kept apart, it is still possible to use self-calibration after a continuum model has been installed.
3. It may be quicker given that the subtraction is only performed on channels that are actually mapped.

The disadvantages are:

1. It is not possible to take continuum-subtracted visibilities into another package, since the model is saved as a model file separate from the UV FITS file.

Another feature that may be seen as an advantage by some, and a disadvantage by others, is that `radplot` and other visibility plotting commands, display un-subtracted visibilities.

USAGE

The 'setcont' command adds the current tentative and established models to any existing continuum model.

For example, say one had a source with spectral-line emission only in channels 21-29 of a 40 channel observation:

1. To determine a continuum model one would use the `select` command to select channels 1..20 and channels 30..40, since these channels only contain continuum.
2. Make the best possible map from these channels, using `clean` or `modelfit`, combined with `selfcal` if the continuum is strong enough. All the `selfcal` corrections are applied across all channels, so in correcting the continuum channels you also correct the spectral-line channels.
3. Use the `setcont` command to add the model made in (2) to the continuum model.
4. To check the continuum subtraction, select a single one of the continuum only channels, and display the residual map with `mapplot`. If this reveals any structure then the continuum model is not correct, and further cleaning etc.. should be applied. Go back to step 1, and perhaps try selecting a better set of continuum

channels. If you do any further cleaning, then use the `setcont` command to update the continuum model with the new components.

5. If the continuum was too weak for self-calibration to be feasible, but one or more of the spectral-line features is strong enough, select the strong channel, make a map of it and perform self-cal as necessary. If significant corrections were made then, go back to step 1, to improve on the continuum model.
6. Map each channel containing spectral-line features. Use the

```
mapl cln
```

command to see the clean map, uncontaminated by the continuum.

OTHER COMMANDS THAT AFFECT THE CONTINUUM MODEL

`wmodel:`

With the addition of the continuum model, `difmap` now has three models that can be addressed by the user. For instance the `clrmod` command allows you to specifically select which of these get cleared.

After using the `select` command, you are advised to type:

```
0>clrmod true
```

This clears both the established and tentative models that were built up during the previous channel selection, but preserves the continuum model for the new channel selection.

If you want to clear the continuum model, the optional 3rd argument of `clrmod` must be given as `'true'`.

`wmodel` and `rmodel:`

The `wmodel` and `rmodel` commands default to writing/reading the non-continuum parts of the established and tentative models. The optional second argument is required to ask it to separately handle the continuum part. See `help wmodel` and `help rmodel`, for details.

`save` and `get:`

The `'save'` command saves continuum models separately from the normal models, in a file with suffix `".cmo"`. The `'get'` command looks for a file with this extension and if found, reads the file, placing the model directly in the continuum model.

shift *xshift, yshift*

Shift the observation phase-center, CLEAN window and model positions.

EXAMPLES

```
1. 0>shift 10,-1
    shift: Shifting UV data, clean models and windows by: 10 (mas) East
    shift: Shifting UV data, clean models and windows by: -1 (mas) North
    Total accumulated eastward shift = 10 (mas).
    Total accumulated northward shift = -1 (mas).
    0>
```

All but the first line are responses from the 'shift' command.

- Shifts are applied incrementally. To see what the accumulated shifts are, omit all arguments:

```
0>shift
Total accumulated eastward shift = 10 (mas).
Total accumulated northward shift = -1 (mas).
0>
```

- To remove the accumulated shift use the 'unshift' command.

```
0>unshift
unshift: Removing accumulated position shifts.
0>
```

PARAMETERS

xshift - The distance to shift the source Eastwards.
yshift - The distance to shift the source Northwards.

CONTEXT

This command may be called at any time. It is useful for checking whether there is flux beyond the bounds of the map. It is also useful if one has a single sided source with a bright core at the center of the map. In the latter case, half of the map area is wasted. In this case the shift command may be used to better utilize the space available and thus allow a smaller map grid. A more complex use is to map a large area containing several widely separated sources. In this case the shift command is used to center the map on one source at a time while it is cleaned. This has the advantage that a smaller map grid may then be utilised to realize faster mapping and less memory usage.

The 'shift' command modifies the observed and model visibilities simply by moving the phase center of the map. It also modifies the

latest and established model component lists and any clean windows to have positions wrt the new phase center.

SIDE EFFECTS

The modifications to the UV data invalidate the current residual map. This will be re-calculated on the next 'invert'. Thus shifting the data incurs the overhead of an extra 'invert'.

ASSOCIATED COMMANDS

unshift - Undo accumulated position shifts from the data, windows and models.

shiftto *ra, dec*

Move the phase-center, CLEAN window and model positions to a given RA,Dec.

EXAMPLES

1. The following example results in the apparent phase center of the observation, and the corresponding positions of the models and windows, being shifted to a Right Ascension of 12 hours, 58 minutes and 14 seconds, and a Declination of -6 degrees, 12 arc-minutes and 32 arc-seconds.

```
0>shiftto 12:58:14, -6:12:32
```

```
Shifting UV data, models and windows by: -30.5279 (arcmin) East
```

```
Shifting UV data, models and windows by: 25.1883 (arcmin) North
```

```
Total accumulated eastward shift = -30.5279 (arcmin).
```

```
Total accumulated northward shift = 25.1883 (arcmin).
```

```
0>
```

PARAMETERS

ra - The target Right Ascension expressed in hours, minutes and seconds of time.

dec - The target Declination expressed in degrees, minutes and seconds of arc.

CONTEXT

This command is useful when creating maps that are to be mosaiced in another package (eg. AIPS). In such cases it is usually necessary to produce a grid of maps which all have the same center coordinate. This command can thus be used to shift each map individually to the desired mosaic center.

The Right Ascension and Declination should be specified for the same epoch as the data were recorded for. This can be seen by running the 'header' command and looking at the number in parentheses which follows the RA listed under "Source parameters:".

SIDE EFFECTS

The modifications to the UV data invalidate the current residual map. This will be re-calculated on the next 'invert'. Thus shifting the data incurs the overhead of an extra 'invert'.

ASSOCIATED COMMANDS

shift - This allows a relative shift to be specified.

unshift - Undo accumulated position shifts from the data, windows and models.

showhist

Display the history that was recorded with the current observation.

This command displays a chronological list of entries made in the history file that was loaded with the current observation.

See 'help help' for directions on using the difmap pager.

showmodel *file_name*

Report detailed information about the current models.

EXAMPLE

The following example shows many features of a two component model that was fitted to visibility data using the `modelfit` command. There are a lot of parameters per model component, so this may look messy on a narrow terminal window.

```
0>showmod
```

```
Partitioning the model into established and variable parts.
```

```
Extracting 2 model components from the UV plane model.
```

```
The fixed established model contains 0 components (0 Jy).
```

```
The variable part of the model contains 2 components (5.84897 Jy).
```

```
# Source: 2230+114
```

```
# RA: 22 30 07.812 Dec: 22 30 07.812 Epoch: 1950.0
```

```
# Tentative model.
```

#	Flux (Jy)		East (arcsec)		North (arcsec)		Shape	R.A. (deg)	Dec (deg)
#	Value	Stdev	Value	Stdev	Value	Stdev			
#	-----		-----		-----		-----	-----	-----
	4.4795e+00	8.8e-03	-1.5506e-04	3.2e-06	4.4635e-04	1.3e-05	gauss	337.5325512451	11.4729
	1.3694e+00	9.8e-03	6.9116e-03	3.0e-05	-7.4671e-03	4.2e-05	gauss	337.5325532480	11.4729

```
0>
```

PARAMETERS

`file_name` - (Default=standard output)

The name of a file to write the model to. If this argument is omitted or specified as an empty string "", then the model will be displayed on the standard output and also in the log file.

CONTEXT

Whereas the `wmodel` command displays or records the current model in VLBI model format, this command shows them in a format designed to be human readable, given a wide enough terminal to see the long lines without line wrapping.

The `showmodel` command shows the following columns for each model component:

Flux (Jy)

This is the total flux density of the model component (ie. not its peak flux).

Flux stdev (Jy)

The standard deviation of the above flux if it was fitted by `modelfit`.

East (arcsec)

The eastward offset of the center of the component relative to the center of the image.

East stdev (arcsec)

The standard deviation of the above offset if it was fitted by modelfit. Zero if the eastern offset was not fitted.

North (arcsec)

The northward offset of the center of the component relative to the center of the image.

North stdev (arcsec)

The standard deviation of the above offset if it was fitted by modelfit. Zero if the northern offset was not fitted.

Shape (component type):

point - A delta function.
 gauss - A two dimensional gaussian.
 disk - Uniform disk.
 sphere - Optically thin sphere.
 ring - A ring.
 rect - A rectangle
 sz - Sunyaev Zeldovich

RA (degrees)

The Right Ascension of the component centroid (J2000)

Dec (degrees)

The Declination Ascension of the component centroid (J2000)

Major (arcsec)

For two dimensional components, this is the full-width at half maximum of the component along the major axis of the elliptically stretched component.

Major stdev (arcsec)

The standard deviation of the above major axis FWHM, if it was fitted by modelfit. Zero if the major axis FWHM was not fitted.

Minor (arcsec)

For two dimensional components, this is the full-width at half maximum of the component along the minor axis of the elliptically stretched component.

Minor stdev (arcsec)

The standard deviation of the above minor axis FWHM, if it was fitted by modelfit. Zero if the minor axis FWHM was not fitted.

Theta (degrees)

The rotation angle of the major axis of the elliptically stretched component, eastward of north (ie. counterclockwise in maps where east is towards the left).

Theta stdev (degrees)

The standard deviation of the above theta parameter, if it was fitted by modelfit. Zero if theta was not fitted.

Freq (Hz)

The reference frequency of the following spectral index.

Spectral Index

The spectral index of the component, where

```
Flux = freq ** spectral_index
```

Beware that the reported standard deviations of fitted parameters will only be good estimates if the weights of the visibilities are statistical weights. In other words, the weight assigned to each visibility should be $1/\text{var}$, where var is the expected variance of the real and imaginary parts of that visibility. A good indication of whether correct weights were recorded with the visibilities, is to look at the reduced chi-squared that modelfit reports. If the residual image after fitting only shows noise, then the chi-squared value should be around 1. If it is significantly greater than 1, then either the model didn't fit very well, or the weights are larger than they should be (ie. the estimates variances are too small). Alternatively, if chi-squared is significantly less than 1, then the weights must be too low (ie. the estimated variances are too large).

One way to correct the visibility weights, is to use the uvaver command to compute new weights based on the scatter of the visibilities (see the doscatter argument of uvaver). For this to work, however, the averaging time specified to uvaver must cover a few integrations of the original observation. Care must be taken not to choose an long averaging time that will cause time-smearing. To avoid doing this, you can use the check_uvaver to see whether the chosen averaging time will adversely affect features in the map.

RELATED COMMANDS

```
wmodel - Write model components to a VLBI model file.
rmodel - Read a new CLEAN model from a file.
save    - Save all data files and the difmap running parameters.
```

showpar

Display a list of difmap configuration parameters.

EXAMPLE

```
0>showpar
Difmap configuration state:
mapunits mas
select I, 1, 1
uvtaper 0, 0
uvrange 0, 0
uvweight 2, 0, false
uvzero 0
selftaper 0, 0
selflims 0, 0
xyrange 0, 0, 0, 0
integer niter; niter=100
float gain; gain=0.05
float cutoff; cutoff=0
float cmul; cmul=0
logical docont; docont=true
mapcolor grey, 1.000000, 0.500000
mapfunc linear, 0.000000, 0.000000
string vflags; vflags=""
string rflags; rflags=""
string pflags; pflags=""
selfflag true, 3, 4
float levs(1)
levs(1:1) = 0
0>
```

CONTEXT

Some aspects of Difmap behavior are controlled by parameters set through configuration variables and commands. The 'showpar' command lists the current state of these parameters. It does this in an indirect manner, by showing the command file that the 'save' command would record if invoked to record difmap's internal state.

This is not the most convenient way that one could conceive of, but unless I get a chance to re-write the difmap interface, I have little choice but to show the parameters in this fashion.

specbase *mode, baseline_group1, baseline_group2, ...*

Select baselines to be displayed by specplot.

EXAMPLE

1. To see what baselines are currently selected, omit all arguments.

```
0>specbase
Specplot will plot baselines of *
0>
```

2. To request that all baselines be plotted separately.

```
0>specbase split,*
Specplot will plot baselines of *
0>
```

3. To request that all baselines be combined to construct a single spectrum:

```
0>specbase group,*
Specplot will plot baseline groups:
*
0>
```

3. Specify multiple groups of baselines to plot, each consisting of all baselines of a different telescope from sub-array 2.

```
0>specbase group, 2:br, 2:fd, 2:hn, 2:kp
Specplot baseline selections:
2:BR
2:FD
2:HN
2:KP
0>
```

4. To request two spectra, one from all baselines of the BONN telescope except those of the OVRO telescope, and a second comprised from all baselines of the OVRO telescope except those of the BONN telescope:

```
0>specbase group, bonn ! ovro, ovro ! bonn
Specplot will plot baseline groups:
1:BONN ! 1:OVRO
1:OVRO ! 1:BONN
0>
```

5. To display one spectrum for each baseline of antenna AN2 from sub-array 2:

```

0>specbase split, 2:an2
Specplot will plot baselines of 2:AN2
0>

```

ARGUMENTS

- mode - This tells specplot how to interpret the baseline group arguments. It takes two values:
- split - Split the first group into its component baselines and plot one spectrum per baseline.
- group - Plot one spectrum per group of baselines.
- baseline-group,... - One or more groups of baseline specifications. Type 'help antenna_names' for legal baseline specifications, and see below to determine how to combine them.

To see the current selection, type the specbase command with no arguments.

CONTEXT

The specplot command plots scalar and vector averaged visibility spectra. Visibilities are selected by polarization, time range, UV radius range, and baseline selections. The specbase command is responsible for supplying baseline selections, either as groups of baselines or as single baselines taken from a single group.

Where more than one baseline selection is given, spectra can be plotted for each of them separately. The 'specorder' command can be used to direct when to change from one baseline selection to the next while plotting spectra.

BASELINE GROUPS

Each argument of the 'specbase' command is a group of baseline specifications, each specification describing one or more baselines to be included or excluded. Each specification is preceded by either '+' or '!'. Those preceded by '+' add to the group of baselines to be averaged, whereas those preceded by '!' subtract from the group of baselines established by earlier specifications. For example a group consisting of:

```
"ovro + bonn ! ovro-bonn"
```

means include all baselines of OVRO and all baselines of BONN except baseline OVRO-BONN. Note that the order in which specifications are listed is important. If the above were re-ordered to:

```
"ovro ! ovro-bonn + bonn"
```

The !ovro-bonn component would have no effect because the +bonn component would re-introduce the baseline that was just removed.

The first specification is implicitly preceded by '+' to make it additive, however if the first specification is explicitly preceded by '!' then this is interpreted as being equivalent to '* !'. For example:

```
"!2"
```

means included all baselines of all sub-arrays except those of sub-array 2.

Other examples are given in the EXAMPLE section at the beginning of this help file.

RELATED COMMANDS

```
-----
specplot   - Plot time-averaged visibility spectra.
specopt    - Preset specplot display options.
specsmooth - Preset the spectral resolution displayed by specplot.
specorder  - Set the order in which spectra are to be displayed.
specpol    - Select the list of polarizations to be displayed by specplot.
spectime   - Select the time ranges to be displayed by specplot.
specuvr    - Select the UV radius ranges to be displayed by specplot.
```

specopt *x_units, flags*

Preset specplot display options.

EXAMPLES

1. To see the currently selected options without changing them, omit all arguments.

```
0>specopt
Specplot x-axis=channels  flags="3jJe".
0>
```

2. To select an X-axis that is labelled in units of frequency instead of channels, and change the flags such that lines are drawn as simple vectors instead of bins.

```
0>specopt 5, frequency, 3j
Specplot nplot=5  x-axis=frequency  flags="3j".
0>
```

ARGUMENTS

- x_units** - Remembered. Initial default = channel
 The units plotted on the X-axis of each spectrum.
 Valid values are:
 channels - Spectral-line channels.
 frequency - Spectral-line channel center frequencies.
 Unambiguous abbreviations are recognized.
- flags** - Remembered. Initial default = "3jJ".
 Plot option flags.
 Many of the toggled options that are bound to keyboard
 keys for interactive use in specplot, can be preset by
 placing those same keys in a string.
 The initial default string of flags specifies that
 both amplitude and phase will be plotted (3), that
 neighboring points will be joined (j) to form a
 histogram style plot (J) and that error bars be plotted
 (e). See 'help specplot' for the meaning of other toggle
 options.

CONTEXT

The specplot command plots time-averaged visibility spectra. By default it plots one spectrum for each baseline in one's observation, at 3 plots per page, plotting each as amplitude and phase versus channel number. Neighboring points are joined by lines, using a style similar to a histogram to indicate how the data are binned into channels.

The x-axis of the plotted spectra is always plotted in order of increasing channel number, and IF number. However the values plotted along the axis can be changed to other forms with the 'x_unit' argument. This effects the X-axis scale, but not the ordering of points. Thus two IFs plotted side by side may show frequencies that increase in opposite directions.

There are a number of other binary options that effect the way that spectra are plotted but do not require that the spectra be reconstructed. Interactively these are toggled via keyboard keys and those options will be remembered between specplot sessions. These can be seen by typing specopt without any arguments, or they can be replaced by specifying a new string of option flags with the 'flags' argument. The individual flags are described in the specplot help pages.

Related commands

```
specplot    - Plot time-averaged visibility spectra.
specsmooth - Preset the spectral resolution displayed by specplot.
specorder   - Set the order in which spectra are to be displayed.
specbase    - Select the baselines to be displayed by specplot.
specpol     - Select the list of polarizations to be displayed by specplot.
spectime    - Select the time ranges to be displayed by specplot.
specuvr     - Select the UV radius ranges to be displayed by specplot.
```

specorder *selection-type1, ...*

Set the order in which specplot plots spectra.

EXAMPLES

1. To see the currently selected plot order without changing it, omit all arguments.

```
0>specorder
Specplot selection order: baseline
0>
```

2. To plot one spectrum for each scan that is distinguishable from its neighbor by a 1 hour time gap:

```
0>spectime 306/07:56:44, 398/08:16:19, -60
Specplot time range 306/07:56:44 - 398/08:10:52, scan separation=60 mins
0>specorder time
Specplot selection order: time
0>specplot
```

3. To change from plotting one spectrum per baseline selection, to plotting 1 page per baseline, partitioned into one spectrum of each of 4 stokes parameters.

```
0>specpol I,Q,U,V
Specplot polarization selections: I, Q, U, V
0>specopt 4
Specplot nplot=4 x-axis=channels flags="3jJ".
0>specorder polarization, baseline
Specplot selection order: polarization baseline
0>specplot
```

4. To see how the spectrum averaged over all baselines, changes with UV radius range.

```
0>specuvr 0,0,10
Specplot UV range: uvmin=0 uvmax=151.407 uvstep=10 (kilo-wavelengths)
0>specbase group,*
Specplot will plot baseline groups:
*
0>specorder uvrage
Specplot selection order: uvrage
0>specplot
```

ARGUMENTS

selection-type_1 - The type of the selection that will be incremented the fastest. The recognized selection types are:

- baseline - Baseline selections established with the 'specbase' command.
- polarization - Polarization selections established with the 'specpol' command.
- time - Time-range selections established with the 'spectime' command.
- uvrange - UV-radius range selections established with the 'specuvr' command.

...

selection-type_n - The type of the selection that will be incremented each time that selection types 1 -> n-1 have been exhausted.

CONTEXT

The specplot command plots scalar and vector averaged visibility spectra. The visibilities that are averaged to form each spectrum are selected by U-V radius range, polarization, time and baseline.

On a given page only one of these quantities is allowed to change between plotted spectra, but when all values of that quantity have been exhausted, one of the other quantities may be incremented before starting the next page. The order in which the different quantities are incremented is determined by the 'specorder' command.

By default, only the baseline selection specified by the 'specbase' command is allowed to increment. This behavior is the equivalent of what one would see after typing:

```
0>specorder baseline
```

Alternatively, if one wanted to look at the spectrum of a specific group of baselines at different times, one would call the specbase and spectime commands to select the desired baseline group and the time ranges and then type:

```
0>specorder time
```

If one wanted to extend this such that when one group of baselines had been displayed to each of the desired time ranges, a second group of baselines was plotted over the same time ranges, one would call the specbase command to select two baseline groups and type:

0>specorder time, baseline

RELATED COMMANDS

- specopt - Preset specplot display options.
- specsmooth - Preset the spectral resolution displayed by specplot.
- specbase - Select the baselines to be displayed by specplot.
- specpol - Select the list of polarizations to be displayed by specplot.
- spectime - Select the time ranges to be displayed by specplot.
- specuvr - Select the UV radius ranges to be displayed by specplot.

specplot *nplot, aver_mode, ca, cb, amin, amax, pmin, pmax, npage*

Plot time-averaged visibility spectra.

EXAMPLES

1. Plot vector averaged spectra, autoscaled in amplitude, phase and channels, using the current defaults for what gets plotted.

```
0>specplot
```

For help move the cursor into the plot window and press 'H'.

```
0>
```

2. As in example 1, but use scalar averaging to construct spectra instead of vector averages, and restrict the X-axis range to channels 10 to 20.

```
0>specplot 0, scalar, 10, 20
```

For help move the cursor into the plot window and press 'H'.

```
0>
```

2. Change all the default parameters that determine what goes into each spectrum and how to plot them, then plot the result.

```
0>specbase split, kp
```

Specplot will plot baselines of 1:KP

```
0>specpol I,Q,U,V
```

Specplot polarization selections: I, Q, U, V

```
0>spectime 355/14:43:29, 356/05, -20
```

Specplot time range 355/14:43:29 - 356/04:20:30, scan separation=20 mins

```
0>specsmooth channels, hanning, 3
```

Specplot smoothing: units=channels window=hanning fwhm=3

```
0>specopt frequency, "3j+"
```

Specplot x-axis=frequency flags="3+j".

```
0>specorder polarization, baseline
```

Specplot selection order: polarization baseline

```
0>specplot 4, vector, 10, 20, 0, 1.3, -180, 180
```

For help move the cursor into the plot window and press 'H'.

```
0>
```

ARGUMENTS

- nplot** - Default = 0 (this actually selects 3 plots per page).
 The number of spectra to plot on each page.

Note that 'nplot' spectra are simultaneously constructed in a single pass through the data, so it is faster to plot a few multi-plot pages than to plot a lot of pages of single plots.

- aver_mode - Default = vector
How to time-average visibilities. Valid modes are:
vector - Vector average visibilities.
scalar - Scalar average visibilities.
- ca,cb - Default = (All channels observed).
The numbers of the first and last channels to be plotted, or 0,0 to select all channels. Channels are numbered contiguously from the first channel of the first observed IF to the last channel of the last observed IF.
- amin,amax - Default = (Autoscale).
The plotted range of amplitudes, or 0,0 to select individually autoscaled amplitude sub-plots.
- pmin,pmax - Default = -180,180.
The plotted range of phases (degrees).
- npage - Default = 0
When plotting to a non-interactive device and npage is greater than 0, then a maximum of npage pages will be plotted.

CONTEXT

The specplot command plots time-averaged visibility spectra as amplitude and phase versus either spectral-line channel or frequency. The individual spectra of multiple IFs are plotted side by side on the page. Each page is also divided vertically to display multiple spectra, each differing from the last in either the polarization, time range, UV radius range, or the set of baselines averaged.

Note that the current stream selection is irrelevant to 'specplot', and the 'select' command need not have been called before invoking 'specplot'. This allows one to decide what channels and polarization to select in advance by looking at visibility spectra with 'specplot'.

The averaging mode, 'aver_mode'

Spectra are formed from a weighted average of selected visibilities. Specplot supports two averaging modes:

1. Vector averaging.

When visibilities are vector averaged the real and imaginary parts of each visibility are individually averaged and later converted to amplitude and phase for plotting.

2. Scalar averaging.

When visibilities are scalar averaged the real and imaginary parts of each visibility are first converted to amplitude and phase. The amplitudes and phases are then individually averaged.

The resulting amplitude spectrum is biased upwards by noise but is independent of phase calibration and position shifts.

Selecting the number of plots per page.

To form a spectrum the 'specplot' command has to read raw visibilities from disk and apply accumulated edits, self-calibration corrections, resoff corrections, and position shifts. For large data-sets this is time consuming, so rather than repeat this separately for each spectrum, the 'specplot' command constructs a whole page full of spectra from a single pass through the data. This takes more memory but is potentially much faster when more than one plot is requested per page. The number of plots per page can be specified on the command line and/or interactively.

The order in which spectra are plotted.

The visibilities that are averaged to form each spectrum are selected by U-V radius range, polarization, time and baseline.

On a given page only one of these quantities is allowed to change between plotted spectra, but when all values of that quantity have been exhausted, one of the other quantities may be incremented before starting the next page. The order in which the different quantities are incremented is determined by the 'specorder' command. By default, only the baseline selection specified by the 'specbase' command is allowed to increment. This behavior is the equivalent of what one would see after typing:

```
0>specorder baseline
```

If one wanted the polarization selection to be incremented each time that spectra for all baselines had been plotted, one would type:

```
0>specorder baseline, polarization
```

The increment order can also be changed interactively with the 'o' key (see below).

See 'help specorder' for the names of each of the selection types.

X-axis options

The type of X-axis that is plotted can be changed with the 'specopt' command. By default, amplitude and phase are plotted versus channel number.

Spectral resolution

The resolution of the plotted spectra can be reduced by smoothing them. The specsmooth command specifies the smoothing parameters to be used. The smoothing width may be specified in terms of channels or frequency.

Error bars

By default error bars are plotted for each spectral-line channel. They are toggled on or off with the 'e' key. They are formed from the individual visibility weights that go into the weighted average for each channel, which are interpreted as being the reciprocal of the square of the amplitude uncertainty. If this is not true, see the 'wtyscale' command. The error bars also take into account any smoothing that has been applied during plotting. This is described in the help pages of the specsmooth command.

Plot styles

By default, neighboring channels are plotted joined by lines in a staircase or histogram-like manner. This shows the real channel widths and sampling. Alternatively, lines can be drawn between the centers of neighboring channels. The latter looks aesthetically better when multi-channel smoothing has been selected. A third alternative is to simply plot each spectral-line channel as an isolated point. Points are then plotted at the center of each channel.

Interactive options

At any time while running specplot, a list of the key bindings that specplot understands will be displayed if you hit the 'H' key on your keyboard. The current key bindings include:

- H - List all key bindings.
- X - Exit specplot (right-mouse-button).
- L - Redisplay the current plot.
- N - Display the next page of spectra.
- P - Display the preceding page of spectra.

- s - Change the number of spectra plotted per page.
Beware, this requires a new pass through the data and may take some time with a large data-set.
- S - Change the plotted spectral resolution by setting or de-selecting smoothing parameters.
- 1 - Plot only amplitudes.
- 2 - Plot only phases.
- 3 - Plot both amplitudes and phases.
- E - Toggle whether error bars are plotted.
- V - Toggle between vector and scalar averaged spectra.
Beware, this requires a new pass through the data and may take some time with a large data-set.
- t - Change the baselines, polarization or times used to construct spectra.
You will be asked to specify a selection type name and zero or more specification arguments. The specification arguments have the same format as the equivalent command-line commands:


```

baseline, group|split, baseline_group1, ... (see help specbase).
polarization, polarization_name1, ...      (see help specpol).
time, start_time, end_time, scan_time      (see help spectime).
uvrange, uvmin, uvmax, uvstep              (see help specuvr).
```
- o - Change the order in which selections are incremented.
You will be asked for one or more selection names, in the same format as documented for the 'specorder' command.
See 'help specorder' for details.
- u - Delimit a X-axis range with the cursor (hit u twice for full range).
Either press this key (u) a second time to undo a previous range selection, or delimit the desired X-axis range by pressing the 'A' key (left mouse button), once at each end of its extent.
The range selection can be cancelled while in this mode by pressing the 'D' key (middle mouse button).
- U - Change the X-axis type.
You will be prompted for the name of a recognized axis type.
See help specopt for a list of valid type names.
- Z - Select a new amplitude or phase range (hit Z twice for full range).
Either press this key (Z) a second time to undo a previous range selection, or delimit the desired amplitude or phase range by pressing the 'A' key (left mouse button), once at each end of its

extent.

The range selection can be cancelled while in this mode by pressing the 'D' key (middle mouse button).

j - Toggle whether to join points with lines.

This toggles whether channel values are plotted as isolated points, or are whether they are joined by lines. If joined points are selected, there format can be further selected with the 'J' key described below.

J - Toggle between two possible line styles.

By default the lines are drawn as bins, in a staircase like pattern. The alternative is to simply draw straight lines directly between the centers of each channel.

+ - Toggle whether to use a crosshair cursor if available.

If the plot device supports a cross-hair cursor this will be substituted for the normal cursor if toggled on with the '+' key.

Related commands

specopt - Preset specplot display options.
 specsmooth - Preset the spectral resolution displayed by specplot.
 specorder - Set the order in which spectra are to be displayed.
 specbase - Select the baselines to be displayed by specplot.
 specpol - Select the list of polarizations to be displayed by specplot.
 spectime - Select the time ranges to be displayed by specplot.
 specuvr - Select the UV radius ranges to be displayed by specplot.

specpol *polarization1, polarization2, ...*

Select the list of polarizations to be displayed by specplot.

EXAMPLE

1. Select a single polarization.

```
0>specpol I
Specplot polarization selections: I
0>
```

2. Select a list of polarizations.

```
0>specpol I,Q,U,V
Specplot polarization selections: I, Q, U, V
0>
```

3. To see what polarizations are currently selected, omit all arguments.

```
0>specpol
Specplot polarization selections: I, Q, U, V
0>
```

ARGUMENTS

polarization1, ... - A list of one or more names of polarizations.
Type 'help polarization' for the list of
recognized names.

To see the current list omit all arguments.

CONTEXT

The specplot command plots scalar and vector averaged visibility spectra. Visibilities are selected by polarization, time range, UV radius range, and baseline groups. The specpol command is responsible for supplying a list of polarizations.

Where more than one polarization is given, spectra can be plotted for each of them separately. The 'specorder' command can be used to direct when to change from one polarization to the next while plotting spectra.

Recognized polarization names are listed in the on-line help for polarization. To see the list, type:

```
help polarization
```

RELATED COMMANDS

- specplot - Plot time-averaged visibility spectra.
- specopt - Preset specplot display options.
- specsmooth - Preset the spectral resolution displayed by specplot.
- specorder - Set the order in which spectra are to be displayed.
- specbase - Select baselines to be displayed by specplot.
- spectime - Select time ranges to be displayed by specplot.
- specuvr - Select the UV radius ranges to be displayed by specplot.

specsmooth *units, window, fwhm*

Preset the spectral resolution displayed by specplot.

EXAMPLES

1. To examine the current settings, invoke the specsmooth command with no arguments:

```
0>specsmooth
Specplot smoothing:  units=channels  window=none  fwhm=0
0>
```

2. To have specplot plot smooth spectra versus frequency (GHz) using a Hanning window of width 10 KHz (1e-5 GHz):

```
0>specsmooth frequency, hanning, 1e-5
Specplot smoothing:  units=channels  window=hanning  fwhm=1e-05
0>
```

ARGUMENTS

- | | |
|--------|---|
| units | - Default=channels
The units of the smoothing width, chosen from:
channels - Spectral-line channels.
frequency - Spectral-line channel frequencies.
Unambiguous abbreviations are recognized. |
| window | - Default=none
The window function used to smooth spectra to a given spectral resolution, from:
none - No smoothing.
hanning - Hanning smoothing window.
gaussian - Gaussian smoothing window.
boxcar - Boxcar smoothing window.
sinc - Sinc smoothing window
Unambiguous abbreviations are recognized.
Details of the smoothing functions are described below. |
| fwhm | - Default=0.0 (no smoothing)
The full width at half maximum of the smoothing function, in terms of the 'units' argument described above. |

CONTEXT

The specplot command plots time-averaged visibility spectra. By default it plots amplitude and phase versus channel number. The specsmooth command allows one to smooth spectra to reduce the apparent

spectral resolution. This potentially reduces noise, so the optionally plotted error bars are modified to reflect this.

While it is often most convenient to specify the width of one's smoothing in terms of spectral-line channels, different IFs can have different frequency increments per channel, so `specsmooth` allows one to specify the units in which to interpret the width. This choice is independent of the choice of the displayed X-axis units. The latter can be changed with the `'specopt'` command.

As with the other `specplot` setup commands, `'specsmooth'` arguments can be changed interactively while plotting, via the capitalized `'S'` key. This option prompts for the same arguments as taken by `'specsmooth'`.

Smoothing is performed on-the-fly every time that as each spectrum is displayed. The unsmoothed spectra are retained unchanged, so that future changes to the smoothing are independent of previous smoothing choices.

Smoothing functions

Smoothing is accomplished by convolving the data with the chosen smoothing function. Smoothing is performed on the real and imaginary parts of vector averaged spectra, and to the amplitudes and phases of scalar averaged spectra.

In order to describe the details of each smoothing function, let the value plotted along the X-axis be denoted by x , and let:

$$v = x / (0.5 * \text{fwhm}).$$

With this tranformation each smoothing function, $s(v)$ has

$$s(1) = 0.5 * s(0)$$

The area-normalized smoothing functions are then given by

$$\text{none} \quad s(v) = 1.0 \text{ (No smoothing, so fwhm is ignored)}$$

$$\text{hanning} \quad s(v) = \frac{\sin(\pi \cdot v)}{2 \cdot \pi \cdot v \cdot (1 - v^2)}$$

$$\text{gaussian} \quad s(v) = \sqrt{\ln(2)/\pi} * \exp[-\ln(2) \cdot v^2]$$

$$\text{boxcar} \quad s(v) = 0.5 \text{ for } |v| \leq 1, \text{ and } 0 \text{ elsewhere}$$

$$\text{sinc} \quad s(v) = \frac{c}{\pi} * \frac{\sin(c \cdot v)}{(c \cdot v)}, \text{ with } c=1.8954942670340$$

If one denotes convolution by the operator (*), the smoothing function by s , the data by f , and the data weights by, w , then the final smoothed spectrum and associated weights are given by:

$$\langle f \rangle = \frac{f \cdot w (*) s}{w (*) s}$$

$$\langle w \rangle = \frac{[w (*) s]^2}{w (*) s^2}$$

Note that data weights are interpreted as being related to error bars (e) by:

$$w = 1/e^2$$

Related commands

```
specplot   - Plot time-averaged visibility spectra.
specopt    - Preset specplot display options.
specsmooth - Preset the spectral resolution displayed by specplot.
specbase   - Select the baselines to be displayed by specplot.
specpol    - Select the list of polarizations to be displayed by specplot.
spectime   - Select the time ranges to be displayed by specplot.
specuvr    - Select the UV radius ranges to be displayed by specplot.
```

spectime *start_time, end_time, scan_time*

Select the range(s) of times to be averaged by the specplot command.

EXAMPLES

1. To see the current time ranges omit all arguments.

```
0>spectime
Specplot time range 306/07:56:44 - 398/08:16:19, scan length=132500 mins
0>
```

2. Select a single time range between 7am and 9am on day 306 of the year of observation.

```
0>spectime 306/07, 306/09
Specplot time range 306/07:56:44 - 306/08:59:46, scan length=63.0333 mins
0>
```

Note that the time range actually selected is determined by what times are actually sampled between the requested times.

3. Split a single time range into a number of 10 minute scans.

```
0>spectime 306/07, 306/09, 10
Specplot time range 306/07:56:44 - 306/08:59:46, scan length=10 mins
0>
```

3. Split a single time range into its component integrations.

```
0>spectime 306/07, 306/09, 0
Specplot time range 306/07:56:44 - 306/08:59:46, scan length=0 mins
0>
```

4. Split the time range at time gaps of at least 1 hour (60 minutes).

```
0>spectime 306/07, 306/09, -60
Specplot time range 306/07:56:44 - 306/08:59:46, scan separation=60 mins
0>
```

ARGUMENTS

- | | |
|-------|--|
| stime | - Default = The time of the first observed integration.
The start time of the overall time range as a string of form: DDD/HH:MM:SS.sss, where all but the initial DDD can be omitted (they each default to zero). |
| etime | - Default = The time of the last observed integration.
The end time of the overall time range, in the same form |

as 'stime'.

scan_time - Default = etime - stime
 A scan length or interval (minutes).
 If scan_time >= 0 then the time range stime -> etime will be split into sampled periods of up to 'scan_time' minutes.
 If scan_time < 0 then the time range stime -> etime will be split wherever there is at least '|scan_time|' minutes separating neighbouring integrations.

CONTEXT

The specplot command plots scalar and vector averaged visibility spectra. Visibilities are selected by polarization, time range, UV radius range, and baseline groups. The spectime command is responsible for supplying time ranges.

Where more than one time range is given, spectra can be plotted for each of them separately. The 'specorder' command can be used to direct when to change from one time range to the next while plotting spectra.

The arguments of the 'spectime' specify both the overall duration over which to plot spectra and how to apportioned this duration between spectra. The spectime command supports two ways to apportion times.

1. To separate the specified duration into multiple scans, each of a given length, specify that length, using the 'scan_time' argument, as a positive number of minutes.
 Each scan starts with the first sampled integration that follows the end of the preceding scan. Individual scans will be shorter than the given length where insufficient integrations are available.
 Note that if the specified length is zero, then each scan will encompass a single integration.
2. To separate the specified duration into scans that are separated by at least a given interval of time, specify the interval, using the 'scan_time' argument, as a negative number of minutes.

RELATED COMMANDS

specplot - Plot time-averaged visibility spectra.
 specopt - Preset specplot display options.
 specsmooth - Preset the spectral resolution displayed by specplot.
 specorder - Set the order in which spectra are to be displayed.
 specpol - Select the list of polarizations to be displayed by specplot.
 specbase - Select baselines to be displayed by specplot.
 specuvr - Select the UV radius ranges to be displayed by specplot.

specuvr *uvmin, uvmax, uvstep*

Select the UV radius ranges to be displayed by specplot.

EXAMPLES

1. To see what UV ranges are currently selected, omit all arguments.

```
0>specuvr
Specplot UV range: uvmin=0  uvmax=151.407  uvstep=10 (kilo-wavelengths)
0>
```

2. To restrict the UV range to 10 -> 30 kilo-wavelengths.

```
0>specuvr 10,30
Specplot UV range: uvmin=10  uvmax=30  uvstep=20 (kilo-wavelengths)
0>
```

3. To see how the spectrum averaged over all baselines, changes with UV radius range.

```
0>specuvr 0, 0, 10
Specplot UV range: uvmin=0  uvmax=151.407  uvstep=10 (kilo-wavelengths)
0>specorder uvrage
Specplot selection order: uvrage
0>specbase group, *
Specplot will plot baseline groups:
*
0>specplot
```

ARGUMENTS

- | | |
|--------|---|
| uvmin | - Initial default=0.0
The minimum UV radius to take visibilities from, measured in the wavelength units last established with the mapunits command. By default this is mega-wavelengths. |
| uvmax | - Default=Max UV radius in data.
The maximum UV radius to take visibilities from, or 0.0 to select the maximum UV radius available in the data. |
| uvstep | - Default = (uvmax - uvmin)
The UV radius increment between spectra, measured in the same units as uvmin and uvmax. |

To see what the current values of these arguments are, invoke the specuvr command without any arguments.

CONTEXT

The `specplot` command plots scalar and vector averaged visibility spectra. Visibilities are selected by polarization, time range, UV radius range, and baseline selections. The `specuvr` command is responsible for supplying UV radius ranges.

Where more than one UV radius range is given (ie `uvstep < uvmax-uvmin`), spectra can be plotted for each range separately. The '`specorder`' command can be used to direct when to change from one UV radius range to the next while plotting spectra.

RELATED COMMANDS

```
-----
specplot   - Plot time-averaged visibility spectra.
specopt    - Preset specplot display options.
specsmooth - Preset the spectral resolution displayed by specplot.
specorder  - Set the order in which spectra are to be displayed.
specbase   - Select baselines to be displayed by specplot.
specpol    - Select the list of polarizations to be displayed by specplot.
spectime   - Select time ranges to be displayed by specplot.
```

startmod *starting_model_file_name, solint*

Read a starting model from disk and phase self-calibrate against it.

EXAMPLE

O>startmod

Applying default point source starting model

Performing phase self-cal

Adding 1 model components to the UV plane model.

The established model now contains 1 components and 1 Jy

Correcting IF 1.

A total of 21 un-correctable visibilities were flagged in sub-array 1.

Fit before self-cal, rms=3.283113Jy sigma=10.445014

Fit after self-cal, rms=3.282763Jy sigma=10.437578

Reinstating the default IF.

clrmod: All model representations cleared.

Redundant starting model cleared

O>

All but the first line are responses from the startmod command and its clients.

PARAMETERS

model_file - Default = ""

The file name containing delta and/or gaussian model components. If no file name is given, a point source model will be substituted.

This model will be the starting model used to correct the phases via self-cal before mapping begins.

The model will be discarded after use.

solint - Default = 0.0.

The solution interval (minutes). See 'help selfcal' for details.

CONTEXT

Invoking startmod is equivalent to the following sequence of commands:

clrmod true

rmodel "somefile.mod"

selfcal

clrmod true

This results in the UV data being phase self-cal'd against a starting model (say from modelfit), without producing residuals between the model and the observed data. Since the model does not become part of the established model, the starting model need only be a coarse estimate of the true flux distribution.

SIDE EFFECTS

Any existing models will be cleared before reading in the starting model and the UV data will be re-inverted by the next operation that requires the residual map.

telname *itel*, *isub*

Returns the name of telescope '*itel*' of sub-array '*isub*'.

EXAMPLE

example 1 - Print out all (*ntel*) telescope names in sub-array 1.

```
0>do i=1,ntel
1> print telname(i)
1>end do
1:BONN
1:WSRT
1:JBNK1
1:BOLOGNA
1:VLBA_KP
1:VLBA_PT
1:VLBA_LA
1:VLA27
1:OVRO
0>
```

example 2 - Display the self-cal corrections for each telescope in turn from sub-array 2.

```
0>
0>do i=1,ntel
1> corplot 'telname(i,2)'
1>end do
```

Example 3 - Print the name of the third telescope in sub-array 2.

```
0>print telname(3,2)
2:OVRO
```

The "number:" prefix indicates the number of the sub-array to which the telescope belongs.

PARAMETER

itel - The sequential number of the telescope whose name is required.

isub - Default: 1
The number of the sub-array from which the telescope name is required.

RELATED FUNCTIONS

ntel - Return the number of telescopes in a given sub-array.

nsub - Return the number of sub-arrays in an observation.

tplot *subarray, IF_index*

Plot time sampling for each telescope of an observation.

EXAMPLES

1. Displaying telescope sampling on an interactive device.

```
0>tplot
Graphics device/type (? to see list, default /NULL): /xw
Move the cursor into the plot window and press 'H' for help
```

All but the first line of the above is output from tplot.

2. Displaying telescope sampling of all sub-arrays, on a hard-copy device.

```
0>dev /ps
0>tplot
Page 01: Subarray 1
Page 02: Subarray 2
0>
```

One page was plotted per sub-array to a postscript file.

3. Displaying telescope sampling of a single selected sub-array, on a hard-copy device.

```
0>dev /ps
0>tplot 3
Page 01: Subarray 3
0>
```

A single page was plotted to a postscript file, showing telescope sampling in sub-array 3.

4. Starting the plot from sub-array 1 of IF 2.

```
0>tplot 1, 2
Move the cursor into the plot window and press 'H' for help
```

ARGUMENTS

- | | | |
|----------|---|---|
| subarray | - | Default="" which selects all sub-arrays, starting from 1. To start plotting at a specific sub-array, provide its number here. The first sub-array is number 1. If the output device is a non-interactive (eg. hard-copy) device then only this sub-array will be plotted. |
| IF_index | - | Default=The index of the first sampled IF. |

The index of the IF to plot first.

CONTEXT

tplot plots the sampling of all telescopes, using one page per sub-array and IF. Each telescope is assigned a line parallel to the time axis of the plot. At the time-stamp of each integration, each telescope whose visibilities have not been completely deleted is shown as a dot. In general, each telescope is sampled on a number of baselines per integration. If the visibilities to all those baselines are not flagged, then the point is plotted in green. If some are flagged but at least one is not flagged then orange is used, and if all visibilities are flagged then the point is plotted either in red or blue depending on whether the flagging is dominated by visibility flags or by selfcal correction flags.

Note that what the visibilities that go into determining the flag status of the plotted points, are those selected with the 'select' command, rather than all visibilities in the observation.

INTERACTIVE MODE

If the device that you are plotting on possesses a cursor, then tplot goes into an interactive mode. In this mode, move the cursor into the plot area and press one of the following keys:

- H - (Help) Briefly list all keyboard options.
- X - Quit the tplot session, and return to the difmap prompt.
- L - Re-display the plot.
- . - Toggle the marker symbol used to display the data between the smallest available and the next largest. By default the smallest possible marker symbol is used for speed, but if the data are sparse then the larger marker symbol may be more convenient.
- N - Display the sampling of the next sub-array.
- P - Display the sampling of the previous sub-array.
-] - Display sampling of the next IF.
- [- Display sampling of the previous IF.
- T - Prompt for the next sub-array to be plotted. This is then entered by its numeric ID, at the keyboard.
- U - Select the time range to be plotted. Initially tplot plots

sampling over the full range of integration times. To restrict the range press U, move the cursor to the start of the new range and press A (left mouse button), then move the cursor to the end of the required range and press A again. Sampling of the current sub-array will be re-plotted within the new range. To return to the full time range press U twice in succession.

- B - Break the displayed time range into scans. A scan is defined as a block of data, separated from the next block of data by more than an hour (this can be changed with the 'scangap' command). If there are no gaps this large then no change in plotting will result. Otherwise, each scan will be plotted within its own box and the gaps between scans will no longer be plotted. Scan mode will remain in effect until B is pressed again.
- C - Flag any un-flagged samples within a given area (see below).
- R - Restore any flagged samples within a given area (see below).
- W - Toggle the scope of edits with respect to spectral-line channels. Each plotted point comes from a single IF, and its displayed flag status is determined from the combined status of all the channels currently selected from that IF on each baseline of the telescope being sampled. By default, if you select such a point to be flagged, then all spectral-line channels of the source IF will be flagged for each of the effected baselines, irrespective of the channels which actually went into making the plot. The alternative is to edit just the channels that went into making the plot. The 'W' key toggles which of the two options is used, and a mode line above the plot describes which option is currently in effect.
- + - This key toggles whether the cursor is shown as a small cross or as a cross-hair that extends the full width and height of the display. Currently only the /xserve and /xwindow PGPLOT devices support this feature.
- @ - This key toggles between plotting universal time and Greenwich mean sidereal time along the X-axis of the plot.

EDITING

The Clip and Restore keys, 'C' and 'R' enable you to edit all points inside a rectangular box selected with the cursor. With the 'C' option, all visibilities related to the selected stations over the selected integrations are flagged. Similarly, the 'R' option unflags all related visibilities in the area. To use these facilities, press the respective key and then move the cursor to one corner of the required region and press 'A' (left mouse button). Then move the cursor to the opposite corner of the required region and again press

'A'. The display will be updated to show the results.

Examples of how one might use this facility include, removing a whole scan at a time for all telescopes, or flagging a bad station over the whole time range.

Also see the description of the 'W' toggle, which determines whether to restrict the edits to the channels that went into making the plot, or whether to edit all channels of the displayed IF.

CONFIGURATION VARIABLES

The display-mode option keys listed above may be preset in the variable, 'tflags'. If tflags is an empty string, "", then the default option string is used. This is currently also an empty string. To override this default and have radplot start up by displaying with large dot sizes, on the command line one would type:

```
O> tflags="."
```

If you always want to override the default, then place such an assignment line in your difmap login file (the file named in the optional DIFMAP_LOGIN environment variable).

RELATED COMMANDS

```
scangap -   Change the time gap used to delimit neighboring scans.
radplot  -   Display visibility amplitude/phase vs. UV radius.
projplot -   Display visibility amplitude/phase vs. projected UV distance.
uvplot   -   Display the sampling of the UV plane.
vplot    -   Display observed and model visibilities from the current IF.
cpplot   -   Display observed and model closure phases interactively.
corplot  -   Display the self-cal corrections of the current IF.
```

uncalib *dophase, doamp, doflag*

Undo recorded telescope calibrations (eg. self-cal corrections).

EXAMPLE

1. To remove all phase corrections amplitude corrections and correction flags, type:

```
0>uncal true,true,true
uncal: All telescope phase corrections have been un-done.
uncal: All telescope amplitude corrections have been un-done.
uncal: All telescope correction flags have been un-done.
0>
```

2. To remove just amplitude corrections, while leaving selfcal correction flags and phase corrections in place, type:

```
0>uncal false,true
uncal: All telescope amplitude corrections have been un-done.
0>
```

All but the first line are messages from 'uncalib'.

PARAMETERS

dophase - If true then undo telescope phase calibrations (Default=false).

doamp - If true then undo telescope amplitude calibrations (Default=false).

doflag - If true, then remove all telescope correction flags (Default=false).

CONTEXT

Contained in every data-set is the list of integration based telescope amplitude and phase corrections per sub-array and IF, that have been applied to the data. These corrections are incrementally updated whenever further telescope based calibrations are applied, such as when 'selfcal' is applied. By running the 'uncalib' command it is possible to undo the effects of either or both of the amplitude and phase calibrations. It is thus easy to go back to the un-calibrated data-set or undo the effects of a badly constrained self-calibration. Note that this removes the effects of ALL previous self-calibrations etc.. It is also possible to remove all visibility flags applied by selfcal without effecting normal visibility flags.

SIDE EFFECTS

The current residual map is invalidated by the data modification, so the next map oriented task invokes 'invert'.

RELATED COMMANDS

selfcal - Self-calibrate data with the current model.
corplot - Display and allow editing of corrections.

unflag *baselines, allchan, start_time, end_time*

Unflag a set of baselines over a selectable time range.

EXAMPLES

The following example unflags all visibilities of the currently selected channels.

```
0>unflag *
```

Unflagging all baselines in the currently selected channels.

The next example unflags all visibilities of sub-array 2 of the currently selected channels.

```
0>unflag 2:
```

Unflagging baselines of subarray 2 in the currently selected channels.

The next example unflags just the visibilities of baselines that are associated with an antenna called rx3, (in the default subarray, which is 1). If rx3 were in sub-array 2, you would type 2:rx3.

```
0>unflag rx3
```

Unflagging baselines of antenna 1:RX3 in the currently selected channels.

The next example unflags just a single baseline, but this time for all channels, irrespective of the current selection.

```
0>unflag rx3-rx5, true
```

Unflagging baseline 1:RX3-RX5 in all channels.

The final example unflags all baselines in just the selected channels, between 10:45 and 10:50 on the 20th of November 2001.

```
0>unflag *, false, 20-nov-2001:10:45, 20-nov-2001:10:50
```

Unflagging all baselines in the currently selected channels.

PARAMETERS

baselines - The set of baselines to unflag, specified by one of the following forms:

- * - All baselines in all sub-arrays.
- sub: - All baselines in the sub-array specified by the number 'sub'.
- sub:ta - All baselines of the telescope named 'ta', in sub-array 'sub'.

sub:ta-tb - The single baseline of the
telescopes named 'ta' and 'tb'.

In the above, if sub: is omitted from the last
two specifications, 1: is quietly substituted.

allchan - Default = false
If true, unflag visibilities of the specified
baselines over all channels and IFs. Otherwise
only unflag visibilities in the channels that
were last selected with the 'select' command.

start_time - Default = ""
If not omitted, or specified as an empty string,
visibility editing will start from the start of
the observation. Otherwise this argument should
specify the date and time at which editing
should start, specified in the following format.

dd-mmm-yyyy-mm:hh:mm:ss

Where dd is the day of the month 1-31, mmm is
a standard 3-letter abbreviation of a month name,
and yyyy is a fully qualified year (ie. 2003, not
03). Trailing components of the time that follows
the date can be omitted. The hour part, hh, is
an hour number 0-23. The minute part, mm and the
seconds part, ss go from 0-59.

end_time - Default = ""
If omitted, or specified as an empty string,
visibilities from the start time to the end of
the observation will be edited. Otherwise this
argument should specify the date and time at
which editing should end, using the same format
as described for the start_time argument.

CONTEXT

In most cases it is best to edit visibilities visually, using a
visibility inspection command like vplot, tplot or radplot. However
sometimes it is desirable to be able to perform editing
from a script, without any user interaction. The 'unflag' command
facilitates this.

RELATED COMMANDS

flag - Flag a set of baselines over a selectable time range.

unshift

Undo accumulated position shifts from the data, windows and models.

EXAMPLE

0>unshift

unshift: Removing accumulated position shifts.

0>

CONTEXT

The 'shift' command can be used to incrementally shift the position of the phase center of the data, and the associated positions of models and CLEAN windows. The 'unshift' command puts the data back to the position that they had when the observation was read into difmap.

ASSOCIATED COMMANDS

shift - Shift the observation phase-center, CLEAN window and model positions.

`uv_to_wav` *distance*

Convert a UV plane distance from UV units to wavelengths.

EXAMPLE

The following example converts 10 kilowavelengths to wavelengths.

```

0>mapunits arcmin
Distances in the map plane now have units: arcmin.
Distances in the UV plane now have units: kilo-wavelengths.
0>print uv_to_wav(10)
10000
0>

```

PARAMETERS

```

distance    - A distance in the UV plane, expressed in the currently
               selected UV units.

```

CONTEXT

Use of the 'mapunits' command to specify which units should be used for specifying and displaying distances in the image plane, also changes the units that are used for the display and entry of UV coordinates.

The 'uv_to_wav()' function allows the user to convert a distance in these units, into wavelengths.

RELATED COMMANDS

```

wav_to_uv   - Convert a UV plane distance from wavelengths to UV units.
map_to_rad  - Convert an image plane distance from map units to radians.
rad_to_map  - Convert an image plane distance from radians to map units.

```

uvaver *aver_time, doscatter*

Time average a UV data set.

EXAMPLE

```
0>uvaver 240
Averaging into 240 second bins.
Selecting channels: 1 -> 1, polarization: RR.
0>
```

All but the first line above are informational messages from the 'uvaver' command.

PARAMETERS

aver_time - The duration of the new integrations in seconds.

doscatter - Default: FALSE

If doscatter=FALSE then the new visibility weights will be the sum of the old visibility weights. This is statistically correct under the assumption that the input weights equal or are proportional to the reciprocal of the amplitude variance. This assumption is based on the rules described in AIPS documentation for UV FITS files.

If doscatter=TRUE then the new visibilities will be given weights derived from the data by using the scatter within a solution interval to derive an estimate of the reciprocal amplitude variance. Effective use of this option requires that each output bin encompass many input integrations. In particular, if only one visibility exists in a given bin, uvaver will leave this visibility in place with its original weight, but make it flagged.

CONTEXT

There are three main things that people commonly use uvaver for.

1. The obvious application is to reduce redundantly high time sampling to speed up subsequent processing.
2. When the doscatter argument is true, the statistical weight of each averaged visibility is recomputed from the scatter of the original visibilities that contributed to it.

This is very useful when the weights in the original FITS file were not the reciprocal of the variance of the real and imaginary parts, as difmap expects. In such cases, unless uvaver is used to replace

these weights, then the `modelfit` command will not report the correct values for either chi-squared or the estimated uncertainties of fitted parameters.

3. Finally, a less obvious use is to bring together visibilities on different baselines that were sampled at slightly different times.

This is useful when the data that were read in from a FITS file had slightly different time-stamps on different baselines within the same logical integration. In this case when `difmap` reads such a file the end result is that each baseline is assigned to its own integration. Not only does this waste large amounts of memory and subsequent CPU time, but it also means that station-based editing in `vplot` actually does baseline-based editing, and `selfcal`, when run without a solution time, will not make any corrections due to the lack data within each integration. The solution is to re-bin the data into integrations with `selfcal`.

CAVEAT

Beware that long averaging times can significantly smooth real variations in the UV data. This can have harmful effects on the resulting map, especially on features in the image that are far from the center of the map. The effects include a reduction in flux that varies with spatial frequency and thus produces a smearing effect.

To avoid this, before you use the `uvaver` command, you are advised to use the `check_uvaver` command to get an idea of how much effect a given averaging time will have on important regions of the map.

See 'help check_uvaver' for more information.

CAUTION: *** If uvaver fails you may loose your data ***

When `difmap` reads a UV FITS file, it places the raw visibilities in a scratch file. This file is then read whenever the 'select' command is invoked, or when edits need to be applied. The `uvaver` command averages data in this file and writes the averaged integrations to a new scratch file before deleting the original. If it fails to write the new scratch file - perhaps because of insufficient disk space, then `difmap` will discard the current data. Thus you should check that you have sufficient disk space for the new file (look at the existing file called `uvdata.scr` to get an idea of the required disk space) before running `uvaver`. If you have done anything significant to the data, then it would also be prudent to save the merge file before running `uvaver`.

DETAILS

The uvaver command reads raw integration records from its existing uvdata scratch file and writes binned averaged integrations to a new uvdata file. When all integrations have been averaged, the old uvdata scratch file is deleted to be replaced by the new one.

For each new integration read within a given solution bin, uvaver adds to a weighted vector running mean of the data versus time and a weighted running mean of baseline U,V,W coordinates versus time. As each solution bin is completed the averaged integration record is written to the output uvdata scratch file. The new U,V,W coordinates and time-stamps are recorded in memory as before.

Solution bins are regularly spaced on a grid of width equal to the averaging time. Input integrations are binned onto this grid.

RELATED COMMANDS

check_uvaver - Check if a given averaging time would cause time smearing.

uvplot *station, umax, vmax, docursor*

Plot observed UV points on the UV plane.

EXAMPLE

```
0> uvplot "BONN"
0>
```

This example plots the positions of visibilities in the UV plane, plotting all but the data associated with BONN in green. BONN data from the first sub-array is plotted in a different color to highlight it. If the display device has a cursor then interactive mode will be invoked after the initial display has been completed.

PARAMETERS

- station - (Optional) The name of a station to highlight in the plot. If not given or given as an empty string "", no station will initially be highlighted. (Stations can be specified subsequently with the cursor).

- umax - (Optional) Default=0 (0 -specifies the whole range). The maximum U coordinate to display. The plot will go from -umin to +umin. Unless changed with the mapunits command, the units are million-wavelengths.

- vmax - (Optional) Default=0 (0 -specifies the whole range). The maximum V coordinate to display. The plot will go from -vmin to +vmin. Unless changed with the mapunits command, the units are million-wavelengths.

- docursor - (Optional) Default=TRUE. If the display device has a cursor but you don't want to enter the interactive cursor mode, make this FALSE.

CONTEXT

This task displays the UV coverage of all un-flagged visibilities selected with the 'select' command. For each baseline, one point is plotted per IF. Where multiple spectral-line channels are selected from a given IF, the visibility is the average of those channels and is displayed at the corresponding mean U and V coordinates.

INTERACTIVE MODE

If your display device has a cursor, then there are a number of

interactive options that you can use. These are selected by moving the cursor into the plot window, and pressing the following keys:

- X - Quit the uvplot session, and return to the difmap prompt.
- L - Re-display the plot.
- Z - Zoom in on a rectangular sub-plot of the displayed region. After pressing 'Z' either press 'Z' again to revert to the original un-zoomed display, or select the two vertexes of the desired sub-plot area by sequentially pressing the 'A' key (left-mouse-button) over each. To cancel an incomplete selection, press 'D' (middle-mouse-button).
- . - Toggle the marker symbol used to display the data between the smallest available and the next largest. By default the smallest possible marker symbol is used for speed, but if the data are sparse then the larger marker symbol may be more convenient.
- H - (Help) Briefly list all keyboard options.
- n - Highlight visibilities from the next telescope in the currently highlighted sub-array. Initially no telescope is highlighted. If this key then pressed, the visibilities of the first telescope of the first sub-array will be highlighted. Pressing this key again highlights the second telescope, etc.. When the last telescope of the current sub-array has been plotted, the first telescope of the next sub-array will be plotted. So by repeatedly pressing this key you will eventually have seen each telescope of all sub-arrays individually highlighted.
- p - This has the reverse the effect to the 'n' key, highlighting the visibilities of the previous telescope in the current sub-array, and wrapping back to the last telescope in the previous sub-array, after the first telescope of the current sub-array is highlighted.
- N - [Upper-case n] Similar to the lower-case 'n' key this key results in the first telescope of the next sub-array in the observation to be highlighted. In this way you can jump between sub-arrays without having to see all the telescopes of the intermediate sub-arrays highlighted one at a time.
- P - [Upper-case p] This has the reverse effect of the 'N' key, allowing one to directly step backwards through sub-arrays.
- T - To have a specific telescope of a specific sub-array highlighted without having to step through all the intermediate

telescopes and sub-arrays, press this key. On the text window you will then be prompted for the name of the telescope to be highlighted. Telescopes are specified by an optional sub-array number delimited by a colon, followed by an unambiguous telescope name prefix. If the sub-array number and colon are omitted then the sub-array from which telescopes are currently being highlighted will be assumed. For instance, both 3:BONN and BON refer to a telescope named BONN in sub-array 3 if another telescope of sub-array 3 is currently being highlighted. Also, provided that sub-array 2 does not contain more than one telescope starting with the prefix JBN, 2:JBN is sufficient to refer to a telescope called JBNK1 in sub-array 2.

- S - This causes uvplot to display details about the visibility nearest to the cursor. It shows the sub-array, baseline, IF number and date of the visibility.
- C - This initiates cut area selection, through which a rectangular area may be selected within which to flag data. After pressing 'C' you will be expected to enter the locations of two opposite corners of the area, by pressing 'A' (left-mouse-button) over the corresponding points of the plot. To cancel a half selected area press 'D' (central-mouse-button). Once two corners have been selected, all visibilities displayed in that area will be flagged and will then disappear from the plot.
- W - Toggle the scope of edits with respect to spectral-line channels. Each point in the plot comes from a single IF, and its position in the plot is determined by the mean frequency of the channels currently selected from that IF. By default, if you select such a point to be flagged, then all spectral-line channels of the source IF will be flagged, irrespective of the channels which actually went into making the plot. The alternative is to edit just the channels that went into making the plot. The 'W' key toggles which of the two options is used, and a mode line above the plot describes which option is currently in effect.
- + - Toggle whether the cursor is shown as a small cross or a cross-hair that extends the full width and height of the display. Currently only the /xserv and /xwin PGPLOT devices support this feature.
- % - Toggle whether to display the conjugate symmetric versions of each visibility. This option is enabled by default, so if you place this character in your 'uflags' variable (see below), it will be disabled when uvplot starts.

CONFIGURATION VARIABLES

The display-mode option keys listed above may be preset in the variable, 'uflags'. If uflags is an empty string, "", then the default option string is used. This is currently also an empty string. To override this default and have radplot start up by displaying with large dot sizes, on the command line one would type:

```
0> uflags="."
```

If you always want to override the default, then place such an assignment line in your difmap login file (the file named in the optional DIFMAP_LOGIN environment variable).

OTHER DISPLAY COMMANDS

```
radplot -   Display visibility amplitude vs. radius.
projplot -  Display visibility amplitude/phase vs. projected UV distance.
vplot      -  Display observed and model visibilities from the current IF.
tplot      -  Display the time-sampling of each telescope of the current IF.
corplot    -  Display the self-cal corrections of the current IF.
cpplot     -  Display observed and model closure phases interactively.
```

uvrange *uvmin, uvmax*

Set the min and max UV radii of usable visibilities.

EXAMPLE

```
0>uvrange 0.1,10
```

Only data in the UV range: 0.1 -> 10 (mega-wavelengths) will be used.

```
0>
```

The second line is the response from the 'uvrange' command, confirming the user's selection. To see what the currently assigned range is, type:

```
0>uvrange
```

Only data in the UV range: 0.1 -> 10 (mega-wavelengths) will be used.

```
0>
```

To cancel a previously selected range, type:

```
0>uvrange 0,0
```

The full UV range of the data is currently selected.

```
0>
```

PARAMETERS

uvmin - The minimum UV radius from which visibilities will be used during gridding, self-calibration, modelfitting and by the uvstat function. Unless changed with the mapunits command, the units are million-wavelengths.

uvmax - The maximum UV radius from which visibilities will be used during gridding, self-calibration, modelfitting and by the uvstat function. Unless changed with the mapunits command, the units are million-wavelengths.

If uvmax <= 0 or uvmin < 0 or uvmax < uvmin, no range is applied and any previous select range is cancelled.

eg. 'uvrange 0,0' cancels any previous select range.

Whenever a new UV data set is read with the 'observe' command, any existing values of uvmin and uvmax are discarded.

CONTEXT

This function effects subsequent invocations of the 'invert', 'selfcal', and 'modelfit' commands, plus the 'uvstat' function. The select range remains in force until this command is called again, or

until a new data-set is read with the 'observe' command.

The range is applied, such that all visibilities whose radial distance in the UV plane exceeds uvmax, or is less than uvmin, are ignored. If no select range has been specified, or zero values are given for both the min and the max, all visibilities are used regardless of their UV radii.

SIDE EFFECTS

Use of this command will cause the next 'clean', 'mapplot' etc.. to re-invert the modified UV data-set to produce an updated dirty beam and residual map.

RELATED COMMANDS

- uvtaper - Apply a gaussian taper to the gridding weights.
- uvweight - Selects combinations of natural, uniform, uncertainty, and radial weighting.
- uvrange - Constrain the min and max UV radii within which visibilities are gridded.
- uvzero - Set a zero-UV-spacing flux.

uvstat *statistic*

Return UV plane visibility statistics.

EXAMPLE

```

0>print uvstat(rms)
0.333969
0>
0>print uvstat(chisq)
1.07537
0>
0>print uvstat(nvis)
10756
0>
0>print uvstat(clphs_chisq)
0.230628
0>
0>print uvstat(clphs_nused)
21679
0>

```

PARAMETERS

statistic - The name of the statistic wanted. Recognised names are:

- | | | |
|--------------------|---|---|
| rms | - | The root-mean-square difference between observed and model visibilities (Jy). |
| chisq | - | The reduced chi-squared of the fit between the observed and model visibilities. |
| nvis | - | The number of un-flagged visibilities. |
| clphs_chisq | - | The reduced chi-squared of the discrepancy between the observed and model closure phases, divided by clphs_nused. |
| clphs_nused | - | The number of closure phases that were used to calculate clphs_rchisq. If no data are flagged then this will be the sum over all subarrays of:
$\text{ntime} * \text{nif} * \text{nt} * (\text{nt}-1) * (\text{nt}-2) / 6$ where ntime is the number of integrations, nif is the number of IFs, and nt is the number of telescopes in each subarray. |

CONTEXT

This function provides access to a few statistics relating model and observed visibilities in the UV plane. The return value may be used in

expressions, and thus may be used in automatic mapping procedures.

Note that the 'uvrange' command can be used to limit the statistics to those of visibilities that lie within a given distance from the center of the UV plane.

SIDE EFFECTS

The tentative model will be established in the UV plane if not already there. This in turn will force difmap to re-invert the data before the next operation that involves the current map.

RELATED COMMANDS

imstat - Return image plane map statistics.

uvrange - Set the range of visibilities to be inspected.

uvtaper *gaussian_value, gaussian_radius*

Set a gaussian taper to weight down long baselines and reduce sidelobes

EXAMPLE

```
0>uvtaper 0.5,10
```

```
Gaussian taper: value 0.5 at UV radius = 10 Mega-wavelengths
```

```
0>
```

The second line is the response from the 'uvtaper' command, confirming the selection made by the user. To see what the current values are without changing them:

```
0>uvtaper
```

```
Gaussian taper: value 0.5 at UV radius = 10 mega-wavelengths.
```

```
0>
```

To cancel a previously set taper type:

```
0>uvtap 0
```

```
No UV-taper is currently set.
```

```
0>
```

PARAMETERS

gaussian_value - The value of the gaussian taper at 'gaussian_radius'. Valid values lie between 0 -> 1 exclusive. Other values cause the cancellation of any existing gaussian taper.

gaussian_radius - The UV radius at which gaussian taper should have fallen to 'gaussian_value'. (values <= 0 de-select gaussian tapering). Unless changed with the mapunits command, the units are million-wavelengths.

To cancel a previous gaussian taper you should type: uvtaper 0
Whenever a new UV data set is read with the 'observe' command any existing gaussian taper is cancelled.

CONTEXT

Before the visibilities are inverted to the image plane they are first gridded. In the process of gridding, individual visibilities are weighted. Gaussian tapering can be applied in addition to the other forms of weighting specified with command 'uvweight'. It is generally used to weight down long baselines in order to increase the signal to noise ratio of low-surface brightness, extended emission or to reduce side-lobes during the initial stages of mapping. It does this at the expense of spatial resolution. The taper selected remains in force for use by all subsequent uses of the 'invert' command, until 'uvtaper'

is used again to change or cancel the taper.

SIDE EFFECTS

Use of this command will cause the next 'clean', 'mapplot' etc.. to re-invert the modified UV data-set to produce an updated dirty beam and residual map.

RELATED COMMANDS

- uvweight - Selects combinations of natural, uniform, uncertainty, and radial weighting.
- uvrange - Constrain the min and max UV radii within which visibilities are gridded.
- uvzero - Set a zero-UV-spacing flux.

uvweight *bin_size, error_power, do_radial*

Set the type and degree of weighting for subsequent UV gridding.

EXAMPLE

1. To select uniform weighting and to scale the uniform weighting grid weights by the errors implied by the data raised to the power -1, type:

```
0>uvweight 2,-1
Uniform weighting binwidth: 2 (pixels).
Gridding weights will be scaled by errors raised to the power -1.
Radial weighting is not currently selected.
0>
```

The first line illustrates a typical use of the command, and the following three lines are the response by the 'uvweight' function, confirming the choices made.

2. To see the current choice of weighting, omit all arguments:

```
0>uvweight
Uniform weighting binwidth: 2 (pixels).
Gridding weights will be scaled by errors raised to the power -1.
Radial weighting is not currently selected.
0>
```

PARAMETERS

- bin_size** - If this is > 0 then it selects uniform weighting. The value sets the bin size in UV-grid pixels within which to count neighboring visibilities. The reciprocal of these counts are then used to weight the visibilities by the local number density. A value of 2 is recommended. Non-integral values are legal but fractional values below 1 are not.
- error_pow** - If < 0 then this selects additional weighting by amplitude errors. The value selects the exponent to raise the error to. Statistically this should be -2, but -1 is recommended. Make sure that the relative errors between baselines are meaningful before using this option or you may end up with some baselines making no contribution to the final map!
- do_radial** - If true then the weight given to each visibility will be divided by its UV radius. Only recommended when one's UV coverage is approximately circular.

CONTEXT

This command sets the weights given to individual visibilities as they are gridded, prior to Fourier inversion into the image plane. The choices made remain in effect until this function is called again.

When a new data set is read with the 'observe' command, the defaults are reset to apply uniform weighting with a bin-width of 2 and no other form of weighting.

Unless the commands mentioned in the last section are of interest, you would typically follow this command by invoking the 'invert' command. (See help on 'invert' for details).

SIDE EFFECTS

Use of this command will cause the next 'clean', 'mapplot' etc.. to re-invert the modified UV data-set to produce an updated dirty beam and residual map.

RELATED COMMANDS

- uvtaper - Apply a gaussian taper to the weights.
- uvrange - Constrain the min and max UV radii within which visibilities are gridded.
- uvzero - Set a zero-UV-spacing flux.

uvzero *flux, weight*

Set the estimated flux expected from a baseline of zero length.

EXAMPLE

```
O>uvzero 10, 100
Zero-baseline flux set to 10 Jy. Weight=100.
O>
```

The second line is the response from the 'uvzero' command, confirming the user's selection. To see the value of the current zero-spacing flux omit all arguments:

```
O>uvzero
Zero-baseline flux set to 10 Jy. Weight=100.
O>
```

To cancel a previously set zero-baseline flux specify zero.

```
O>uvzero 0,0
Zero-baseline flux not set.
O>
```

PARAMETER

- flux - The estimated total flux of the observation (Jy). This will be treated as though it were the flux density of a normal visibility at U=0 and V=0.
- weight - The visibility weight to be assigned to the zero-baseline flux. This is the weight to be seen after application of wtscale and has the same form as the weights of normal visibilities. However note that the addition of a zero-baseline visibility is like the addition of a single extra sample in the UV plane and will have very little effect unless a large weight is used.

Whenever a new UV data set is read with the 'observe' command, the default value of flux=0.0 is set.

CONTEXT

If an observation is so deficient in short baselines that no value is added to the pixel at the origin of the U,V plane grid, then the total flux in the dirty map will equal zero (as required by Fourier transform theory). For very extended sources this may result in low brightness regions of the source appearing to be negative because the flux in every pixel of the map is offset by a small negative amount. The effect may be countered to some extent by adding a fake visibility at the origin of the UV plane. Unfortunately, while the

total estimated flux of the source can be used to set the amplitude of the visibility and the phase at the origin is known to be zero, the weight to give the visibility is a matter of experimentation. A weight value that is much higher than the average weight of the observed visibilities will be needed if any effect is to be seen. The best value will depend on the distribution of the observed visibilities and their weights.

In practice it is probably easier to get rid of negative bowls by model fitting rather than experimenting with the zero-spacing flux.

SIDE EFFECTS

Use of this command will cause the next 'clean', 'mapplot' etc.. to re-invert the modified UV data-set to produce an updated dirty beam and residual map.

vis_stats *observable*

Compute the statistics of a given observable.

EXAMPLES

When used as a command, the 'vis_stats' command prints the statistics of the specified observable to the terminal and the log file.

```
0>vis_stats amplitude
N=261 Mean=3.56569 +/- 0.177582 Scatter=2.86893 Min=0.0365752 Max=17.9686
```

When used as a function, vis_stats() returns an array containing the statistics of the specified observable.

```
0>print vis_stats(amplitude)
261 3.56569 0.177582 2.86893 0.0365752 17.9686
```

If you just want one of the statistics, you can index the returned array directly as follows.

```
0>print vis_stats(amplitude)(2)
3.56569
```

Otherwise if you want all of the values, rather than repeating the above for each statistic, it is more efficient to assign the results to a local array, and index the array.

```
0>float stats(6)
0>stats = vis_stats(amplitude)
0>fprintf 0, "Amplitude range = %g -> %g\n", stats(5), stats(6)
Amplitude range = 0.0365752 -> 17.9686
```

PARAMETERS

observable - The name of the observable who's statistics are wanted.
The recognized names are the following:

amplitude	-	The amplitude of each visibility.
phase	-	The phase of each visibility.
real	-	The real part of each visibility.
imaginary	-	The imaginary part of each visibility.
umag	-	The magnitude of the U coordinate of each visibility.
vmag	-	The magnitude of the V coordinate of each visibility.
uvrad	-	The radial distance of each visibility from the center of the UV plane.

CONTEXT

The 'vis_stats' procedure can be used either as a command or as a function. As a command it prints the statistics of the specified observable to the screen. As a function it returns an array containing the statistics. In each case the statistics are computed using just the visibilities that are within the current uvrage (see `uvrange.hlp`).

When used as a function, 'vis_stats' returns a one dimensional array of floating point numbers. The elements of this array record the following statistics, in the following order:

nvis	-	The number of visibilities used to compute the statistics.
mean	-	The mean value of the observable.
sigma	-	The standard deviation of the mean.
scatter	-	The root-mean-square scatter of the observable about the mean.
min	-	The minimum value of the observable.
max	-	The maximum value of the observable.

Note that strictly speaking the 'sigma' and 'scatter' values aren't very meaningful when looking at amplitudes or phases, since neither amplitude nor phase comes from a Gaussian distribution. They are however meaningful for the real and imaginary parts.

Also note that when asking for the statistics of visibility phases, the returned values are expressed in degrees.

The visibility plane is conjugate symmetric. This means that a visibility with value $a+ib$ at U,V is equivalent to a visibility of $a-ib$ at $-U,V$. Thus when computing the statistics of phases and imaginary parts, there is an ambiguity. In 'vis_stats' the ambiguity is resolved by placing all visibilities in the positive U part of the plane.

RELATED COMMANDS

radplot	-	Display visibility amplitude, phase, error vs. UV radius.
uvrange	-	Limit the UV range used by vis_stats and other commands.

vplot *nplots, reftel, IF_index, npage*

General visibility plotting and editing command.

PARAMETERS

-
- nplots** - (Optional) Default=0 (All baselines to each station).
The number of baselines to be displayed per page. This may be changed interactively. Since only the baselines of one station are displayed on one page, the maximum number of plots per page at a given time is 1 less than the number of stations in the sub-array being displayed.
- reftel** - (Optional) Default=""
A baseline specification or the form
- [sub-array-index:][antenna_name_1][antenna_name_2]
- On interactive devices this specifies which baseline should be plotted first. On non-interactive devices it specifies the range of baselines to be plotted. For instance on a non-interactive device "1:" means plot all baselines of sub-array 1, whereas on an interactive device, it means, start plotting baselines for the first available antenna of sub-array 1. For further details of baseline specifications, read the antenna_names help topic.
- IF_index** - (Optional) Default = The first sampled IF.
The index of the IF to start plotting visibilities from.
- npage** - (Optional) Default = 0 (with requests no page limit).
The maximum number of pages to plot when plotting to a non-interactive device.

CONFIGURATION VARIABLES

-
- vflags** - This is a string variable that over-rides the default display attributes. It is particularly useful when displaying to a hard-copy device, where interactive changes to the default attributes can not be made.

By default vflags="". In this case vplot substitutes the equivalent of vflags="efmb3".

When vflags is not "", all toggled attributes are initially set to off, and the contents of the vflags string determine which of the attributes get turned back on.

The entries in the vflags string are the keys used in interactive mode

to toggle the respective attributes on and off, and are interpreted exactly as though you had pressed them interactively. This means that if an option appears twice in vflags the second instance cancels the first! The available keys are listed further below.

For example after typing:

```
vflags="efm1"
```

Subsequent use of vplot after typing this will initially plot visibilities with error-bars (e), will show flagged-data (f), will show the model (m), and only display amplitudes (1).

CONTEXT

This command can be used in non-interactive mode to display visibility data on a hard-copy device, or interactively to examine and/or edit visibilities.

When displayed, flagged data appear as red '+' symbols, un-flagged data as green points and the model as a continuous light-blue line. In addition, if a visibility is flagged by a selfcal correction flag, but not by a normal visibility flag, it is displayed as a blue 'x' symbol.

Only visibilities from a single IF are displayed at one time, but the selected IF can be changed interactively, or preset on the command line.

HARD COPY MODE

If this command is invoked when the PGPLOT output device is a hard-copy device, such as a laser printer, one can specify which baselines to plot and how to arrange them. This is illustrated in the following examples:

1. Plot all of the baselines in the observation, grouped by telescope name and sub-array, displaying 5 baselines per page. This could take some time so be prepared to wait.

```
0> vplot 5
```

2. Plot all baselines from sub-array 1, at 5 baselines per page.

```
0> vplot 5, 1
```

3. Plot all HSTK baselines of sub-array 2 in IF 3, again at 5 baselines per page.

```
0> vplot 5, 2:HSTK, 3
```

4. Plot all HSTK baselines of sub-array 2, starting from baseline HSTK-BONN, using as many plots per page as there are telescopes in sub-array 2 (minus 1).

```
0> vplot 0, 2:hstk bonn
```

INTERACTIVE MODE

If the display device has a cursor, then an interactive session will be invoked. In this mode, one page of baselines is initially plotted and mouse-keys and keyboard keys are used to guide the task through plotting and data editing.

Once vplot has been invoked, move the cursor into the display window and press the 'H' key on your keyboard. Whenever this is pressed during the vplot session a list of key bindings will be displayed in the terminal window from which vplot was invoked.

Most keys are case-insensitive, such that both 'H' and 'h' have the same effect.

On standard PGPLOT conforming devices with a mouse or other cursor control device, mouse buttons are treated as equivalent to the following keyboard keys:

```
Left button   = 'A'   (This is the normal key for selecting positions).
Middle button = 'D'   (This is used to cancel incomplete select ranges).
Right button  = 'X'   (Use this to exit vplot).
```

Note that on a windowing system, the cursor must be within the display window before pressing keyboard keys (and mouse buttons) will be noticed. However, after pressing a key that needs more complex answers, such as the entry of the number of baselines to display or the entry of a telescope name, the cursor should be moved to the original text window where a prompt will be displayed to guide your entry.

MOVING ABOUT BETWEEN BASELINES

```
n - (Lower-case n) Display next set of baselines
p - (Lower-case p) Display preceding set of baselines
```

Pressing 'n' in vplot causes the Next set of baselines of the current telescope to be displayed from the current sub-array. Once all baselines to that telescope have been displayed, pressing 'n' will proceed to display those of the next telescope in the observation. Once all telescopes have been displayed in this manner, pressing 'n' will display start to display baselines from the next sub-array. So by repeatedly pressing 'n' you will see all baselines in every

sub-array of the observation. The 'p' key works identically but shows the Previous set of baselines.

N - (Upper-case n) Display baselines from the next sub-array.

P - (Upper-case p) Display baselines from the previous sub-array.

Upper-case 'N' and 'P' provide a means to jump to the next or previous sub-arrays without having to view all of the intermediate baselines.

T - Specify reference telescope from the keyboard (text screen)

To move to another telescope without having to wade through all the baselines of previous telescopes in the observation, press 'T'. Back on the text screen you will be prompted for the name of the telescope. The name entered may be any unambiguous abbreviation of the full telescope name.

eg BOLOGNA and BONN are ambiguous up to BO. BOL or BON are the shortest names that could be used to refer to them.

A optional second telescope name may also be entered on the same input line, separated from the first by spaces or a hyphen, in which case the combination of the two telescope names will be used to designate the first baseline to be displayed to the new reference telescope.

If the required telescope(s) are from a different sub-array than the one currently being displayed, then specify the sub-array number preceding the first telescope name and separated from it with a colon. Thus BONN-BOL selects baseline BONN-BOLOGNA in the currently displayed sub-array, while 3:BONN-BOL selects baseline BONN-BOLOGNA in sub-array 3.

MOVING BETWEEN IFs

Visibilities are only displayed from one IF at a time. The identity of this IF is displayed above the collection of plots. To move onto the next or previous IF in which any channels were selected with the 'select' command, use the following keys:

] - Move to the next sampled IF.

[- Move to the previous sampled IF.

The title above the plots will change to indicate which IF is being displayed.

CHANGING HOW BASELINES ARE DISPLAYED

The following keys toggle attributes on or off. The display will not be re-plotted immediately whenever one of these is toggled, so you can press a sequence of toggle keys before re-displaying with

the new attributes. When the sequence is complete, simply press any key that is not in the list below, such as the RETURN key and the display will be updated with the new attributes.

- M - Toggle display of model visibilities
- F - Toggle display of flagged visibilities
- E - Toggle display of error bars
- 1 - Plot only amplitudes.
- 2 - Plot only phases.
- 3 - Plot both amplitudes and phases.
- B - Toggle breaking the plot into scans (where present)
- V - Toggle inclusion of flagged data in auto-scaling
- - Toggle between the default of displaying absolute amplitudes and/or phases, and the option of displaying the amplitudes and/or phases of the vector difference between the data and the current model.

NB. By default, a scan is taken to be any set of integrations which is separated by more than an hour from its nearest neighbors. This can be changed with the 'scangap' command.

CHANGING THE NUMBER OF BASELINES DISPLAYED

- S - Select number of sub-plots per page

To change the number of baselines displayed, press the 'S' key. You will be prompted for a number on the text screen. The first baseline plotted will be the same as before, but only the requested number will follow.

CHANGING THE DISPLAYED UT RANGE

- U - select UT range to be displayed (hit U twice for full range)

Press 'U' to request a change in the displayed UT range. You will then be expected either to press 'U' again to display the whole UT range, or to move the cursor to the start UT of the required range and press 'A' (left mouse button), then move the cursor to the end UT of the required range and press 'A' again. The plot will then be re-displayed over the new range and all subsequent plots will be displayed over this range. At any point before the UT range selection has been completed, pressing the 'D' key (middle mouse button) will quietly abort the selection.

SWITCHING BETWEEN TIME SYSTEMS

- G - Toggle between displaying the default Universal Coordinated

Time and Greenwich Means Sidereal Time along the X axis
of the plot.

CHANGING THE DISPLAYED AMPLITUDE OR PHASE RANGE

Z - Select amplitude or phase range to be displayed.

Press 'Z' to zoom in on a selected amplitude or phase range. You should then either to press 'Z' again to revert to the default range, or move the cursor to the first (top or bottom) amplitude or phase value of the required vertical range and press 'A' (left mouse button), then move the cursor up or down to the end of the required range and press 'A' again. The plot will then be re-displayed over the new range and all subsequent plots will be displayed over this range. At any point before the selection has been completed, pressing the 'D' key (middle mouse button) will quietly abort the selection.

CHANGING WHETHER REDUNDANT BASELINES ARE DISPLAYED

By default, for each reference telescope that it plots, vplot displays all of the baselines that include that telescope. This is recommended because it aids in finding telescope dependent errors. However, it does result in you seeing each baseline twice if you step through all antennas. This can be annoying if you have a very large number of baselines. For this reason vplot also provides the option to only show each baseline once, by only showing baselines in which the second antenna of the baseline comes later in the antenna listing than the reference antenna. Thus for the first antenna you see all of the baselines for that antenna, but for the second you see one less baseline, and so on, until by the time you reach the last antenna, there are no baselines to be plotted.

0 - Toggle telescope ordering to only show non-redundant baselines.

EDITING DATA

There are two main modes for editing and it is important to select the right one before proceeding.

Station based edits flag or un-flag the selected visibilities on ALL baselines that include the current reference telescope. If you have more than one baseline displayed at a time, then this will be evident, as the selected visibilities will change color on all the displayed baselines.

Baseline based editing only affects the baseline in which the selection was made.

- (SPACE BAR) Toggles station based vs. baseline based editing.

The current editing mode is displayed above the plots on the right-hand side of the display and can be toggled by pressing the SPACE-BAR.

It is also possible to further specify the scope of edits.

- I - Toggle IF editing scope.
- W - Toggle spectral-line editing scope.

By default, edits are applied to all polarizations, spectral-line channels and IFs related to the selected visibility. To specify that edits only be applied to the displayed IF, press the 'I' key. To specify that edits be only applied to the channels that were specified with the 'select' command, press the 'W' key. The mode line above the collection of plots, describes which of these editing modes is in effect.

- A - Flag or unflag the nearest visibility to the cursor.
(Left mouse button)

Pressing 'A' on the keyboard or the left mouse button near a displayed visibility point causes the status of that visibility to be toggled from un-flagged to flagged or vice versa. If flagged data are displayed, then toggling an un-flagged visibility will produce a red cross signifying its new flagged status. Otherwise the point will magically disappear, but can be seen subsequently if the display of flagged data is enabled.

- C - Flag all data inside select box
- R - Restore data inside select box

If you wish to flag or un-flag an extended region of data, point by point editing is tedious. The Clip and Restore keys, 'C' and 'R' enable you to edit all points inside a square box selected with the cursor. To use this facility, press the respective key and then move the cursor to one corner of the required region and press 'A' (left mouse button). Then move the cursor to the opposite corner of the required region and again press 'A'. The selected points will be re-displayed with their changed statuses. Note that it is especially important to know whether station or baseline based editing is required before doing this since in station based editing restoring a region that you have just clipped may restore points on related baselines that were originally clipped for other reasons.

FLAGGING A WHOLE SCAN

To flag a whole scan on a given baseline, move the cursor into the selected scan on the baseline to be edited and press the 'Z' (ie. zap)

key. All points on that baseline within the scan will be flagged, even those not in the currently displayed UT range. In this context a scan is either a single scan obtained after using the 'B' (break-into-scans) key, or the whole UT time range otherwise. Note that the current station/baseline editing mode has no effect on the behavior of this command - so only one baseline will be flagged at a time.

ENDING A vplot SESSION

To end a vplot session, press 'X' or the right mouse button. If substantial editing was performed in vplot then it is advisable to use the wmerge command to save the modified file before continuing to map the data.

MISCELLANEOUS features

The following interactive key bindings have not been discussed above.

- L - When you press this key the current plot is re-displayed.
- + - This key toggles whether the cursor is shown as a small cross or as a cross-hair that extends the full width and height of the display. Currently only the /xserve and /xwindow PGPLOT devices support this feature.

RELATED COMMANDS

 scangap - Change the time gap used to delimit neighboring scans.
 radplot - Display visibility amplitude vs. UV radius.
 projplot - Display visibility amplitude/phase vs. projected UV distance.
 uvplot - Display the sampling of the UV plane.
 tplot - Display the time-sampling of each telescope.
 corplot - Display accumulated self-calibration corrections.
 cpplot - Display observed and model closure phases interactively.

wav_to_uv *distance*

Convert a UV plane distance from wavelengths to UV units.

EXAMPLE

The following example converts 10000 wavelengths to kilo-wavelengths.

```
0>mapunits arcmin
Distances in the map plane now have units: arcmin.
Distances in the UV plane now have units: kilo-wavelengths.
0>print wav_to_uv(10000)
10
0>
```

PARAMETERS

distance - A distance in the UV plane, expressed in wavelengths.

CONTEXT

Use of the 'mapunits' command to specify which units should be used for specifying and displaying distances in the image plane, also changes the units that are used for the display and entry of UV coordinates.

The 'wav_to_uv()' function allows the user to convert a distance in wavelengths, into these units.

RELATED COMMANDS

uv_to_wav - Convert a UV plane distance from UV units to wavelengths.
 map_to_rad - Convert an image plane distance from map units to radians.
 rad_to_map - Convert an image plane distance from radians to map units.

wbeam *file_name*

Save the dirty beam to a FITS file.

EXAMPLES

```
0>wbeam beam.fits
```

```
Writing dirty beam to FITS file: beam.fits
```

```
0>
```

PARAMETERS

file_name - The file name to be given to the new FITS file.

CONTEXT

'wbeam' can be used to save the restored beam to a FITS file at any point during mapping. Many other programs, such as AIPS can read such files.

The whole area of the beam grid is saved.

RELATED COMMANDS

wmap - Write the clean (restored) map to a FITS file.

wdmap - Save the dirty (residual) map.

wobs - Write the UV data to a random-groups UV FITS file.

wmodel - Write the CLEAN model to a (text) VLBI model file.

wwins - Write CLEAN windows to a text file.

save - Save UV data, models, windows and the restored map.

wdmap *file_name*

Save the residual dirty map to a FITS file.

EXAMPLES

```
O>wdmap dirty.fits
Writing dirty map to FITS file: dirty.fits
O>
```

PARAMETERS

file_name - The file name to be given to the new FITS file.

CONTEXT

'wdmap' can be used to save the residual dirty map to a FITS file at any point during mapping. Many other programs, such as AIPS can read such files. If necessary the map will be inverted first. If a model exists then it is also recorded in an 'AIPS CC' binary extension table.

Only the inner cleaned region (the same as the region that mapplot displays by default) is saved to the file.

RELATED COMMANDS

```
wmap    - Saves the clean (restored) map.
wbeam   - Writes the dirty beam to a FITS file.
wobs    - Writes the UV data to a random-groups UV FITS file.
wmodel  - Write the CLEAN model to a (text) VLBI model file.
wwins   - Write CLEAN windows to a text file.
save    - Save UV data, models, windows and the restored map.
```

winmod *discard_selected*

Delete all components outside (or inside) the current clean windows.

EXAMPLE

0>winmod

The latest clean model now contains 31 of the original 92 components.

The established clean model now contains 65 of the original 72 components.

0>

PARAMETERS

discard_selected - (Default=false)

If false then discard all model points outside the current clean windows.

If true then discard all model points inside the current clean windows.

CONTEXT

This command is intended for the situation where one finds that the choice of clean windows in a previous clean allowed model components to be placed in suspect positions. Running this command allows one to constrain the model components to lie within the latest estimate of the clean windows. You should beware of the fact that the components so deleted may have had effects on the flux within the windows, so run mapplot again after calling this command to see how the deleted parts of the model effected the residual map and be prepared to continue cleaning a bit further.

SIDE EFFECTS

Use of this function will cause the next command that requires the residual map, to re-invert it.

RELATED COMMANDS

mapplot - Use this to look at the model components as well as the map.

wmap *file_name*

Save the clean (restored) map to a FITS file.

EXAMPLES

```
0>wmap map.fits
```

```
Writing clean map to FITS file: map.fits
```

```
0>
```

PARAMETERS

file_name - The file name to be given to the new FITS file.

CONTEXT

'wmap' can be used to save the restored map to a FITS file at any point during mapping. Many other programs, such as AIPS can read such files. If necessary the map will be restored first. The model used to restore the map is also recorded in an 'AIPS CC' binary extension table.

Only the inner cleaned region (the same as the region that mapplot displays by default) is saved to the file.

RELATED COMMANDS

wdmap - Save the dirty (residual) map.

wbeam - Write the dirty beam to a FITS file.

wobs - Write the UV data to a random-groups UV FITS file.

wmodel - Write the CLEAN model to a (text) VLBI model file.

wwins - Write CLEAN windows to a text file.

save - Save UV data, models, windows and the restored map.

wmarkers *file_name*

List the commands that would create the current list of map markers.

EXAMPLE

```
0>wmodel markers.list
Writing marker commands to markers.list
0>
```

This file can be executed subsequently to restore the list of markers, by typing:

```
0>clear_markers
0>@markers.list
```

PARAMETERS

file_name - (Default=standard output)
 If a file name is given, the marker commands are written to that file. Otherwise, if the file-name argument is either omitted, or specified as "", the marker commands are written to standard output for display on your terminal and in the log file.

CONTEXT

This command serves two purposes. Without a filename argument, it allows the user to see the current list of map markers interactively. With a filename it allows the user to write a command file which can later be executed to reproduce the current list of markers.

RELATED COMMANDS

```
save               - Save all data files and the difmap running parameters.
mark_xy            - Specify a map marker relative to the map center.
mark_radec         - Specify a map marker to be drawn at a given RA,Dec.
clear_markers      - Delete the current list of markers.
delmarker          - Delete a given map marker.
```

wmodel *file_name, do_continuum*

Write CLEAN models to a VLBI model file.

EXAMPLE

0>wmodel test.mod

Writing 26 model components to file: test.mod

0>

PARAMETERS

file_name - (Default=standard output)

The name to be given to the new model file. If this argument is omitted or "" is specified, then the model will be displayed on the standard output and also in the log file.

do_continuum - (Default=false)

If omitted or given as FALSE, the normal established and tentative models will be written. If it is given as TRUE, then just the continuum model (if any) will be written.

CONTEXT

This command is used to save both the established followed by the latest clean models in one file. On a later date the model can be re-read using the 'rmodel' command and used as a starting model.

It can also be used to save the continuum model, if one exists, by presenting the optional second argument with the value TRUE.

RELATED COMMANDS

rmodel - Read a new model from a file.

showmodel - Report detailed information about the current models.

save - Save all data files and the difmap running parameters.

wobs *file_name, do_shift*

Save UV data to a random-groups UV FITS file.

EXAMPLE

```
O>wobs 3c345.uvf
Writing UV FITS file: 3c345.uvf
O>
```

ARGUMENTS

```
file_name - The name to give the new FITS file.
do_shift  - (Default=false)
            By default the recorded data don't include any shifts
            applied with the 'shift' command. This is the mode used
            by the 'save' command, which arranges instead to reapply
            the shift with another 'shift' command on rereading the
            data into difmap. To force difmap to freeze the accumulated
            position shifts in the output file, specify this argument
            as true.
```

CONTEXT

Once UV data have been read into difmap with the 'observe' command, it may be saved to a new random-groups UV FITS file with the 'wobs' command. If visibilities have been edited then the new file will contain the edited data. Any edits that were applied in difmap, as well as self-calibration and residual offset corrections, will be applied to the saved data. This command is automatically called by the 'save' command.

RELATED COMMANDS

```
observe - Read a new random-groups UV FITS file.
wmap    - Write the clean (restored) map to a FITS file.
wdmap   - Save the dirty (residual) map.
wbeam   - Write the dirty beam to a FITS file.
wmodel  - Write the CLEAN model to a (text) VLBI model file.
wwins   - Write CLEAN windows to a text file.
save    - Save UV data, models, windows and the restored map.
```

write_models *filename*

Write a table of models, indexed by channel-range/polarization, to a file.

EXAMPLE

```
0>write_models 3c84.mtab
```

PARAMETERS

filename - The name of the file to write the models to.

CONTEXT

In multi-model mode (see help multi_model), different models are associated with different channel-range/polarizations selections. This command, in conjunction with the read_models command allow one to save and restore these models. In particular, it is used by the 'save' command for this purpose. Each model in the file, is written in the same format as by the 'wmodel' command, with the exception that each model is preceded by a select line, specifying the polarization and channel range to which the model refers.

RELATED COMMANDS

```
multi_model - Switch to and from multi-model mode.
read_models - Restore a set of multi-model models from a file.
clear_models - Discard all multi-model models.
```

wtscale *scale*

Change the scale factor that multiplies the raw visibility weights.

EXAMPLE

```

0>wtscale
Current weight scale factor is: 1
0>
0>wtscale 15
Changed weight scale factor to: 15
0>
0>wtscale wtscale()*2.3
Current weight scale factor is: 15
Changed weight scale factor to: 34.5
0>

```

PARAMETERS

scale - The new scale factor to multiply the raw visibility weights by. Note that increasing the weights corresponds to decreasing the apparent visibility amplitude uncertainties, and vice versa. In particular, to scale the error bars by a factor C, the argument of the wtscale command would have to be $1/C^2$.

CONTEXT

In UV FITS files each visibility is given a weight that is proportional to the reciprocal of the associated visibility amplitude variance. Unfortunately the constant of proportionality that is optionally given in the same files, is rarely anywhere near the right magnitude, let alone correct. The two adverse results are that 'vplot' displays amplitude uncertainty error bars that are totally non-sensical, and that the self-calibration commands report scaled Chi-squared values that make it difficult to judge the significance of the fit.

The 'wtscale' command has been provided to enable one to change the scale factor. In some cases it is possible to deduce the appropriate weight scale factor by knowing where the data originated from, but in other cases getting the "right" factor is unfortunately a matter of guess work. Trial and error based on inspection of the error bars in 'vplot' is the only way to make this guess at the moment. Note that scaling the weights by X scales the error bars by $1/\sqrt{X}$.

If the data-set is written back to a UV FITS file the weights will be left un-scaled, but the accumulated scale factor will be recorded via the AIPS mechanism of writing a HISTORY line of the form:

```
HISTORY AIPS WTSCAL=....
```

Both AIPS and Difmap recognise such lines, so if the file is later read back into difmap or AIPS the weight scale factor will be correctly recovered. Note that since history lines are written incrementally, there may be many history lines of the above form. The last such line is the one actually used.

VLA DATA

The VLA writes weights equal to the number of 10s intervals in an integration. The correct weight scale factor can be determined from the equation for the expected noise variance:

$$\frac{1}{\text{variance}} = \frac{(1\text{e-}26)^2 * \text{Bw.t.}(\text{A.na.nc})^2}{2.0 * (\text{k.Tsys})^2}$$

Boltzmann's constant : k = 1.38066e-23 J/K
 System temperature : Tsys = 40K (typical for the VLA).
 Observing bandwidth : Bw
 Integration time : t = const * 10s
 Antenna area : A = 490.8739 m^2
 Antenna efficiency : na = 0.63
 Correlator efficiency: nc

The correlator efficiency depends on the VLA observing mode, but is documented in the synthesis imaging workshop book.

Thus given that the raw data weights equal the number of 10 second intervals in an integration, the correct value of wtscale is:

$$\text{wtscale} = \text{Bw} * \text{nc}^2 * 1.5678\text{e-}4$$

Note that when used to combine two stokes parameters into a derived polarization (eg. RR+LL->I), AIPS SPLIT doesn't combine the weights correctly. If you are using data that have been processed in this fashion, the above wtscale factor should be scaled by a factor of 2. If you combine RR+LL->I with 'select' in difmap there is no need for this.

wwins *winfile, do_old*

Save CLEAN windows to a text file.

EXAMPLE

```

O>wwins "myfile.win"
wwins: Wrote 9 windows to myfile.win
O>

```

PARAMETERS

```

winfile  - (Default=standard output)
           The name to call the new window file. If this argument is
           omitted or "" is specified, then the windows will be
           displayed on the standard output and also in the log file.

do_old    - (Default=FALSE)
           If TRUE then write the windows in the format required
           by the Caltech Package 'clean' program, ie.
           LRTB = n,n,n,n,....

```

CONTEXT

CLEAN windows are set interactively in mapplot and/or directly via the addwin command. The resulting list of windows may be saved to a file using the 'wwins' command and retrieved later using the 'rwins' command. This makes it possible to use the same windows for data at a different epoch or to continue cleaning a data-set at a later date using the same windows.

Please see help on rwins for a description of the default format used.

If do_old is specified as TRUE then the alternate format used by the 'clean' program in the Caltech VLBI Package is written. This format can NOT be read back into difmap.

RELATED COMMANDS

rwins - Read previously saved CLEAN windows from a text file.

xysize *xa, xb, ya, yb*

Set the bounds of the sub-image to be displayed in mapplot.

EXAMPLE

```
O>xysize -10,20,0,0
The mapplot display area will be limited to:
  Relative RA: -10 -> 20 (milli-arcsec)
  Relative DEC: (The whole available range)
O>
```

The second and subsequent lines are confirmatory messages produced by the xysize command.

PARAMETERS

- xa - The X (center-relative RA) coordinate of one edge of the required sub-image. Unless changed with the mapunits command, the units are milli-arcsec.
- xb - The X (center-relative RA) coordinate of the opposite edge of the required sub-image to xa.
- ya - The Y (center-relative DEC) coordinate of one edge of the required sub-image. Unless changed with the mapunits command, the units are milli-arcsec.
- yb - The Y (center-relative DEC) coordinate (milli-arcsec) of the opposite edge of the required sub-image to ya.

CONTEXT

By default, the whole available map or beam area is displayed when you invoke mapplot. To change this behavior use the xysize command to select the required sub-image for subsequent calls to mapplot. Alternatively, the displayed area can be changed interactively with the cursor and optionally saved (as though xysize had been called) for later calls to mapplot - see help mapplot.

RELATED COMMANDS

- mapplot - Display map and beam images.
- loglevs - Set logarithmic contour levels for mapplot.
- xysize - Set the boundaries of the sub-image to be displayed in mapplot.
- mapcolor - Select mapplot colormap type and transfer function.
- mapfunc - Change the transfer function used in mapplot.
- beamloc - Change when and where mapplot plots the clean beam ellipse.