IMAGER

A GILDAS software

November 19th, 2018

Version 1.0

Questions? Comments? Bug reports? Mail to: `gildas@iram.fr`

The GILDAS team welcomes an acknowledgment in publications
using GILDAS software to reduce and/or analyze data.
Please use the following reference in your publications:
`http://www.iram.fr/IRAMFR/GILDAS`

Documentation
In charge: E. Di Folco[3].
Active developers: S. Guilloteau[3].
Main past contributors: J. Pety[1,2].

Software
In charge: S. Guilloteau[3].
Active developers: S. Bardeau[1,2].

1. IRAM
2. Observatoire de Paris
3. Laboratoire d'Astrophysique de Bordeaux

Related information is available in:

- IRAM Plateau de Bure Interferometer: Introduction
- IRAM Plateau de Bure Interferometer: OBS Users Guide
- IRAM Plateau de Bure Interferometer: Atmospheric Calibration
- IRAM Plateau de Bure Interferometer: Calibration Cookbook
- CLIC: Continuum and Line Interferometric Calibration
- MIS: Millimeter Interferometry Simulation Tools
- GREG: Graphical Possibilities
- SIC: Command Line Interpretor

Contents

1 Warning

This document is under construction; it is partly based on the older documentation of MAPPING**.**

IMAGER is an interferometric imaging package, tailored for usage simplicity and efficiency for multi-spectral data sets. IMAGER **is *** NOT ***** MAPPING

Although the underlying algorithms are the same as in the MAPPING program, the overall concepts for user interaction differ:

- The user interface to MAPPING is file-oriented. The user interacts with MAPPING generally by a script, packaging a complex sequence of commands, tasks, reads and writes to intermediate files.
- The basic concept of IMAGER is opposite: a single `READ` of data, a few simple processing commands with built-in intelligent parameter guesses, a visual user control, and a single `WRITE` of the results once the user is satisfied of it.

2 Basic concepts

2.1 Objectives

The main goals of IMAGER are

1. to offer a proper implementation of imaging in case of wide relative bandwidth, where the natural angular resolution changes with frequency.
2. to implement a simpler (and incidentally faster) scheme to process Mosaics, including short spacings from single dish data
3. to minimize image sizes
4. to minimize processing time by using parallel programming as much as possible and reducing Input/Output to the strict minimum.
5. to simplify user interfaces, by providing sensible defaults.
6. to take advantage of improved capabilities of NOEMA and ALMA, by offering new tools like self-calibration or wide bandwidth analysis.

IMAGER was developed and optimized to handle large data files. Therefore, IMAGER works mostly on internal buffers and avoids as much as possible saving data to intermediate files. File saving is done ultimately once the data analysis process is complete, which offers an optimum use of the disk bandwidth.

2.2 Overview of the data reduction and analysis

Once the data has been acquired by an interferometer such as the NOrthern Extended Millimter Array (NOEMA) or ALMA, two different approaches may be used for its reduction and analysis:

- The first possibility is to clearly separate 1) the calibration, 2) the imaging and deconvolution and 3) the analysis.
- The second possibility is to merge in a single step calibration and imaging. This possibility is known as self-calibration.

While CASA uses the second paradigm, GILDAS mainly implements the first approach, as the program and the data format used for each step is different. The calibration of NOEMA data is done inside CLIC on the NOEMA raw data format and the outcome is a *uv* table, which contains only calibrated visibilities of the astronomical source. The imaging and deconvolution is done inside IMAGER on the calibrated *uv* table and deliver mainly an `lmv` spectral cube (2 axes of coordinates and 1 axis of velocity/frequency). Finally, the GREG program implements several generic tools to visualize and analyze `lmv` spectral cubes, which are not specific to an interferometric use (*e.g.* they can be used with the IRAM 30 m spectral cubes as well). IMAGER includes GREG for the visualization and analysis functionalities.

The choice of clearly separating calibration and imaging+deconvolution was taken at start of the Plateau de Bure Interferometer (PdBI), when the limiting number of antennas prevented the use of self-calibration. While many points of the calibration algorithms inside CLIC are specific to NOEMA data (in particular its range of Signal-to-Noise ratio), the algorithms of imaging+deconvolution can be used in many different contexts and the visualization and analysis of spectra cubes is mainly independent of the instrument that delivered the data. This last

point implies that users can import data (mainly through FITS format) in IMAGER for imaging and deconvolution, and in GREG for visualization and analysis. But the reverse is also true. While calibration of NOEMA data should be done inside CLIC, imaging+deconvolution and visualization+analysis can be done in other softwares (*e.g.* MIRIAD, AIPS, CASA for the imaging and deconvolution and KARMA for the visualization and analysis).

With the improvement of NOEMA (increase of the number of antennas and better receiver sensitivities) and with the advent of a new generation of interferometer (ALMA), an additional step of self-calibration may improve the consistency of the final results by imposing additional consistent constraints on the calibration. This step is further presented in chapter 4.

2.3 The structure of the IMAGER program

2.3.1 Structure and recommendations

The IMAGER program supports

- The manipulation (*e.g.* resampling), visualization and flagging of *uv* tables;
- The imaging of *uv* tables in dirty maps and beams;
- The deconvolution of dirty maps;
- The inclusion of short-spacings;
- The visualization and analysis of spectra cubes;
- The self-calibration.

It consists in a collection of commands, either dedicated to image and deconvolution (the CLEAN\ language) or implementing basic functionalities (the SIC\, GREG\, or CALIBRATE\ families of languages).

2.4 Imaging/deconvolution: a brief sequence of commands

For the user, IMAGER reduces the number of actions to the strict minimum. The imaging sequence is always the same:

```
1- Reading data
READ UV MyData.uvt /RANGE Min Max Type
! here, optionally use UV_TIME, UV_COMPRESS, UV_BASELINE to average data
! or UV_FILTER, UV_CONT to filter lines or remove continuum
2- Imaging
UV_MAP
3- Deconvolving
CLEAN
! here, optionally use UV_RESTORE
4- Looking at the result
VIEW CLEAN      ! or SHOW CLEAN
5- Writing the result
WRITE * MyData
```

- **Step 1:** Reading the specified internal buffer (here UV) from the input file (.uvt), loading only the channels falling in the range defined by the variables Min and Max, of Type CHANNEL, VELOCITY or FREQUENCY. IMAGER recognizes whether the UV table is for a single field or a mosaic. The only difference between the single field and mosaic cases is that IMAGER yields a Sky brightness image for Mosaics, while the computed sky brightness of a single field is not automatically corrected for the primary beam attenuation. Imaging for multiple fields will be presented in chapter 4. Single Dish data can also be loaded in the following way : READ SINGLE File.

- **Step 2:** Computing a dirty map and beam from a UV data. UV_MAP processes single fields as well as Mosaics.

- **Step 3:** Deconvolving the DIRTY image map (a Single-field or Mosaic) using the dirty BEAM with the current METHOD. The default for the SIC variable METHOD is HOGBOM, the other supported methods being CLARK, MRC, MULTI and SDI. See CLEAN ? for the other SIC variables controlling the deconvolution process. The outputs are the CLEAN and RESIDUAL images, and the Clean Component Table CCT, all being stored in dedicated SIC variables.

- **Step 4:** Plotting the result in the specified internal buffer (CLEAN). Optionaly, the user can restrict the plot to a subset of channels through the optional arguments First and Last. SHOW CLEAN can also be used instead, and produces a different type of plot.

- **Step 5:** Writing all modified image-like buffers (not the UV tables) under the common file name "MyData". In the case of the present example, the following files are produced: MyData.lmv, MyData.lmv-clean, MyData.cct, MyData.beam, which correspond to the buffers: DIRTY, CLEAN, CCT, and BEAM, respectively. WRITEUV MyData would only write the internal buffer (UV) in the file MyData.uvt (the default extension corresponds to the specified buffer).

2.4.1 Implementation issues

The new implementation of the UV_MAP command uses most of the older code, but re-arranged such that ensembles of contiguous channels ("chunks") are treated at once and share the same synthesized beam. Deconvolution with CLEAN then proceeds by using the synthesized beam with the appropriate frequency for each channel. The user can control the "chunk" size, and hence the precision of the process given the desired field of view.

As a result of the new concept, beams (whether primary or synthesized) can be 4-D arrays, as they may depend on Frequency and Field.

2.5 Usage of the HELP

A simple call to HELP will display the various languages (e.g., SIC\, GREG\, CALIBRATE\, CLEAN\) accessible to the help documentation and list some commands with available documentation. Note that CLEAN is a command and CLEAN\ a language (i.e., a family of commands). The language of a given command is recalled in the help of each command.

Example: the command APPLY belongs to the language CALIBRATE\, it has one argument which can be AMPLI or PHASE exclusively, one optional argument gain, and one option /FLAG.

```
IMAGER> help apply
        [CALIBRATE\]APPLY [AMPLI|PHASE [gain]] [/FLAG]
```

A brief description of the imager program can be obtained through:

```
IMAGER> help imager
USER\IMAGER = "@ welcome.ima"

      IMAGER is a interferometric imaging package, tailored for usage
  simplicity and efficiency for multi-spectral data sets.

     The basic concept of IMAGER is the use of a simple
         READ data  - ACTION(s) - [SHOW or VIEW] - WRITE
  sequence of commands, minimizing the data I/O as much as possible.
  Automatic guesses of appropriate default values for the ACTIONs
  parameters is implemented whenever possible.

Additional Help Available:
 Actions      MAPPING      UV_Handling  MAP_Handling
```

To further explore the difference between IMAGER and MAPPING:

```
IMAGER> help imager mapping
USER\IMAGER = "@ welcome.ima"
IMAGER MAPPING
     Caution:  IMAGER is *** NOT *** MAPPING

     Although the underlying algorithms are the same as in the
  MAPPING program, the concepts are quite different.

    The user interface to MAPPING is file-oriented. The user interacts
  with MAPPING generally by a script, packaging a complex sequence of
  commands, tasks, reads and writes to intermediate files. Fine control
  of the script parameters are in general done through widgets.

    The basic concept of IMAGER is opposite: a single READ of data, a
  few simple processing commands with built-in intelligent parameter
  guesses, a visual user control, and a single WRITE of the results
  once the user is satisfied of it.
```

Finding documentation and help for the IMAGER commands can be done in three different ways:

- a simple call to the `HELP` command provides a description of the command and its arguments and options
- the command name followed by one or more questions marks will display some partial help on the command and its most useful parameters ("`Command ?`"), its second level parameters for advanced users ("`Command ??`"), all its parameters ("`Command ???`).
- `INPUT Command` provides a list of default values for the most commonly used parameters.

Documentation on subtopics of a given command (e.g., Variables, Arguments, or Results) can be obtained though: HELP command subtopic. (Warning: subtopic is case sensitive!). The list of available subtopics is found at the bottom of the documentation of each command:

```
IMAGER> help uv_map
[...]
Additional Help Available:
 Mosaics      /FIELDS      /TRUNCATE    Variables    MAP_BEAM_STE MAP_CELL
 MAP_CENTER   MAP_CONVOLUT MAP_FIELD    MAP_POWER    MAP_PRECIS   MAP_ROBUST
 MAP_ROUNDING MAP_SHIFT    MAP_SIZE     MAP_TAPEREXP MAP_TRUNCATE MAP_UVTAPER
 MAP_UVCELL   MAP_VERSION  MAP_WEIGHT   MCOL         WCOL         Old_Names:
 convolution  map_angle    map_dec      map_ra       uv_taper     uv_cell
 taper_expo   weight_mode
```

Example: the following command will list the control variables of the UV_MAP function and describe the associated parameter(s):

```
IMAGER> help uv_map Variables
UV_MAP Variables
                [CLEAN\]UV_MAP ?
        Will list all MAP_* variables controlling the UV_MAP parameters.

    The  list  of  control  variables  is (by alphabetic order, with the old
    names used by Mapping on the right)
    New names       [   unit]       -- Description --    % Old Name
    MAP_BEAM_STEP   [       ]  Number of channels per single dirty beam
    MAP_CELL        [ arcsec]  Image pixel size
    MAP_CENTER      [ string]  RA, Dec of map center, and Position Angle
    MAP_CONVOLUTION [       ]  Convolution function    % CONVOLUTION
    MAP_FIELD       [ arcsec]  Map field of view
    MAP_POWER       [       ]  Maximum exponent of 3 and 5 allowed in MAP_SIZE
    MAP_PRECIS      [       ]  Fraction of pixel tolerance on beam matching
    MAP_ROBUST      [       ]  Robustness factor        % UV_CELL[2]
    MAP_ROUNDING    [       ]  Precision of MAP_SIZE
    MAP_SIZE        [       ]  Number of pixels
    MAP_TAPEREXPO   [       ]  Taper exponent           % TAPER_EXPO
    MAP_TRUNCATE    [      %]  Mosaic truncation level
    MAP_UVCELL      [      m]  UV cell size             % UV_CELL[1]
    MAP_UVTAPER     [m,m,deg]  Gaussian taper           % UV_TAPER
    MAP_VERSION     [       ]  Code version (0 new, -1 old)

    NAME is no longer used, and WEIGHT_MODE is obsolete.
    MAP_RA          [  hours]  RA of map center
    MAP_DEC         [    deg]  Dec of map center
    MAP_ANGLE       [    deg]  Map position angle
    MAP_SHIFT       [Yes/No ]  Shift phase center
    are obsolescent, superseded by MAP_CENTER.
    They are  provided  only  for compatibility with older scripts.
```

A more detailed description (type, size) of a given variable can be obtained through "help command variable", as in this example:

```
IMAGER> help uv_map map_uvtaper
UV_MAP MAP_UVTAPER

      MAP_UVTAPER[3]  Real

    Parameters of the tapering function (Gaussian if MAP_TAPEREXPO = 2): ma-
    jor axis at 1/e level [m], minor axis at 1/e level [m], and position an-
    gle [deg].
```

MAP_UVTAPER requires 3 values of type Real.

The default values of the useful parameters are checked through `Command ?`[1]

```
IMAGER> uv_map ?

 UV_MAP makes a dirty image and a dirty beam from the UV data

* Variable MAP_CENTER controls shifting and rotation
* MAP_CELL[2], MAP_SIZE[2], MAP_FIELD[2] control the map sampling
* MAP_UVTAPER[3], MAP_UVCELL and MAP_ROBUST
     control the beam shape and weighting scheme
* MAP_BEAM_STEP and MAP_PRECIS control the dirty beam precision

  Map Size (pixels)                MAP_SIZE        [ 0 0 ]
  Field of view (arcsec)           MAP_FIELD       [ 0 0 ]
  Pixel size (arcsec)              MAP_CELL        [ 0 0 ]
  Map center                       MAP_CENTER      [  ]
  Robust weighting parameter       MAP_ROBUST      [ 0 ]
  UV cell size (meter)             MAP_UVCELL      [ 7.5 ]
  UV Taper (m,m,deg)               MAP_UVTAPER     [ 0 0 0 ] MAP_TAPEREXPO [ 2 ]
  Channels per single beam         MAP_BEAM_STEP   [ 0 ]
  Tolerange at map edge (pixels)   MAP_PRECIS      [ 0.1]
  Rounding method                  MAP_POWER       [ 2 ]     MAP_ROUNDING [ 0.05 ]
  Gridding Convolution method      MAP_CONVOLUTION [ 5 ]
```

Example: The default Gridding Convolution method is number 5 (Spheroidal).

2.5.1 Notes for MAPPING users

The names of variables and most commands have been kept from MAPPING, old names appear in the `HELP` whenever they have been replaced. In addition, for the sake of compatibility, SIC procedures can still be activated as in MAPPING for the sake of compatibility, although most of the `GO` procedures should have been replaced by a dedicated IMAGER command or procedure. For instance, `GO PLOT` (or its variants `GO BIT`, `GO NICE` and `GO MAP`) and `GO UVSHOW` offer similar

[1] Users familiar with MAPPING can still use `INPUT Command` instead, although the output format may be slightly different.

features to the SHOW command, but take data from files or SIC image variables, depending on variables NAME and TYPE.

3 UV Tables

The main goal of IMAGER is to convert interferometric measurements stored in *uv* tables into images suitable for astrophysical interpretation. *uv* tables contain a set of visibilities. *uv* tables being the starting point, IMAGER contains a number of commands to handle them.

3.1 *uv* table description

3.1.1 *uv* table data format

A *uv* table is a specific 2-D Gildas table, with a few additional informations in the header, and a special interpretation of the data organisation.

In a standard *uv* table, each *line* describes a visibility. Here a *line* designate either the first or second axis of the table, and a *column* the other one. *uv* tables may appear in both orders. The default one is *line* on 1st axis (`.uvt` ordering, used by most application). The `.tuv` ordering obtained by a 21 transposition is used essentially for display, as in this case the *column* as the same meaning as for the `COLUMN` of GREG .

The number of lines of a *uv* table is thus the number of visibilities described in the table. Each *column* of the table stores a particular property of the visibilities, namely:

Column 1 U in meters;

Column 2 V in meters;

Column 3 Scan number;

Column 4 Observation date (integer CLASS/CLIC Day Number[2]);

Column 5 Time in seconds since 0:00 UT of above date;

Column 6 Number of the first antenna used to measure the visibility;

Column 7 Number of the second antenna used to measure the visibility;

Column 8 Real part for the first frequency channel;

Column 9 Imaginary part for the first frequency channel;

Column 10 Weight for the first frequency channel;

Columns 11-13 Same as column 8-10 but for the second frequency channel, or for the second Stokes parameter of this channel.

... etc. . . for all channels

Columns N-ntrail+1 ... N Trailing columns after the channel visibilities.

If a *uv* table describes `nvis` visibility spectra composed of `nchan` frequency channels, each with `nstokes` Stokes parameters, the size of the table will thus be: `nvis` lines of `7+3*nchan*nstokes+ntrail` columns, where `ntrail` is the number of trailing column.

[2]The CLASS/CLIC is a "radio Julian date" (or "Jansky Julian date"), which starts as -2^{15} on the date of the first radio observation by Karl Jansky. It is thus the Modified Julian date minus 60549. That choice was made to maximize the time interval over which radio astronomical data could be usefully stored in an `integer*2`, back when 2 bytes of header space per spectrum were a significant consideration. This date has little meaning outside the rather sparse community of souls gathered around the CLASS and CLIC programs, however...

3.1.2 *uv* header

A *uv* table header contains all the informations of a GDF header but some of these informations have a special meaning in this context. Command `HEADER` is the standard way inside GILDAS to display in a human readable way the header of GDF file. For instance, the command
`IMAGER> header gag_demo:demo-line.uvt`
would display

```
 1  W-GDF,  UNKNOWN Velocity type defaulted to LSR
 2  File : /Users/guilloteau/gildas/gildas-exe-dev/demo/demo-line.uvt  REAL*4
 3  Size        Reference Pixel           Value                   Increment
 4       103   16.0000000000000       220398.688000000     -0.183792725205421
 5      9146   0.00000000000000       0.00000000000000       1.00000000000000
 6  Blanking value and tolerance      1.23455997E+34   0.0000000
 7  Source name         GG_TAU
 8  Map unit            Jy
 9  Axis type           UV-DATA      RANDOM
10  Coordinate system   EQUATORIAL          Velocity    LSR
11  Right Ascension   04:32:30.34200        Declination       17:31:40.5230
12  Lii        0.000000000000000            Bii       0.000000000000000
13  Equinox            2000.0000
14  Projection type     AZIMUTHAL           Angle     0.000000000000000
15  Axis 0     A0     04:32:30.34200        Axis 0     D0     17:31:40.5230
16  Baselines               0.0       0.0
17  Axis 1 Line Name    13CO(21)            Rest Frequency   220398.6880000000
18  Resolution in Velocity   0.25000000     in Frequency        -0.18379273
19  Offset in Velocity        6.3000002     Doppler Velocity     -40.755900
20  Beam                   0.00               0.00                  0.00
21  NO Noise level
22  NO Proper motion
23  NO Telescope section
24  UV Data    Channels:     32, Stokes: 1 None        Visibilities:        9146
25  Column            1 (Size 1) contains U
26  Column            2 (Size 1) contains V
27  Column            4 (Size 1) contains DATE
28  Column            5 (Size 1) contains TIME
29  Column            6 (Size 1) contains IANT
30  Column            7 (Size 1) contains JANT
31  Column            3 (Size 1) contains SCAN
```

Comments:

Line 1 Indicates the velocity frame. If not present in the table (as here), it is assumed to be LSR.

Line 2 Indicates the filename associated to the currently displayed header.

Lines 3-5 Display the dimensions of the associated array. Here it is a rank 2 array of dimension 9146 lines times 9146 lines, *i.e.* 9146 visibility spectra of 32 frequency channels. Line 4 describes the frequency axis of the visibility spectra stored in the *uv* table. Be careful that this is a convention, *i.e.* it must be decoded using the particular form of the table. In

our case, each spectra has 32 frequency channels of width -183.8 kHz, the frequency of the reference pixel 16.0 corresponding to 220398.688 MHz. This last frequency is the frequency delivered by the correlator, *i.e.* seen by the observatory. In particular, this is the frequency that must be used to compute the primary beam of the interferometer.

Line 8 Indicates the unit of the real and imaginary parts of the visibilities, normally the Jansky (Jy).

Line 9 Indicates that this is *uv* table (`UV-DATA` and `RANDOM`).

Lines 10-13 Describe the coordinate system.

Lines 14-15 Describe the projection system. In the *uv* table format, `A0` and `D0` indicate the phase center while `Right Ascension` and `Declination` indicate where the antenna pointed when acquiring the signal. These information are in general identical for single field imaging and different for mosaicing.

Lines 16 Indicates the baseline range in meters (m).

Lines 17-19 Describe additional information about the frequency axis of the visibility spectra. In particular, the rest frequency (here 220398.688 MHz, that of the ^{13}CO J=2-1 line) corresponding to a velocity of 6.3 km/s in the velocity frame indicated at line 1 (in general LSR).

Line 20 Indicates the primary beam size of the interferometer in radian. This is an obsolescent way to pass the size of the interferometer antennas.

Line 21 The noise section has no meaning for the UV table.

Line 22 If present, proper motions are given in mas/yr. The epoch is used as the time origin.

Line 23 If the TELESCOPE section is present, this line would indicate telescope name, its geographic coordinates and the antenna diameter (in m). This is the new way to specify the primary beam.

Line 24 UV data section: number of channels, number of Stokes parameters and number of visibilities.

Line 15 to end Special columns description.

3.2 UV Table handling

Besides the `READ UV` and `WRITE UV` commands to read or write *uv* tables, IMAGER has a number of commands to manipulate the current *uv* table buffer. These commands have names starting by `UV_`. Most of them are in the `CLEAN\` language, some in the `NEWSTUFF\` one.

IMAGER works using UV buffers. Most commands only work on the current UV buffer, but some of them keep track of the previous buffer to allow the user to revert the operation.

Data inspection and editing :

- `SHOW COVERAGE` display the *uv* coverage
- `SHOW UV` display the *uv* data
- `UV_FLAG` allow flagging visibilities
- `UV_PREVIEW` provides a quick view of the visibilities as a function of frequencies, and attempts to automatically find the continuum level and parts of the bandwidth with spectral line emissions.

Data size reduction routines :

- UV_COMPRESS is a simple spectral smoothing, providing only channel averaging by integer number of channels.
- UV_RESAMPLE provides a more flexible spectral smoothing and resampling facility.
- UV_TIME can be used to time-average the UV data set, leading to faster processing. However, using UV_TIME too early may limit your ability to perform accurate phase self-calibration.

Continuum processing commands :

- UV_BASELINE allows to remove the continuum baseline, by 0th or 1st order baseline fitting of each visibility.
- Conversely, UV_FILTER will filter the spectral line range to leave only the channels with continuum emission. Both UV_BASELINE and UV_FILTER can use the results provided by UV_PREVIEW to specify where spectral lines may be found.
- UV_CONTINUUM converts a spectral line uv table into a bandwidth synthesis continuum uv table. UV_CONTINUUM requires some knowledge of the image size to evaluate how many channels should be averaged together. This is done using the same parameters and subroutines as for commands UV_STAT SETUP and UV_MAP.

Image preparation :

- UV_CHECK inspects the uv data to figure out how many different synthesized beams are needed.
- UV_SHORT adds the short (or zero) spacing information provided by an additional single dish data, read by READ SINGLE.
- UV_STAT evaluates the impact of robust weighting and tapering on the synthesized beam. It provides recommandations for the image and pixel sizes.
- UV_TRUNCATE restricts the uv baseline length range.

Miscellaneous :

- UV_DEPROJECT de-project the (u, v) coordinates given a specified phase center, orientation and inclination. This can be useful for inclined, flattened, nearly axi-symmetric structures such as proto-planetary disks or galaxies.
- UV_RADIAL computes the azimutal average of the visibilities. It is useful for rotationally symmetric structures such as proto-planetary disks, for example.
- UV_REWEIGHT changes the visibility weights.
- UV_SHIFT changes the phase center

The remaining UV_... commands are related to imaging and deconvolution: UV_MAP computes the dirty image, UV_RESTORE computes the Clean image from a Clean component list by removal of the Clean components in the uv plane, and imaging of the residuals. UV_RESIDUAL just computes the residuals by subtraction of the Clean components.

Finally, UV_SELF, in the CALIBRATE\ language, is a specific variant of UV_MAP used to compute the intermediate images required for self-calibration. It is not intended for direct use by normal users.

4 Single-field imaging and deconvolution

4.1 In a nutshell

```
1 read uv YourData
2 uv_map
3 clean
4 view clean
5 write * YourResult
```

1. Read your *uv* data
2. Image
3. Deconvolve
4. see the result
5. Save the result if OK.

You are done. And this is often good. However, it may take a while, and the angular resolution and/or the brightness sensitivity may not be optimal. So, it may be worth for you to read the information below and adjust the control variables of `UV_MAP`

4.2 Measurement equation and other definitions

The measurement equation of an instrument is the relationship between the sky intensity and the measured quantities. The measurement equation for a millimeter interferometer is to a good approximation (after calibration)

$$V(u,v) = \mathrm{FT}\left\{B_{\mathrm{primary}}.I_{\mathrm{source}}\right\}(u,v) + N \tag{1}$$

where $\mathrm{FT}\{F\}(u,v)$ is the bi-dimensional Fourier transform of the function F taken at the spatial frequency (u,v), I_{source} the sky intensity distribution, B_{primary} the primary beam of the interferometer (almost a Gaussian whose FWHM is the natural resolution of the single-dish antenna composing the interferometer), N some thermal noise and $V(u,v)$ the calibrated visibility at the spatial frequency (u,v). This measurement equation implies different kinds of problems.

1. The presence of noise leads to sensitivity problems.
2. The presence of the Fourier transform implies that visibilities belongs to the Fourier space while most (radio)astronomers are used to interpret images. A step of *imaging* is thus required to go from the *uv* plane to the image plane.
3. The multiplication of the sky intensity by the primary beam implies a distortion of the information about the intensity distribution of the source.
4. Finally, the main problem implied by this measurement equation is certainly the irregular, limited sampling of the *uv* plane because it implies that the information about the source intensity distribution is incomplete.

Deconvolution techniques are needed to overcome the incomplete sampling of the *uv* plane. To show how this can be done, we need additional definitions

- Let us call $V = \mathrm{FT}\left\{B_{\mathrm{primary}}.I_{\mathrm{source}}\right\}$ the continuous visibility function.

- The sampling function S is defined as
  - $S(u,v) = 1/\sigma^2$ at (u,v) spatial frequencies where visibilities are measured by the interferometer. σ is the rms noise predicted from the system temperature, antenna efficiency, integration time and bandwidth. The sampling function thus contains information on the relative weights of each visibility.
  - $S(u,v) = 0$ elsewhere.
- We finally call $B_{\rm dirty} = {\rm FT}^{-1}\{S\}$ the dirty beam.

If we forget about the noise, we can thus rewrite the measurement equation as

$$I_{\rm dirty} = {\rm FT}^{-1}\{S.V\}\,. \tag{2}$$

Using the property #1 of the Fourier transform (see Appendix), we obtain

$$I_{\rm dirty} = B_{\rm dirty} * \{B_{\rm primary}.I_{\rm source}\}\,, \tag{3}$$

where $*$ is the convolution symbol. Thus, the incompleteness of the uv sampling translates into the image plane as a convolution by the dirty beam, implying the need of deconvolution. From the last equation, it is easy to show that the dirty beam is the point spread function of the interferometer, *i.e.* its response at a point source. Indeed, for a point source at the phase center, $\{B_{\rm primary}.I_{\rm source}\} = I_{\rm point}$ at the phase center and 0 elsewhere and the convolution with a point source is equal to the simple product: $I_{\rm dirty} = B_{\rm dirty}.I_{\rm point} = B_{\rm dirty}$ for a point source of intensity $I_{\rm point} = 1$ Jy.

We note that Fourier transform are in general done through Fast Fourier Transform, which implies first a stage of re-interpolation of the visibilities on a regular grid in the uv plane, a process called *gridding*. This gridding step introduces a convolution in the uv space, and thus a multiplication by the Fourier transform of the gridding function in the image plane, which needs to be corrected later by division by this Fourier transform. It can be shown that despite this step, the convolution property mentionned before still holds.

4.3 Imaging

The process known as *imaging* consists in computing the dirty image and the dirty beam from the measured visibilities and the sampling function.

4.3.1 Image size and pixel size

Link between image size and uv cell size The gridding stage requires at least Nyquist sampling of the uv plane to avoid the artifact known as aliasing. This sampling depends on the source size.

For the signal, the source size is limited by the primary beam, so that the Nyquist sampling in the uv plane is obtained with a size of the grid cells equals to half the size of the antenna diameter. (this is the smallest spatial frequency that the interferometer can be sensitive to, *i.e.* the natural resolution in the uv plane). In the image plane, this implies to make an image at least twice as large as the primary beam size (see Fourier transform property #2 in Appendix).

Unfortunately, the spatial frequencies of the noise are not bound: the noise increases at the edges of the produced image because of the noise aliasing and gridding correction.

Unless you have good reasons (such as a strong confusion source close to the primary beam), you should not choose too large an image size, since that would slow down imaging and deconvolution.

Link between pixel size and largest uv spatial frequency The largest sampled spatial frequency is directly linked to the synthesized beam size (*i.e.* the interferometer spatial resolution). The pixel size must be at least 1/2 the synthesized beam size to ensure Nyquist sampling in the image plane. However, Nyquist sampling is enough only when dealing with linear processes while deconvolution techniques are non-linear. It is thus recommended to select a pixel size between 1/3 and 1/4 of the synthesized beam to ease the deconvolution. Smaller pixel sizes would lead to larger images, and unduly slow down the imaging and deconvolution process.

4.3.2 Weighting and Tapering

The use of the visibility weights ($1/\sigma^2$) in the definition of the sampling function is called natural weighting as it is natural to weight each visibility by the inverse of noise variance. Natural weighting is also the way to maximize the point source sensitivity in the final image. However, the exact scaling of the sampling function is an additional degree of freedom in the imaging process. In particular, the user may change this scaling to give more or less weight to the long or short spatial frequencies.

We can thus introduce a weighting function $W(u, v)$ in the definitions of $B_{\rm dirty}$ and $I_{\rm dirty}$

$$B_{\rm dirty} = {\rm FT}^{-1}\left\{W.S\right\} \tag{4}$$

and

$$I_{\rm dirty} = {\rm FT}^{-1}\left\{W.S.V\right\}. \tag{5}$$

There are two main categories of weighting functions

Robust weighting In this case, W is computed to enhance the contribution of the large spatial frequencies. This is done by first computing the natural weight in each cell of the uv plane. Then W is derived so that

- The product $W.S$ in a uv cell is set to a constant if the natural weight is larger that a given threshold;
- $W = 1$ (*i.e.* natural weighting) otherwise.

This decreases the weight of the well measured uv cells (*i.e.* very low noise cells) while it keeps natural weighting of the noisy cells. It happens that the cells of the outer uv plane (corresponding to the large interferometer configurations) are often noisier than the cells of the inner uv plane (just because there are less cells in the inner uv plane). Robust weighting thus increases the spatial resolution by emphasizing the large spatial frequencies at moderate cost in sensitivity for point sources (but with a larger loss for extended sources, see below).

Tapering is the apodization of the uv coverage by simple multiplication by a Gaussian

$$W = \exp\left\{-\frac{\left(u^2+v^2\right)}{t^2}\right\}, \tag{6}$$

where t is the tapering distance. This multiplication in the uv plane translates into a convolution by a Gaussian in the image plane, *i.e.* a smoothing of the result. The only purpose of this is to increase the sensitivity to extended structure. Tapering should never be used **alone** as this somehow implies that you throw away large spatial frequencies measured

by the interferometer. It is only a way to extract the most information from the given data set. If you need more sensitivity to extended structures, use compact configuration of the arrays rather than extended configurations and tapering.

For more details on the whole imaging process the interested reader is referred to Guilloteau (2000).

4.3.3 Implementation (READ UV, UV_MAP and UV_STAT)

In IMAGER, gridding in the *uv* plane and computation of the dirty beam and image are coded in the UV_MAP command. This command works on an internal buffer containing the *uv* table read from a file through the READ UV command.

The UV_MAP command is controlled by a set of SIC variables named with the prefix MAP_. Suitable defaults are provided, so that only specific cases should require customization by the user. A description of the all variables can be obtained through HELP UV_MAP. UV_MAP ? also gives their default and current values.

Basic usage - image characterization

Name	Dim/Type	Description
MAP_CELL	[2] Real	Pixel size in arcsecond. Enter 0,0 to let the task find the best values.
MAP_FIELD	[2] Real	Image size in arcsecond. MAP_FIELD has precedence over the number of pixels MAP_SIZE to define the actual map size when both variables are non-zero.
MAP_SIZE	[2] Int	Image size in pixels
MAP_POWER	Int	Rounding scheme for default image size, to numbers like $2^n 3^p 5^q$. p and q are less than or equal to MAP_POWER. Default value is 2, for smallest image size. For MAP_POWER = 0 MAP_SIZE i
MAP_ROUNDING	Real	Maximum error between optimal size (MAP_FIELD / MAP_CELL) and round
MAP_CENTER	Char.	A character string to specify the new Map center and the new map orientation, see the next subsection related to the definition of the projection center of the image.

Weighting :

MAP_ROBUST	Real	Robust weighting factor, in range 0 - $+\infty$. 0 means Natural weighting (as $+\infty$, actually). 0.5 or 1 is usually a good choice for Robust Weighting. Default is 0,i.e. natural weighting. (Old name UV_CELL[1])
MAP_UVTAPER	[3] Real	Array of 3 values controlling the UV taper: major/minor axis at 1/e level [m,m] (first two values), and position angle ([deg], third valu By default (0,0,0). (Old name UV_TAPER[3]).
MAP_UVCELL	[2] Real	UV Cell size for Robust weighting. Default is 0, meaning that the cell size is derived from the antenna diameter. (Old name UV_CELL[2])
MAP_TAPEREXPO	Real	the taper exponent. Default 2, indicating Gaussian function. (Old name TAPER_EXPO).

Advanced

MAP_BEAM_STEP	Int	Number of channels per common dirty beam, if > 0. If 0 (default value), only one beam is produced in total. If -1, an automatic guess is performed from the map size and requested precision (MAP_PRECIS).
MAP_PRECIS	Real	Position precision at the map edge, in fraction of pixel size, used (with the actual image size) to derive the actual number of channels which can share the same beam. Default value is 0.1.
MAP_TRUNCATE	Real	For a Mosaic, truncate the primary beam to the specified level (in fraction). Default value is 0.2.

Debug

MAP_CONVOLUTION	Int	Gridding convolution mode in the *uv* plane. (default 5 for speroidal functions) (old name CONVOLUTION)
MAP_VERSION	Int	Version of code to be used. This is a temporary variable to allow comparison between the new and old codes without quitting IMAGER.

Parameters can be listed by the commands "UV_MAP ?" and "CLEAN ?". A thorough description of each parameter can be obtained by typing: "help UV_MAP MAP_*" or "help CLEAN CLEAN_*"

```
IMAGER> UV_MAP ?
 UV_MAP makes a dirty image and a dirty beam from the UV data

* Variable MAP_CENTER controls shifting ang rotation
* MAP_CELL[2], MAP_SIZE[2], MAP_FIELD[2] control the map sampling
* MAP_UVTAPER[3], MAP_UVCELL and MAP_ROBUST
     control the beam shape and weighting scheme
* MAP_BEAM_STEP and MAP_PRECIS control the dirty beam precision

  Map Size (pixels)                MAP_SIZE        [0 0]
  Field of view (arcsec)           MAP_FIELD       [0 0]
  Pixel size (arcsec)              MAP_CELL        [0 0]
  Map center                       MAP_CENTER      [ ]
  Robust weighting parameter       MAP_ROBUST      [0]
  UV cell size (meter)             MAP_UVCELL      [7.5]
  UV Taper (m,m,deg)               MAP_UVTAPER     [0 0 0] MAP_TAPEREXPO [2]
  Channels per single beam         MAP_BEAM_STEP   [0]
  Tolerance at map edge (pixels)   MAP_PRECIS      [0.1]
  Rounding method                  MAP_POWER       [2]     MAP_ROUNDING [0.05]
  Gridding Convolution method      MAP_CONVOLUTION [5]
```

4.3.4 Defining the Projection Center of the image

The command UV_MAP handles phase tracking center through its arguments, or through the string variable MAP_CENTER.

```
UV_MAP [CenterX CenterY UNIT  [Angle]]  [/FIELDS  FieldList]  [/TRUNCATE Percent]
```

4.3.5 Typical imaging session

```
 1 read uv gag_demo:demo-single
 2 uv_map ?
 3 uv_stat weight
 4 let map_robust 0.5
 5 uv_map ?
 6 uv_map
 7 show beam
 8 show dirty
 9 let map_size 128
10 uv_map
11 show beam
12 show dirty
13 hardcopy demo-dirty /dev eps
14 write dirty demo
15 write beam demo
```

Comments:

Step 1 Read the `demo.uvt` *uv* table in an internal buffer.

Step 2 Check current state of the variables that control the imaging.

Steps 3-5 Select the robust weighting threshold (step 4) from the result of the UV_STAT command (step 3) and recheck the current state of the variables that control the imaging (step 5).

Steps 6-8 Image and plot the dirty beam and image.

Steps 9-12 Try a smaller size of the map as the default imaged field-of-view looked too large from previous plots.

Steps 13 Make a Post-Script file from the dirty image.

Steps 14-15 Write dirty image and beam in `demo.lmv` and `demo.beam` files for deconvolution in a future IMAGER session. These steps are optional as you may directly proceed to the deconvolution stage without writing the files.

4.4 Deconvolution

4.4.1 Principle

Once the dirty beam and the dirty image have been calculated, we want to derive an astronomically meaningful result, ideally the sky brightness. However, it is extremely difficult to recover the intrinsic brightness distribution with an interferometer. Mathematically, the incomplete sampling of the *uv* plane implies that there is an infinite number of intensity distributions which are compatible with the constraints given by the measured visibilities. Fortunately, physics allow us

to select some solutions from the infinite number that mathematics authorize. The goal of deconvolution is thus to find a sensible intensity distribution compatible with the measured visibilities. To reach this goal, deconvolution needs 1) some *a priori*, physically valid, assumptions about the source intensity distribution and 2) as much knowledge as possible about the dirty beam and the noise properties (in radioastronomy, both are well known). The best solution would obviously be to avoid deconvolution, *i.e.* to get a Gaussian dirty beam. For instance, the design of the compact configuration of ALMA has been thought with this goal in mind. However, this goal is out of reach for today's millimeter interferometers, even ALMA.

The simplest *a priori* knowledge that the user can feed to deconvolution algorithm is a rough idea of the emitting region in the source. The user defines a support inside which the signal is to be found while the outside is only made of sidelobes. The definition of a support considerably helps the convergence of deconvolution algorithms because it decreases the complexity of the problem (*i.e.* the size of the space to be searched for solutions). However, it can introduce important biases in the final solution if the support excludes part of the sky region that is really emitting. Support must be thus used with caution.

4.5 The family of `CLEAN` algorithms (`HOGBOM`, `CLARK`, `MX`, `SDI`, `MRC`, `MULTI`)

Radio astronomy interferometry made a significant step forward with the introduction of a robust deconvolution algorithm, known as `CLEAN`, by Högbom (1974).

4.5.1 `CLEAN` ideas

The family of `CLEAN` algorithms is based on the *a priori* assumption that the sky intensity distribution is a collection of point sources. The algorithms have three main steps

Initialization

- of the residual map to the dirty map;
- and of the clean component list to a `NULL` (*i.e.* zero) value.

Iterative search for point sources on the residual map. As those point sources are found,

- they are subtracted from the residual map;
- and then they are logged in the clean component list.

Restoration of the clean map 1) by convolution of the clean component list with the clean beam, *i.e.* a Gaussian whose size matches the synthesized beam size and 2) by addition of the residual map.

Stopping criteria Several criteria may be used to stop the iterative search of this "matching pursuit":

1. When the maximum of the absolute value of the residual map is lower than a fraction of the noise. This stopping criterion is adapted to *noise limited situations*, *i.e.* when empirical measures of the noise in the cleaned image give a value similar to the noise value estimated from the system temperatures.

2. When the maximum of the absolute value of the residual map is lower than a fraction of the maximum intensity of the original dirty map. This stopping criterion is adapted to *dynamic*

limited situations, *i.e.* when some part of the source is so intense that the associated side lobes are larger than the thermal noise. In this case, any empirical measure of the noise in the cleaned image will give a value larger than the noise value estimated from the system temperatures.

3. The total number of clean components. This is a sanity criterium in case the other ones would be badly tuned.

4. When the total flux remains stable.

Choosing the good stopping criterion is important because the deconvolution must go deep enough to be correct but `CLEAN` algorithms start to diverge when the noise is cleaned too deep. Criterium 4 is thus in general preferable, but may lead to insufficient cleaning when the dirty beam is poor (by lack of uv coverage and/or because of phase noise). If (# 4) fails, a good compromise is to clean down to or slightly below (typically 0.5σ) the noise level.

Stability criterion Clean convergence is controlled by the usual `ARES` (#1) , `FRES` (#2) and `NITER` (#3) criteria, plus `CLEAN_NCYCLE` for methods with major cycles. However, these criteria can be set to 0, allowing IMAGER to automatically guess when to stop using criterium #4. In this case, convergence is controlled by `CLEAN_NKEEP`, which is a number of components. Deconvolution of a given channel stops if the cumulative flux at iteration number N is smaller (resp. larger) than at iteration $N-$`CLEAN_NKEEP` for positive signals (resp. negative). In essence, `CLEAN_NKEEP` is the number of components when only noise is present. Experimentation with various types of images has shown that `CLEAN_KEEP`$= 70$ is a good compromise.

Formation of the `CLEAN` map The clean component list may be searched on an arbitrarily fine spatial grid without too much physical sense as the interferometer has a finite spatial resolution. The convolution by the clean beam thus reintroduces the finite resolution of the observation, an information which is missing from the list of clean component alone. This step is often called *a posteriori* regularization.

The shape (principally its size) of the clean beam used in the restoration step plays an important role. The clean beam is usually a fit of the main lobe (*i.e.* the inner part) of the dirty beam. This ensures that 1) the flux density estimation[3] will be correct and 2) the addition of the residual map to the convolved list of clean component makes sense (*i.e.* the unit of the clean and residual maps approximately matches).

The final addition of the residual map plays a double role. First, it is a first order correction to insufficient deconvolution. Second, it enables noise estimate on the cleaned image since the residual image should be essentially noise when the deconvolution has converged.

Super-resolution is the fact of restoring with a clean beam size smaller that the fit of the main lobe of the dirty beam. The underlying idea is to get a bit finer spatial resolution. However, it is a bad practice because it breaks the flux estimation and the usefulness of the addition of the residual maps. It is better to use robust weighting to emphasize the largest measured spatial frequencies.

[3]Some odd dirty beams may lead to incorrect flux measurements. For example, if the dirty beam has a very narrow central peak superimposed on a rather broad plateau, the volume of the Gaussian fitted to the central peak does not match that of the dirty beam, and the flux scale will be incorrect. Data-reweighting is required to cure these peculiar situations.

4.5.2 Basic CLEAN algorithms (HOGBOM, CLARK and MX)

The main difference between the different basic CLEAN algorithms is the strategy for searching the point sources.

HOGBOM The simplest strategy of the iterative search was introduced by Högbom (1974). It works as follows

1. Localization of the strongest intensity pixel in the current residual map: $\max(|I_{\rm res}|)$.
2. Add $\gamma.\max(|I_{\rm res}|)$ and its spatial position to the clean component list.
3. Convolution of $\gamma.\max(|I_{\rm res}|)$ by the dirty beam.
4. Subtract the resulting convolution from the residual map in order to clean out the side lobes associated to the localized clean component.

γ is the loop gain. It controls the convergence of the method. In theory, $0 < \gamma < 2$. $\gamma = 1$ would in principle give faster convergence, since the remaining flux at one position is $\propto (1-\gamma)^{n_{\rm comp}}$, where $n_{\rm comp}$ is the number of clean components found at this position. But, in practice, one should use $\gamma \simeq 0.1 - 0.2$, depending on sidelobe levels, source structure and dynamic range. Indeed, deviations (such as thermal noise, phase noise or calibration errors) from an ideal convolution equation force to use low gain values in order to avoid non linear amplifications of errors.

An important property of HOGBOM algorithm is that only the inner quarter of the dirty image can be properly cleaned when dirty beam and images are computed on the same spatial grid. Indeed, the subtraction of the dirty sidelobes associated to any clean component is possible only in the spatial extent of the dirty beam image. When the user defines a support (*a priori* knowledge), the cleaned region becomes even smaller than the inner quarter of the dirty map.

CLARK The most popular variant to the HOGBOM algorithm is due to Clark (1980). The iterative search for point sources involves minor and major cycles.

In minor cycles, an HOGBOM search is performed with two limitations: 1) Only the brightest pixels are considered in the above step 1, and 2) the convolution of the found point sources (step 3 above) is done with a spatially truncated dirty beam [4]. Both limitations fasten the search but may lead to difficult convergence in cases where the secondary side lobes are a large fraction (*e.g.* 40%) of the main side lobe.

In major cycles, the clean components found in the last minor cycle are removed in a single step from the residual map in the Fourier plane. The use of the Fourier transform enable to clean slightly more than the inner quarter of the map.

CLARK is faster than HOGBOM.

MX The MX (cor Cotton-Schwab, from the names of its authors) algorithm, due to Schwab (1984), is a variant of the CLARK algorithm in which the clean components are removed from the *uv* table at each major cycle. This is the most precise way of removing the found clean components because it avoids aliasing of the dirty sidelobes. A direct consequence is that this method enables to clean the largest region of the dirty map. However, this may be a relatively slow algorithm because the imaging step must be redone at each major cycle.

In IMAGER, MX is only implemented for single channel data set.

[4] It is also theoretically possible to do so with an intensity truncated beam.

4.5.3 Advanced CLEAN algorithms to deal with extended structures (SDI, MULTI and MRC)

When the spatial dynamic of the imaged source is large (*i.e.* when the ratio of the largest source structure over the synthesized resolution is large), the basic CLEAN algorithms may (rarely) turn smooth area of the source into a serie of ridges and stripes. Indeed, when the dirty beam pattern is subtracted from a smooth feature of the dirty map, the sidelobes patterns appear in the residual map. The search for the next clean component will then pick first the pixels in the sidelobes pattern amplifying this pattern. Several variants of CLEAN have been devised to solve this problem.

SDI In the CLEAN variant proposed by Steer et al. (1984), extended features (instead of point sources) around the current maximum of the residual map are selected and removed in a single step. The simplest implementation redefines the notion of minor and major cycles of the CLARK algorithm. In the minor cycles, only the selection of the clean components is done by including all the pixels in the residual map that rise above a contour set at some fraction of the current peak level. In major cycles, all those components are removed together in the Fourier plane. The SDI algorithm may be instable if the fraction used for the selection of the clean components is badly chosen.

MRC The Multi Resolution Clean (MRC, Wakker & Schwarz 1988) is the first try to introduce the notion of cleaning at different scales. MRC works on two intermediate maps (strictly speaking MRC is a double–resolution CLEAN algorithm). The first map is a smoothed version of the dirty map and the second map, called difference map, is obtain by subtraction of the smoothed map from the original dirty map. Since the measurement equation is linear, both maps can be cleaned independently (using a smoothed and a difference dirty beam, respectively). The underlying idea is that extended sources in the dirty map will look like more "point-like" with respect to the smoothed dirty beam in the smoothed map. MRC is faster than the basic CLEAN algorithms because fewer clean components are needed to reproduce an extended source feature in the smoothed map than in the original map.

MULTI The Multi-scale CLEAN algorithm (Cornwell & Holdaway (200X)) has also been designed to improve the performance of CLEAN for extended sources. It is a straightforward extension of CLEAN that models the sky brightness by the summation of blobs of emission having different size scales. It is equivalent to simultaneously deconvolve images obtained with different synthesized beams derived from the highest resolution one by convolution kernels. This algorithms works simultaneously in a range of specified scales. Multi-scale CLEAN can produce good images with a loop gain of 0.5 or even higher.

The implementation of Multi-scale CLEAN in IMAGER slightly differs from that of CASA . It is less optimized in terms of speed, but uses a better convergence scheme in which the scale chosen at each iteration is the one with best signal to noise ratio. Accordingly, it is more stable. Only 3 scales are used so far in IMAGER, with a size ratio controlled by CLEAN_SMOOTH.

4.5.4 Implementation and typical use

Deconvolution parameters are controlled by CLEAN_* variables. Progress has been made on automatic guess for Cleaning parameters. The table below presents the current naming scheme, with previous or equivalent names mentioned in parentheses, since these names were (or are still)

used by several older packages such as MAPPING, AIPS or CASA. The equivalent "old" names (mentioned in Upper case below) will remain as aliases, while those mentioned in mixed case have disappeared as they were seldom used before.

CLEAN_ARES	Absolute residual (ARES)
CLEAN_FRES	Fractional residual (FRES)
CLEAN_GAIN	Loop gain (GAIN)
CLEAN_INFLATE	Inflation factor allowed to display MultiScale clean components
CLEAN_METHOD	Cleaning Method (METHOD)
CLEAN_NCYCLE	Maximum number of Major cycles (Nmajor)
CLEAN_NITER	Maximum number of iterations (NITER)
CLEAN_NGOAL	A number of components for ALMA joint deconvolution only (Ngoal)
CLEAN_NKEEP	Number of iterations used to check convergence (see below)
CLEAN_POSITIVE	Minimum number of positive Clean components
CLEAN_RATIO	Ratio for Dual Resolution clean (Ratio)
CLEAN_RESTORE	Minimum primary beam threshold for restoring (Restore_W)
CLEAN_SEARCH	Minimum primary beam threshold for searching (Search_W)
CLEAN_SMOOTH	Smoothing factor for Multi Scale Clean (Smooth)
CLEAN_SPEEDY	Speeding factor for Clark (Spexp)
CLEAN_WORRY	"Worry" factor for Clark (Worry)

Implementation In IMAGER, the variants of the `CLEAN` algorithms discussed above are coded as the following commands: `HOGBOM`, `CLARK`, `MX`, `SDI`, `MULTI` and `MRC`. All those commands work on two internal buffers containing the dirty beam and dirty image. Both buffers are created directly from *uv* table through the `UV_MAP` command, or they can be loaded from files through the `READ BEAM` and `READ DIRTY` commands. The behavior of those commands is controlled through the following common SIC variables:

Iterative search

CLEAN_POSITIVE Number of positive clean components to be found before enabling the search for negative components. Default is 0.

CLEAN_GAIN Loop gain. Default is 0.2, good compromise between stability and speed.

Stopping criteria

CLEAN_NITER Maximum number of clean components. Default is 0.

CLEAN_FRES Maximum amplitude of the absolute value of the residual image. This maximum is expressed as a fraction of the peak intensity of the dirty image. Default value is 0.

CLEAN_ARES Maximum amplitude of the absolute value of the residual image. This maximum is expressed in the image units (Jy/Beam). Default value is 0.

CLEAN_NKEEP Minimum number of Clean components be- fore testing if Cleaning has converged. Default value is 70.

Support

BLC **and** TRC Bottom Left Corner and Top Right Corner of a square support in pixel units. Default is 0, meaning using only the inner quarter if no other support is defined.

SUPPORT A command that defines the support where to search for clean components. The support can be a Mask, or a Polygon. For a Polygon, the definition can be interactive, using the GREG cursor. This definition can be stored in a file through the WRITE SUPPORT command and read back in memory from the file with the SUPPORT command. The polygon support definition is stored in the SUPPORT% structure. For a Mask, it can be read by command READ MASK or computed by command SUPPORT /THRESHOLD, and saved by command WRITE MASK.

Clean beam parameters

MAJOR, MINOR **and** ANGLE FWHM size of the major and minor axes (in arcsec) and position angle (in degree) of the Gaussian used to restore the clean image from the clean component list. Default is all parameters at 0, meaning use the fit of the main lobe of the dirty image. Changing the default value of those parameters is dangerous.

Other variables control specific aspects of a subclass of the CLEAN algorithm:

CLEAN_NCYCLE Maximum number of major cycles in all algorithms using this notion (CLARK, MX, SDI). Default is 50.

BEAM_PATCH Size (in pixel units) of the dirty beam used to deconvolve the residual image in minor cycles. It is used in CLARK and MRC algorithms only. Default value is 0. This is for development only.

CLEAN_SMOOTH Smoothing factor between different scales in the MULTISCALE methods. Default value is sqrt(3).

CLEAN_RATIO Smoothing factor between different scales in the MRC method. Default value is 0, for which the code automatically derives the best power of 2 adequate for the current problem.

4.5.5 Typical deconvolution session

```
 1 read beam demo
 2 read dirty demo
 3 clean ?
 4 hogbom /flux 0 1
 5 show residual
 6 show clean
 7 write clean demo
 8 let name demo
 9 show noise
10 let ares 0.5*noise
11 clean ?
12 hogbom /flux 0 1
13 let niter 2000
14 clean ?
15 hogbom /flux 0 1
```

```
16 show residual
17 show clean
18 for iplane 1 to 10
19    show clean iplane
20    support
21    hogbom iplane /flux 0 1
22    write support "demo-"'iplane'
23 next iplane
24 show residual
25 view cct
26 view clean
27 write residual demo
28 write clean demo
29 write cct demo
```

Comments:

Steps 1-2 Read dirty beam and dirty image from the `demo.beam` and `demo.lmv` files. Those steps are not needed if the dirty beam and image are already stored in the internal buffer, *i.e.* if you have imaged the *uv* table just before in the same IMAGER session.

Steps 3-6 Print the current state of the control parameters, deconvolve the dirty image using the `HOGBOM` algorithm (step 3) and look at the results (residual and clean images). The `/flux 0 1` option pop-up the visualization of the cumulative flux deconvolved as the clean components are found.

Steps 8-12 Estimate the empirical noise through the `SHOW NOISE` command after this first deconvolution and set the `ares` stopping criterion accordingly. Check that the new value of `ares` has been correctly set (step 11) and restart deconvolution.

Steps 13-17 Increase the number of clean components as the previous deconvolution stopped before the residual image reached the `ares` value. Restart deconvolution and look at results.

Steps 18-23 Attempt to improve deconvolution by definition of a support per plane and deconvolve this plane accordingly. The support is stored in a file for further re-use. The deconvolution results are then displayed.

Steps 24-26 Display the residual images, visualize the cumulative flux as a function of the clean component number and visualize the clean spectra cube in an interactive way.

Steps 27-29 Write residual image, clean image and clean component list in `demo.lmv-res`, `demo.lmv-clean` and `demo.cct` files for later use.

Typical deconvolution session using other `CLEAN` algorithm would look very similar. The main difference would be the possible tuning of other control parameters. A deconvolution session using `MX` would start differently as the imaging and deconvolution are done in the same step:

```
1 read uv demo
2 mx ?
3 mx /flux 0 1
4 show residual
```

```
       5 show clean
       6 write * demo
%       6 write beam demo
%       7 write dirty demo
%       8 write clean demo
%       9 write residual demo
%      10 write cct demo
```

Comments:

Step 1 Read the `demo.uvt` *uv* table in an internal buffer.

Step 2 Check current state of the variables that control the imaging and deconvolution.

Steps 3-5 Deconvolve and look at the results.

Steps 6-10 Write all the internal buffers on disk files.

All the tuning of the typical imaging and deconvolution sessions could be used in this `MX` session although they are not repeated here.

4.6 Practical advices

4.6.1 Comparison of deconvolution algorithms

`HOGBOM` is the basic `CLEAN` algorithm. It is robust but slow. `CLARK` introduces a faster strategy for the search and removal of clean component. However, it can be instable when dirty sidelobes are high. `MX` cleans the largest region of the dirty map because the source removal happens in the *uv* plane. However, it is slower than `CLARK` because of the repeated imaging step and it shares some of the `CLARK` instabilities because it uses the same search strategy.

`HOGBOM`, `CLARK` and `MX` may introduce artifacts as parallel stripes in the clean map when dealing with smooth, extended structures. `SDI`, `MRC` and `MULTI` introduce (in principle) a better handling of those extended sources. `SDI` is a rough attempt while `MRC` and `MULTI` introduce the notion of cleaning at different spatial scales.

4.6.2 A few (obvious) practical recommendations

Map size Make an image about twice the size of the primary beam (*e.g.* $2 \times 55''$ at 90 GHz and $2 \times 22''$ at 230 GHz for NOEMA antenna) to ensure that all the area of the primary beam (inner quarter of the dirty map) will be cleaned whatever the deconvolution algorithm is used. However, avoid making a too large dirty image because the `CLEAN` algorithms will then try to deconvolve region outside the primary beam area where the noise dominates.

Support Start your first deconvolution *without* any support to avoid biasing your clean image. If the source is spatially bound, you can define a support around the source and restart the deconvolution with this *a priori* information. Be careful to check that there is no low signal-to-noise extended structure that could contain a large fraction of the source flux outside your support... Avoid defining a support too close to the natural edges of your source. Indeed, deconvolving noisy regions around your source is advisable because it ensures that you do not bias your deconvolution too much.

Stopping criterion Choose the right stopping criterion.
Use the stability CLEAN_NKEEP parameter preferentially, i.e. keep CLEAN_ARES, CLEAN_FRES and CLEAN_NITER to zero. If it does not work, then

- Estimate an empirical noise on your first deconvolved cleaned image with STATISTIC, CLEAN, or SHOW NOISE.
- If this empirical noise value is larger than the value computed from the visibility weights (this noise value is one of the outputs of the UV_MAP command), your observation is probably dynamics limited, *i.e.* you have a bright source whose dirty sidelobes are much larger that the thermal noise. In this case, set ARES to 0 and FRES to a fraction which depends on the sidelobe level of your dirty beam.
- Else you are in the noise limited case. Set FRES to 0 and ARES to a fraction of the empirical noise value (typically 0.5).

Convergence checks Ensure that your deconvolution converged by checking that:

- The cumulative flux as a function of the number of clean component has reached a roughly constant level (use /FLUX option of the deconvolution commands to see this curves, or SHOW CCT).
- The residuals are similar or smaller in the source region (where Clean components were found) compared to elsewhere.

If not, change the values of the stopping criterion, in particular the number of clean components (CLEAN_NITER).

Deconvolution methods If you want a robust result in all cases, start with HOGBOM. If you prefer obtaining a quick result, use CLARK but you then first need to check that the dirty sidelobes are not too large on the dirty beam. If you obtain stripes in your clean image:

- First check that your deconvolution converged.
- Then check that there is no spurious visibilities that should be flagged : use command UV_FLAG as a last resort.
- If it is clear that you have an extended source structure, you should first ask yourself whether you are in the wide-field imaging case and act accordingly (see next chapter). Else you can try a CLEAN variant which better deals with cases that implies a large spatial dynamic. This is rare at NOEMA, but may happen with ALMA.

Outside help Always consult an expert until you become one.

5 Wide-field imaging and deconvolution

We are often asked why the wide-field imaging and deconvolution steps are more difficult than their equivalent for single-field. The main answer is that doing wide-field observations with an interferometer is kind of paradoxical. Indeed, (sub)millimeter interferometers are before all tuned to get the best possible spatial resolution. A natural consequence is the lack of measurement of the low spatial frequencies which are extremely important in wide-field observations. Hence the paradox.

Progress in the design (ALMA was designed with wide-field imaging as a main goal) or in performances (NOEMA and the 30-m) has led to wide-field images being now customary. The tools have become much simpler and user-friendly (see below !) but because of its paradoxical nature, wide-field imaging with an interferometer implies a knowledgeable use of those tools.

5.1 In a nutshell

```
 1  read uv gag_demo:demo-mosaic-v22.uvt
(2)  read single gag_demo:demo-single-v22.tab
(3)  uv_short
 4  uv_map
 5  clean
 6  write * MyDemo
```

1. Read your *uv* data
2. Optionally, read your single-dish data
3. Optionally, merge it with with the *uv* data set, by using the single-dish data to provide short spacings for the interferometer data.
4. Image as usual
5. deconvolve as usual
6. Save the result.

You are done. If the *uv* data set is a **Mosaic** data set, it works as if it is a single-field. But, now, if the result is crazy, do not blame the software. Rather read carefully the information below: **Mosaics** and `UV_SHORT` are simple to use, but can be tricky to use well !...

5.2 General considerations about wide-field imaging

The measurement equation for a millimeter interferometer is to a good approximation (after calibration)

$$V(u,v) = \mathrm{FT}\left\{B_{\mathrm{primary}}.I_{\mathrm{source}}\right\}(u,v) + N \tag{7}$$

where $\mathrm{FT}\left\{F\right\}(f)$ is the bi-dimensional Fourier transform of the function F taken at the spatial frequency f, I_{source} the sky intensity, B_{primary} the primary beam of the interferometer (*i.e.* a Gaussian of FWHM the natural resolution of the single-dish antenna composing the interferometer), N some thermal noise and $V(u,v)$ the calibrated visibility at the spatial frequency u,v. The product of the sky intensity by the primary beam, which "quickly" decreases to zero, implies that an interferometer looking at a particular direction of the sky will have its field-of-view limited by the size of the primary beam.

To image a field-of-view larger than the primary beam size, the antennae of an interferometer will be successively pointed in different directions of the sky typically separated by half the size of the primary beam. This process is called mosaicing and the result requires specific imaging and deconvolution steps. Another possibility is to acquire data as the interferometer antenna continuously slew through a portion of the sky. This second observing mode is called interferometric On-The-Fly (OTF). While mosaicing is standard at NOEAM (see section 5.3), some efforts are currently done to commission the OTF observing mode.

Mosaicing and OTF clearly belongs to wide-field imaging. However considerations about wide-field imaging start as soon as the size of the source is larger than about 1/3 to 1/2 of the interferometer primary beam. Indeed, a multiplicative interferometer (*e.g.* all interferometer in the (sub)mm range) is a bandpass instrument, *i.e.* it filters not only the large spatial frequencies (this is the effect of the finite resolution of the instrument) but also the small spatial frequencies (all the frequencies smaller than typically the diameter of the interferometer antennas). An important consequence is that a multiplicative interferometer do *not* measure the total flux of the observed source. This derives immediately from the following property of the Fourier Transform: The Fourier transform of a function evaluated at zero spacial frequency is equal to the integral of your function. Adapting this to our notation, this gives

$$V(u=0,v=0) \overset{\mathrm{FT}}{\rightleftharpoons} \sum_{ij \,\in\, \mathrm{image}} \{B_{\mathrm{primary}}.I_{\mathrm{source}}\}_{ij}\,. \tag{8}$$

i.e. the visibility at the center of the uv plane is the total intensity of the source. As a multiplicative interferometer filters out in particular $V(u=0,v=0)$, the information about the total flux of the observed source is lost. In summary, a multiplicative interferometer only gives information about the way the flux of the source is distributed in the spatial frequencies larger than the primary beam but no information about the total flux.

Deconvolution algorithms use, in one way or another, the information of the flux at the smallest *measured* spatial frequencies to extrapolate the total flux of the source. This works correctly when the size of the source is small compared to the primary beam of the interferometer. The extreme case is a point source at the phase center for which the amplitude of all the visibilities is constant and equal to the total flux of the source: Extrapolation is then exact. However, the larger the size of the source, the worst the extrapolation, which then underestimates the total source flux. This is the well-known problem of the missing flux that observers sometimes note when comparing the flux of the source delivered by a mm interferometer with the flux observed with a single-dish antenna. The transition between right and wrong extrapolation is not well documented. It depends on the repartition of the flux with spatial frequencies but also of the signal-to-noise ratio of the measured spatial frequencies. It is often agreed that the transition happens for sizes between 1/3 and 1/2 of the interferometer primary beam. For larger source size, information from a single-dish telescope is needed to fill in the missing information and to thus obtain a correct result. This is the object of section 6.

5.3 Mosaicing

5.3.1 Observations and processing

In a single-field observation, an interferometer tracks a particular direction of the sky, named the phase center. The portion of the sky which can be image around this direction is directly linked to the size of the primary beam. The easiest way to image field-of-view larger than the primary beam size is to track one direction of the sky after another until the desired field-of-view is filled

with small images made around many different tracking directions. This observing mode is called mosaicing and the tracked observations which constitute the mosaic are called fields.

There are many constraints to optimize mosaicing.

Nyquist sampling of the mosaic field-of-view and mosaic pattern The mosaic field-of-view must at least be Nyquist-sampled to obtain a reliable image. Each observed field can produce a reliable image of the same shape than the primary beam, *i.e.* a circular Gaussian (This assumes that the short-spacing problem has been solved). Nyquist sampling thus implies that the mosaic fields follow an hexagonal compact pattern as this ensures a distance between all neighboring fields of half the primary beam size. When the total observing time is fixed, Nyquist sampling is the best compromise between sensitivity and total field-of-view. Indeed, the distance between neighboring fields could be less (in which case the mosaic would be oversampled) than half the primary beam size. In this case, the sensitivity on each pixel of the final image would increase with the share of the time spent to observe this direction.

Uniform imaging properties and quick loop around the fields Getting uniform imaging properties is a desirable feature in the final result. This implies that a uv coverage and a noise level as uniform as possible among the different fields. Quickly looping around the different fields is the easiest way to reach this goal. However, dead time to travel from one field to another must almost be minimized. At NOEMA, the compromise is to pause at least 1 minute on each field and to try to loop over all the fields between two calibrations every 20 minutes. Hence, mosaic done in a single observing run is made of at most 20 fields. Larger mosaic must be observed by group of fields in different observing runs.

5.3.2 Imaging

When combining together (dirty or clean) images, it is important to correct the primary beam attenuation to avoid modulation of the signal in the combined image. If we forget for the moment the dirty beam convolution, the images associated to each fields are noisy measurement of the same quantity (the sky brightness distribution) weighted by the primary beam. The best estimation of the measured quantity is thus given by the least mean square formula

$$M(\alpha,\delta) = \frac{\sum_i \dfrac{B_i(\alpha,\delta)}{\sigma_i^2} F_i(\alpha,\delta)}{\sum_i \dfrac{B_i(\alpha,\delta)^2}{\sigma_i^2}}, \tag{9}$$

where $M(\alpha,\delta)$ is the brightness of the dirty/cleaned mosaic image in the direction (α,δ), B_i are the response functions of the primary antenna beams in the tracking direction of field i, F_i are the brightness distributions of the individual dirty/cleaned maps, and σ_i are the corresponding noise values. As may be seen on this equation, the intensity distribution of the mosaic is corrected for primary beam attenuation. This implies that noise is inhomogeneous. Indeed, if $N(\alpha,\delta)$ is the noise distribution and $\sigma(\alpha,\beta)$ is its standard deviation in the direction (α,β), we have

$$N(\alpha,\delta) = \frac{\sum_i \frac{B_i(\alpha,\delta)}{\sigma_i^2} N_i(\alpha,\delta)}{\sum_i \frac{B_i(\alpha,\delta)^2}{\sigma_i^2}}, \tag{10}$$

and

$$\sigma(\alpha,\delta) = \frac{\sqrt{\sum_i \frac{B_i(\alpha,\delta)}{\sigma_i^2}}}{\sum_i \frac{B_i(\alpha,\delta)^2}{\sigma_i^2}} = \frac{1}{\sqrt{\sum_i \frac{B_i(\alpha,\delta)^2}{\sigma_i^2}}} \tag{11}$$

Not only, the noise strongly increases near the edges of the mosaic field-of-view. But also, the center of each field is contaminated by increased noise level coming from the external regions of the neighboring fields. Indeed, the noise corrected for the primary beam attenuation is largely increasing where the primary beam is going to zero. To limit these effects, both the primary beams used in the above formula and the resulting mosaic are truncated.

5.3.3 Deconvolution

Standard `CLEAN` algorithms must be slightly modified to work on a dirty mosaics. Indeed, the use of truncated primary beam in the above equations is only a first order measure to avoid noise artifacts. However, the noise level still increases at the edges of the mosaic, implying that at some point the `CLEAN` algorithms will confuse noise peaks at the mosaic edges with true signal. To avoid this, the iterative search is made on a signal-to-noise image $M(\alpha,\beta)/\sigma(\alpha,\beta)$ instead of the residual image. At the restoration step, the clean component list is used to produce the residual map and clean map. Nothing particular is done with the remaining signal-to-noise image.

5.3.4 Typical use

The processing of mosaics for NOEMA is essentially similar to that of single fields. There are only two small changes

Creation of *uv* table A mosaic UV table should be created using the /MOSAIC option of command TABLE in CLIC.

Imaging is done through `UV_MAP` as for a single field. However, the process is different. The command takes into account the various fields, the primary beams, and select an optimum projection center (phase center). The later may need to be specified by the user using the `MAP_CENTER` string, or as argument to `UV_MAP`

Deconvolution is also similar, but not all algorithms are available. Only HOGBOM and CLARK are possible so far. The change of behavior of the `CLEAN` algorithms is visualized through the change of prompt from `IMAGER>` to `MOSAIC`. Two additional variables are used for mosaic deconvolution

`CLEAN_SEARCH` The minimum fraction of a primary beam below which no Clean component is searched for.

`CLEAN_RESTORE` The minimum fraction of a primary beam below which no image restoration is performed. Below this threshold, the Clean image is blanked.

Finally, note that the mosaic deconvolution produces sky brightness images while single-field deconvolution produces images attenuated by the primary beam.

Although IMAGER makes no specific assumption about the *uv* coverage of individual fields, mosaicing deconvolution will work better if all fields are equivalent in *uv* coverage and noise level [5]

[5] IS THAT TRUE ? : An additional subtlety of the current IMAGER implementation of mosaicing is that MAPPING assumes that the individual fields of the mosaic have similar noise level.

A mosaicing session would thus just be like a single-field imaging:

```
1 read uv gag_demo:demo-mosaic
2 uv_map
3 hogbom /flux 0 10
4 show residual
5 show clean
6 write * demo
```

Comments:

Step 1 Read the UV table

Step 2 Image the mosaic

Step 3 Deconvolve

Steps 4-5 Look at the result

Steps 6 Save the result

6 Short and Zero spacings

6.1 In a nutshell

```
1  read uv gag_demo:demo-mosaic-v22.uvt
2  read single gag_demo:demo-single-v22.tab
3  uv_short
4  uv_map; clean
5  view clean
6  write * MyDemo
```

1. Read your *uv* data
2. read your single-dish data
3. Merge it with with the *uv* data set,
4. Image and deconvolve as usual
5. check the result
6. Save it.

You are done. This works for mosaics as well as single fields. But, again, if the result are crazy, do not blame the software. Rather read carefully the information below: `UV_SHORT` is simple to use, but can be tricky to use well !...

6.2 Principle

Let's note D the diameter of the single-dish antenna (D = 30m for the IRAM-30m telescope) used to produce the short-spacing information and d the diameter of the interferometer antennas (d = 15m for PdBI). We already mentioned that a multiplicative interferometer filters out all the spatial frequencies smaller than $\sim d$ meters. When this information is needed to get reliable results, the source should also be observed with a single-dish antenna to produce the missing information. The single-dish antenna furnishes information about all spatial frequencies

up to $\sim D$ meters (but this information is weighted by the single-dish beam shape, *i.e.* high frequencies are measured with a worse signal-to-noise ratio than low frequencies). To recover all the information at spatial frequencies smaller than $\sim d$ meters, the diameter of single-dish antenna must be larger or equal to the diameter of the interferometer antennae: $D \geq d$.

6.3 Algorithms to merge single-dish and interferometer information

The measurement equations of a single-dish and an interferometer are quite different from each other. Indeed, the measurement equation of a single-dish antenna is

$$I^{\rm sd}_{\rm meas} = B_{\rm sd} \star I_{\rm source} + N, \tag{12}$$

i.e. the measured intensity ($I^{\rm sd}_{\rm meas}$) is the convolution of the source intensity distribution ($I_{\rm source}$) by the single-dish beam ($B_{\rm sd}$) plus some thermal noise, while the measurement equation of an interferometer can be rewritten as

$$I^{\rm int}_{\rm meas} = B_{\rm dirty} \star \{B_{\rm primary}.I_{\rm source}\} + N, \tag{13}$$

i.e. the measured intensity ($I^{\rm int}_{\rm meas}$) is the convolution of the source intensity distribution times the primary beam ($B_{\rm primary}.I_{\rm source}$) by the dirty beam ($B_{\rm dirty}$) plus some thermal noise. $B_{\rm sd}$ has very similar properties than $B_{\rm primary}$ and very different properties than $B_{\rm dirty}$. In radioastronomy, $B_{\rm sd}$ and $B_{\rm primary}$ both have Gaussian shapes. Moreover, the fact that we will use the single-dish information to produce the short-spacing information filtered out by the interferometer implies that $B_{\rm sd}$ and $B_{\rm primary}$ have similar full width at half maximum. Now, $B_{\rm dirty}$ is quite far from a Gaussian shape with the current generation of interferometer (in particular, it has large sidelobes) and the primary side lobe of $B_{\rm dirty}$ has a full width at half maximum close to the interferometer resolution, *i.e.* much smaller than the FWHM of $B_{\rm sd}$.

Merging both kinds of information obtained from such different measurement equations thus asks for a dedicated processing. There are mainly two families of short-spacing processing: The hybridization and the pseudo-visibility techniques.

6.3.1 Hybridization technique

In this family, most of the processing is done on the interferometric data alone. Indeed, the interferometric data is deconvolved and corrected for the primary beam contribution to obtain

$$I^{\rm int}_{\rm clean} = B_{\rm clean} \star I_{\rm source} + N', \tag{14}$$

where $B_{\rm clean}$ is a Gaussian of FWHM equal to the interferometer resolution and N' is some thermal noise corrected for the primary beam contribution. Two main facts are hidden in this formulation: 1) the field-of-view of the observation is obviously limited to the observed portion of the sky and 2) more importantly, the lack of short-spacings has not yet been overcome and a better formulation would be

$$I^{\rm int}_{\rm clean} = \text{Highpass-filter}\,\{B_{\rm clean} \star I_{\rm source}\} + N'. \tag{15}$$

The simplicity of equation 14 is thus slightly misleading but we will keep it for the sake of simplicity. The hybridization method consists in combining two images ($I^{\rm sd}_{\rm meas}$ and $I^{\rm int}_{\rm clean}$) in the *uv* plane.

1. Both images are first spatially regridded on the same fine grid.

2. The FFT of those two images are computed, and linearly combined by selecting the low spatial frequencies from FFT($I^{\rm sd}_{\rm meas}$) and the high spatial frequencies from FFT($I^{\rm int}_{\rm clean}$). The transition between low and high spatial frequency

3. The result is FFTed back to the image plane to produce a final,unique image, which takes into account both single-dish and interferometric information.

The method has the following free parameters: the transition radius and the detailed shape of that transition. To avoid discontinuity, the transition shape is chosen to be reasonably smooth. The spatial frequency of transition is generally chosen to the smallest spatial frequency reliably measured by the interferometer (*e.g.* about 20 m for NOEMA).

6.3.2 Pseudo-visibility technique

General description In this family, the single-dish information is heavily processed before merging with the interferometric information. The basic idea is to produce from the single-dish observations pseudo-visibilities similar to the ones that would be produced by the interferometer if they were not filtered out.

1. The Single-Dish measurements are re-gridded and then FFTed into the uv plane.
2. The data are deconvolved of the single-dish beam ($B_{\rm sd}$) convolution by division by its Fourier Transform (truncated to the antenna diameter).
3. The data are FFTed back to the image plane and multiplied by the interferometer primary beam, $B_{\rm primary}$.
4. The result is FFTed again in the uv plane where the visibilities are sampled on a regular grid.
5. In the case of a mosaic, the two last operations are performed for each pointing center.

Using the properties of the Fourier transform, we can rewrite the measurement equation of an interferometer as

$$V(u,v) = \{\mathrm{FT}(B_{\rm primary}) \star \mathrm{FT}(I_{\rm source})\}\,(u,v) + N. \tag{16}$$

This equation means that the visibility measured by an interferometer at the spatial frequency (u,v) is the convolution of the Fourier transform of the source intensity distribution by the Fourier transform of the primary beam. Hence, to get pseudo-visibilities truly consistent with interferometric visibilities, we must be able to reliably compute the convolution by the Fourier transform of the primary beam. This implies that we can compute pseudo-visibilities only for spatial frequencies lower than $D - d$. The use of the IRAM-30m to produce the short-spacing information of the PdBI is thus ideal as it enables to recover pseudo-visibilities up to 15 m ($= 30\,\mathrm{m} - 15\,\mathrm{m}$). Once the pseudo-visibilities have been computed, they are merged with the interferometric visibilities and standard imaging and deconvolution are then applied to the merged data set.

Single-dish vs interferometer weight In all cases involving short spacings, the relative weight of the single dish data to interferometer data is critical. Within the restrictions imposed by the noise level, this relative weight is a free parameter. It is all the more important that the Fourier transform of the uv plane density of weights is the dirty beam, a key parameter of the deconvolution. The general goal is to have a dirty beam as close as possible to a Gaussian. As the Fourier transform of a Gaussian is a Gaussian, we search for obtaining a uv plane density

of weights as close as possible to a Gaussian. In general, the short spacing frequencies are small compared to the largest spatial frequency measured by an interferometer. This implies we can use the linear approximation of a Gaussian, *i.e.* a constant, in the range of frequencies used for the short spacing processing. We thus end up with the need to match (as far as possible) the Single-Dish and interferometric densities of weights in the uv plane. In practice, we compute the density of weights from the single-dish in a uv circle of radius $1.25d$ and we match it to the averaged density of weights from the interferometer in a uv ring between 1.25 and $2.5d$. Experience shows that this gives the right order of magnitude for the relative weight and that a large range of relative weight around this value gives very similar final results.

When processing IRAM-30m data to combine to NOEMA data, this criterion implies a large down-weighting of the IRAM-30m data which may make think that too much observing time was used at the IRAM-30m data. However, just using the above criterion to determine the observing time needed at the IRAM-30m would in general lead to very low signal-to-noise ratio of the single-dish map. In such a case, it is very difficult to detect problems which would translate in strong artifacts in the IRAM-30m + NOEMA combination. We recommend to ask for enough IRAM-30m time to get a *median* signal-to-noise ratio of 5 on the single-dish map. This ratio should be achieved for all velocity channels of interest (which may include the line wings).

6.3.3 Comparison

The simplicity of the hybridization technique is its main advantage. It is simple to understand and simple to implement. However, this method works badly in practice because it is truly difficult to obtain a reliable deconvolution of interferometric data alone when short-spacing information is important. An interferometer is a spatial pass-band filter, filtering in particular the zero spacing. This implies that the total flux in the dirty image is zero (*i.e.* as much negative as positive flux in the dirty image) but that the dirty beam integral is also zero (*i.e.* as much negative as positive sidelobes). Adding the short-spacing information (and in particular the zero spacing) through the pseudo-visibility method, we enforces the positivity of the dirty image total flux and of the dirty beam integral. It is well-known that trying to deconvolve a mosaic built only with interferometric data is quite difficult. It almost always requires the definition of support where the `CLEAN` algorithms can search for clean components with the clear risk to bias the final result. In contrast, adding the short-spacing information through pseudo-visibilities enables an almost straightforward `CLEAN` deconvolution *without* the need of any support.

For the sake of illustration, let us assume an intensity distribution made of a large scale structure (*e.g.* a smoothly varying intensity) superimposed with a small scale distribution both in emission and absorption. An interferometer will filter out the smooth distribution. If there is no additional zero spacing information, the smooth distribution is completely lost with the important consequence that the final deconvoled image will have positive (emission) and negative (absorption) structures. Trying to reproduce both negative and positive structures is one of the most difficult task for deconvolution algorithms. In addition, the presence of large negative structures create instabilities in the algorithms of the `CLEAN` family (because it is difficult to distinguish between negative absorption structures and negative sidelobes of emission structures). Only the definition of support around positive emission peaks may succeed to stabilize the `CLEAN` algorithms with the drawback of biasing the result.

Both kind of algorithms (hybridization and pseudo-visibility) are implemented in GILDAS. However, we strongly recommend to use the pseudo-visibility algorithm. That is why only the pseudo-visibility method is packaged in a user-friendly way, in the `UV_SHORT` command. Pety et al. (2001a,b,c) showed through simulations that 1) the pseudo-visibility algorithm implemented in

GILDAS enable extremely reliable results (fidelities of a few thousands) on ideal observations and 2) the accuracy of the wide-field imaging is limited by pointing errors, amplitude calibration errors and atmospheric phase noise (and not by the used algorithms), even for ALMA.

6.4 Hybridization technique and ALMA

A special case where Hybridization can be extremely useful is that of ALMA observations involving the main 12-m array, the ACA and single-dish data with the 12-m antennas. In some circumstances, an optimal result is obtained by hybridizing Cleaned images produced from the mosaics obtained by combining (using the pseudo-visibility method) ACA and Single-dish data as short spacings, with another mosaic obtained with the 12-m data and the Single-dish data together.

The deconvolution of each mosaic is stabilized by the addition of the 12-m single-dish zero or short spacings, and these good mosaics are then merged in an optimal way by using the best region of the *uv* plane that they sample.

6.5 The Zero spacing: an important subset

An important subset of the pseudo-visibility method is the production of only the zero spacing. Indeed, the zero spacing is just the total flux of the observed field-of-view. Hence, if the observed field-of-view is small enough to fit in the single-dish beam (this is in particular always the case if $D = d$), a single spectrum observed with the single-dish telescope in the direction of the interferometer phase center may be used as zero spacing, only a scaling from Kelvin to Jansky is needed. This is the poor man solution as only part of the short spacing information is recovered by this technique.

6.6 Short Spacings in practice: command UV_SHORT

Our algorithm to produce the short-spacing information is coded in the UV_SHORT command. UV_SHORT will **add** the short spacing information to the current *uv* table (read by command READ UV and optionally transformed by further UV_... processing commands).

UV_SHORT has a substantial number (17) of control variables, but with experience, they have been reduced to 5 significant ones, among which only 3 really matter in most cases but often can be used with their default values:

SHORT_SD_FACTOR The single-dish brightness unit to flux conversion factor. If set to zero, UV_SHORT will attempt to derive it from the information available in the single-dish data

SHORT_UV_TRUNC The longest baseline retained in the pseudo-visibilities. It defaults to the maximum theoretically possible, the single-dish diameter minus the interferometer diameter. Smaller values are allowed, and even recommended if the pointing quality of the single-dish data is moderate.

SHORT_SD_WEIGHT The relative weight scaling factor between the pseudo-visibilities and the interferometer visibilities.

The relative weight of these visibilities is derived by UV_SHORT in order to optimize the shape of the overall synthesized beam. SHORT_SD_WEIGHT is a scale factor to this optimum weight, which may need to differ from 1 in case of poor *uv* coverage in the interferometer data or noisy single-dish data (it should be lower than 1 in this case).

UV_SHORT ? will list these 3 major ones, and UV_SHORT ?? the 2 remaining main ones:

SHORT_TOLE The position tolerance in the single-dish map

SHORT_MIN_WEIGHT The minimum (relative) weight for a spectrum in the single-dish map to be included.

as well as four optional ones needed only if the original single-dish and *uv* data lacks the proper information (antenna diameter and beam sizes)

The UV_SHORT command starts from data in a the format produced by the CLASS command TABLE command, and read in IMAGER through command READ SINGLE. Basically, this is a GDF table containing one line per spectrum, the columns representing the lambda offset, beta offset, weight, and the spectrum intensities. [6]. This data must match spectrally the velocity sampling of the interferometric data. This can be obtained using the /RESAMPLING option of command TABLE in CLASS.

The READ SINGLE and UV_SHORT commands also support a 3-D data cube (as produced by e.g. command XY_MAP in CLASS) as input instead of a CLASS table. Again, the velocity axis must match that of the interferometric data.

temporary results as SIC variables and associated WRITE commands to save them ?

UV_SHORT will automatically produce the Zero spacing from the single-dish data when the data does not allow other short spacings to be evaluated.

Finally, as UV_SHORT **adds** the short spacing information, UV_SHORT /REMOVE allows to remove it (there is no direct "replace" possibility).

6.7 Practical considerations

6.7.1 When are short-spacing information needed?

- If the source size is smaller than 1/3 the primary beam size, short-spacing information is superfluous.
- If the source size is between 1/3 and 1/2 the primary beam size of NOEMA antennas, a single spectra obtained at the IRAM-30m telescope in the direction of the source can be used to produce the zero spacing information with the UV_ZERO task. Indeed, the IRAM-30m diameter being twice the diameter of the PdBI antenna, all the flux of the source will be measured by a single IRAM-30m spectrum only if the size of the source is smaller than 1/2 the primary beam size of PdBI antennae.
- If the source size is larger than 1/2 the primary beam size of NOEMA antennas, short-spacing information under the form of an IRAM-30m map is almost always mandatory. The only exception could be wide-field imaging of a region made of unresolved or small (compared to the primary beam size) sources as it may happen when mapping close-by external galaxies for instance. However, adding short-spacing will anyway help the deconvolution. In case of doubt, always add short-spacing information.

6.7.2 How to optimize single-dish observations?

One of the main difficulty of the short-spacing problematic is the need of observations from a single-dish telescope at least as big as the interferometer antennas[7]. In this respect, the IRAM-30m and NOEMA are very complementary. Nevertheless, when observing with the single-dish

[6] This format is subject to change: Please, refer to the TABLE documentation for up-to-date information

[7] If there were no pointing errors, a single-dish of the same size as the interferometer antennas would be strictly sufficient.

telescope, a few precautions are needed to avoid contaminating the interferometric data with possible artifacts of single-dish data.

- The field-of-view of the single-dish map must be twice the field-of-view covered by the mosaic. The only exception to this rule happens when the source intensity decreases to zero in a smaller field-of-view. Indeed, there is no point in observing an empty sky.
- The observing strategy must enforce Nyquist sampling (or better) of the source at the resolution of the single-dish telescope.
- A particular care should be taken of the pointing, tracking and amplitude calibration and baseline removal as those are critical issues in obtaining a high quality single-dish map to produce short-spacing information. For instance, data with too large tracking errors should be discarded.
- We advise to make many On-The-Fly coverages of the observed field-of-view to get homogeneous observing conditions. Scanning in perpendicular directions is needed to decrease stripping.

Sometimes, single-dish telescope time is scarce and some of the above criteria can not be fulfilled. In those cases, you can still try to use your single-dish observations and our algorithm will try to make its best to get a sensible result. However, any artifact in the combination may directly come from wrong single-dish observations. In other words, do *not* blame the software unless you are sure of the quality of the quality of your single-dish (and interferometric) observations...

7 Self calibration

7.1 In a nutshell

```
1  read uv YourData
2  selfcal phase
3  selfcal summary
4  selfcal show
5  selfcal apply
6  uv_map
7  clean
6  write * YourSelf
```

1. Read your *uv* data
2. Self-calibrate the phase
3. Get a summary of the result
4. Show the phase correction between the last 2 iterations
5. Apply the self-calibration
6. Image as usual
7. Clean as usual
8. Save the result if it is worthwhile...

You are done. But, now, if the result is crazy, do not blame the software. Rather read carefully the information below: `SELFCAL` is simple to use, but there are some pitfalls...

7.2 Principle

The self-calibration idea is based on the fact that the dominant error terms are antenna-based, while source information is baseline-based. With N antennas, one gets at any time $N(N-1)/2$ visibility measurements, but N amplitude gains, and only $N-1$ error terms for the morphology of the source (phase gains). The $N-1$ number is because only relative phases count. The absolute flux scale is a separate problem, and therefore also $N-1$ relative amplitude gains count.

The measured visibilities on baselines from antenna i to antenna j at time t are, from the simplified measurement equation:

$$V_{\rm obs}(i,j,t) = G(i,t)G^{\star}(j,t)V_{\rm true}(i,j) + Noise \tag{17}$$

where $G(i,t)$ is the complex (phase and amplitude) gain for the antenna i at time t. The true visibility $V_{\rm true}(i,j)$ only depends on the baseline (i,j), not on the time.

Given a source model $V_{mod}(i,j)$, one can derive the antenna gain products at time t, based on the system:

$$\frac{V_{\rm obs}(i,j,t)}{V_{\rm mod}(i,j)} = G(i,t)G^{\star}(j,t) \tag{18}$$

which is an over-constrained process, since there are $N(N-1)/2$ constraints for $N-1$ unknowns. Solving for this over-constrained problem is similar to deriving the amplitude and phase solution from a calibrator observation. In the calibrator case (i.e., an unresolved source like a distant

bright quasar), $V_{\rm mod(i,j)} = (1.0, 0.0)$ (constant amplitude, zero phase), so there is no risk of noise amplification in the process.

For any (not a point-like) source, $V_{\rm mod}$ must be guessed. Self-calibration will use your source to improve the calibration of the antenna-based (complex) gains as a function of time. The practice is to proceed iteratively, based on a preliminary deconvolution solution. Let $V_{\rm obs}(k)$ be the "observed" visibilities at iteration k, with $V_{\rm obs(k=0)} = V_{\rm obs}$ the raw calibrated visibilities. Some of the Clean components derived from $V_{obs}(k)$ are used to define "model' visibilities $V_{\rm mod}(k)$. Then, solving for the antenna gains, one obtains:

$$V_{\rm obs}(k+1) = \frac{V_{obs}(k)}{(G_i G_j^\star)} \tag{19}$$

The model is thus progressively refined, and in the end, satisfies better the initial constraints on the source shape and on the antenna gains as a function of time provided by the measurements. Note that the absolute phase (and hence the position) can be lost in the self-calibration process and it should not be used for absolute astrometry.

There are two types of self-calibration: phase and amplitude self-calibration. The amplitude gain is a more complex problem than the phase gain. Amplitude gains can (and often do) vary with time, but from the measurement equation, a scale factor in the amplitude gain can be exchanged by a scale factor on the source flux. It is thus customary to re-normalize the gains so that the source flux is conserved in the process. An alternate (perhaps not strictly equivalent) solution is to ensure that the time averaged product of the amplitude gains is 1. The two approaches differ by the averaging process.

For any typical source, $V_{\rm mod}$ is non zero and of magnitude smaller than 1 (using the total flux as a scale factor) since the source is partially resolved. So in computing $V_{obs}/V_{\rm mod}$, there is noise amplification. It may even be the case that V_{mod} is zero (case of an extended, over-resolved emission), and thus some (long) baselines will yield no direct constraint on the antenna gains $G(i)G^\star(j)$. But this should not matter too much for self-calibration, for two reasons. First, other (i.e., shorter) baselines may provide contraints on the gains. Second, if all $V_{\rm obs}$ for an antenna are close to zero, it implies $V_{\rm mod}$ must be close to zero too, so an error on the phase of those visibilities (as well as on its magnitude) is not so important.

Self-calibration is related to the "closure" relations. For any triplet of antennas, the phase of the triple product $V_{ij}V_{jk}V_{ki}$ is independent of the antenna errors, and thus is (within the noise) a bias free constraint on the source. Similarly, for any quadruplet of antennas, the amplitude of the ratio $(V_{ij}V_{kl})/(V_{ik}V_{jl})$ is independent of the antenna errors. But here, the noise amplification can be large because of the likelihood to have two small visibilities. For this reason, amplitude self-calibration requires in practice higher signal to noise ratios than phase calibration in the initial deconvolved data set used as a model.

Among the advantages of self-calibration, one may emphasize that antenna gains are derived at the correct time of the science object observation, while they must be interpolated in the classical calibration approach. Both atmospheric and electronic noises are supposed to vary with time, although with different timescales. Gains are also computed in the correct direction on the celestial sphere, while the calibrator-based approach introduces differences in the pointing direction with respect to the science object. The robustness of the approach increases with the number of baselines.

In order to implement self-calibration, it is however necessary that the signal to noise ratio be large enough (the process will require a sufficient bright source). Self-calibration can especially bring significant improvements to the calibration solution in the case of higher than expected

background noise, or in the presence of convolutional artifacts around objects, especially point sources.

7.3 Implementation

The typical procedure for self-calibration consists in an iterative process based on the following steps:

1. `UV_MAP /SELF + CLEAN + UV_RESTORE /SELF`: from the classically calibrated (and preliminarily flagged) data, define an initial source model through a conservative (not too deep) first deconvolution : the model consists of a reasonably modest number of CLEAN components.
2. `SOLVE`: determine an estimate of the antenna gains (best fit to the observed visibilities)
3. `APPLY`: apply the derived gains to correct the observed data
4. `STATISTIC`: compare to the initial Image, and estimate the improvement through an adequate quality assessment (e.g., improvement of the dynamic range)
5. If necessary, re-build a new model from these corrected data, and iterate until the solution is satisfactory.

Phase-only self-calibration is less stringent on the signal-to-noise ratio (SNR) threshold than amplitude self-calibration, and it should therefore be attempted first. For phase self-calibration to work, the SNR values in the initial data should be at least of $SNR > 3$ per antenna (in a solution interval shorter than the time for significant phase variations for all baselines to a single antenna). The SNR threshold in the initial image depends on the number of antennas and on the adopted time averaging. Depending on the complexity of the source (and the contribution from extended emission), all available baselines may not be considered in the process, and specific preliminary flagging could be necessary. Amplitude errors tend to be negligible for dynamic ranges below about 500. Amplitude self-calibration will thus be eventually attempted in a subsequent step.

Self Calibration is available in IMAGER through the command `SELFCAL`, which uses an iterative scheme driven by a script (gag_pro:p_selfcal.ima). From the above principle, the script controls a) the number of iterations b) the number of selected clean components at each iteration c) the time scale of the solution, i.e. the integration time over which the gains are assumed to be constant

The parameters of the command `SELFCAL` are available as `SELF_Names` SIC variables. The script uses the commands `UV_MAP`, `CLEAN`, and `SOLVE` and `APPLY` from the `CALIBRATE\` language. By default, a solution is searched for the phase calibration only (`SELF_MODE` = `PHASE`), and the number of iterations is 3 (`SELF_LOOP`). For each iteration:

- All components found by `CLEAN` are kept by default (`SELF_NITER`= 0), but for simple source structures, 10 components only may be enough (the maximum number `NITER` of CLEAN components to subtract is automatically guessed by the program in the default process, see `CLEAN_NITER` and other usual clean convergence control variables),
- the minimum flux density per pixel to be considered by CLEAN can be defined (SELF_MINFLUX = 0)
- The "integration" times (gain averaging) are fixed to a default value SELF_TIMES= 45 s (minimum value for NOEMA, while it is only 6 s for ALMA). This can be adapted for each loop (in general, one should start with larger solution times, depending on the SNR values and try to decrease it in order to better sample the atmospheric fluctuations).

The number of iterations can be changed by resizing the SELF_TIMES, SELF_NITER, or SELF_MINFLUX arrays (the number of loops SELF_LOOP is then automatically recomputed). You should make sure that these 3 arrays have the same dimension. If any of the SELF_NITER and/or SELF_MINFLUX array are constant then their dimension is accordingly changed by SELFCAL each time one of these arrays is resized. For instance, the following command allows to define 5 loops with an integration time for solution of 45 sec:

```
let Self_times 45 45 45 45 45 /resize
```

SELF_NITER and SELF_MINFLUX are automatically enlarged to the same size if, and only if, they were already constant. By default, all channels are averaged to compute a "continuum" image, but the range of adequate channels can be specified through SELF_CHANNEL.

SELFCAL PHASE will compute a phase calibration, but will not apply it. One needs to call SELFCAL once more with the argument APPLY in order to apply the solution (the script does really apply the solution if and only if the previously found solution can be considered as a good one (see SELF_STATUS argument value).

SELFCAL APPLY automatically saves the parameters and results in the *selfcal.last* file. By default, data are flagged if no sufficiently good solution is found. SELFCAL APPLY keep tracks of whether the solution has already been applied through SELF_APPLIED. SELFCAL APPLY will refuse to apply "bad" solutions: solutions are declared "bad" if the improvement in dynamic range and noise level is insufficient (i.e. below a precision level controlled by SELF_PRECISION). In this case, the SELF_STATUS variable is negative. In this case, the user can still decide to apply the solution directly using command APPLY, but the SELF_APPLIED variable will not be updated.

The merit criteria for the quality assessment of the computed solution are the final dynamic range, and the Clean map noise at each iteration. These quantities are stored in the SELF_DYNAMIC and SELF_RMSCLEAN variables (at each iteration). The dynamic is defined as the ratio of the peak flux density value to the noise in the clean map (automatically estimated with the command STATISTIC). The minimum signal to noise ratio value (for an antenna) for a valid solution is SELF_SNR= 6 by default.

SELFCAL can be controlled through a widget, using command SELFCAL /WIDGET (see Figure 1)

It is possible to visualize the computed corrections with the command SELFCAL SHOW. The solution computed with the SOLVE command is written in a *'self_sname'.tab* file. By default, the difference between the last two iterations is displayed. For PHASE, the phase difference should be close to zero if the solution converged, and for AMPLI the amplitude values close to 1. It is also possible to show the difference between two specified iterations. In addition, the command SELFCAL SUMMARY will display the results of the process in terms of resulting noise and improved dynamic range. If the solution is satisfactory, the command SELFCAL SAVE can be used to save both the results and the parameters in the *selfcal.last* file. (SELFCAL APPLY performs an implicit SELFCAL SAVE.)

7.4 Basic use

7.4.1 Timescale for averaging solution interval

The solution interval, or timescale used to average the solution for the gain variations, is the result of a tradeoff between the timescale of the true gain variations (e.g., changes in the atmospheric conditions or electronics) and the data averaging which is necessary to reach a minimum SNR value for the visibilities. In principle, this timescale should be smaller than the coherence time

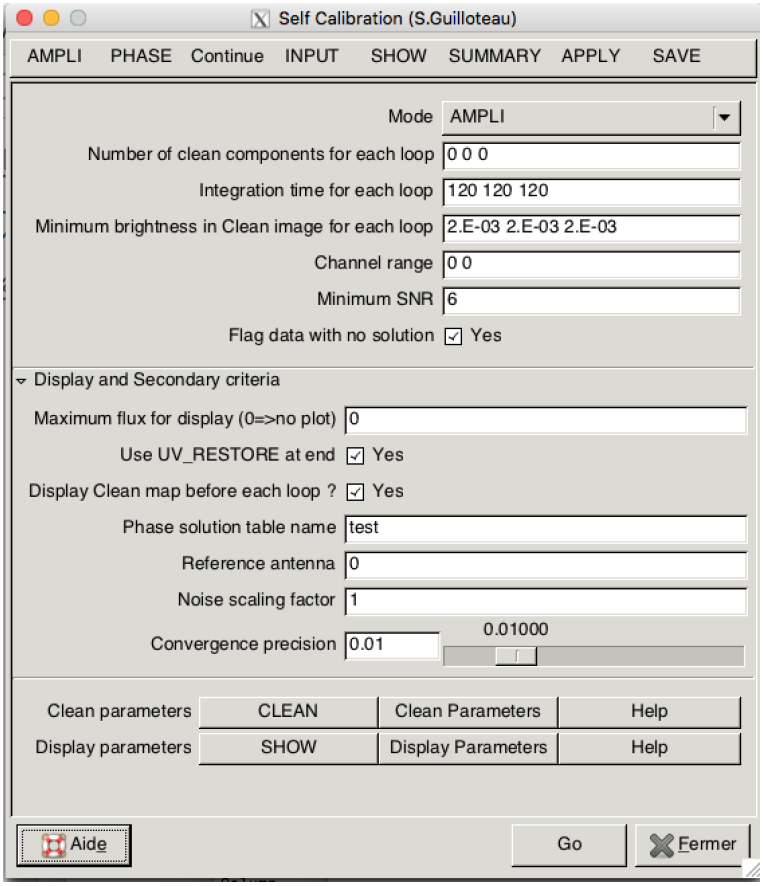

Figure 1: The Self-Calibration widget

of the atmospheric fluctuations for the phase solution, typically one minute. It is in general longer for the amplitude gains, since here these are changes in the atmospheric transparency or antenna gains which matter. It is therefore recommended to start iterations with a not too short averaging time, and to decrease it in a second step if the self-calibration was successful. It is also recommended to use the same integration time for the last two iterations, in order to ease the interpretation of the results and of the `SELFCAL SHOW` display.

The current self-calibration method computes baseline-based gains (observed complex visibility divided by model prediction) for each visibility, and performs the time averaging on these baseline-based gains. Antenna based gains are derived from the time-averaged baseline-based gains. This is in principle an optimal method, since the model is not noisy: the linearity of the first step guarantees a noise decrease as square root of time.

7.4.2 Quality assessment and data flagging

Validation of the solution One of the difficulties of self-calibration is to evaluate whether it has improved the image or not. The self-calibration solution is biased towards the assumed model. If used with insufficient signal to noise, it will tend to produce a point source at the initial peak position, and the bias will be of order of the noise. This may be inappropriate. Currently, the validity of the self-calibration solution is based on the estimated signal to noise ratio for the gains at each time step. If that SNR is below a user-controlled threshold (by default, `SELF_SNR`= 6), the corresponding data is flagged (default value: `SELF_FLAG`=`YES`) or kept WITHOUT self calibration (if `SELF_FLAG`=`NO`).

Flagging or not flagging ? The decision to flag or not results from a trade-off:

- `SELF_FLAG` = `Yes` : will result in no contamination by bad data, but may lead to lower angular resolution since long baselines may be flagged.
- `SELF_FLAG` = `No`: will avoid loosing all long baselines (where the SNR is lower)

Both options may be explored, and it is recommended to check afterwards the final angular resolution with and without flagging.

The following scheme is proposed to check the validity of the self-calibration solution:

- read the status using `SELFCAL SUMMARY`
- use `SELFCAL SHOW` to verify if the solution is converged
- if it looks good, but noise is still far from theoretical, try again to self-calibrate with a shorter integration time (SELF_TIMES).
- if it is not good, try to increase `SELF_TIMES` and find an optimum value. For ALMA data, typical values may be in the range $6 - 60$ s, and for NOEMA in the range $45 - 120$ s. Alternatively, you can also try to decrease `SELF_SNR` to lower values, but never less than 3.

From our experience, the number of loops `SELF_NLOOP` does not impact much the quality of the solution, and 2 to 3 iterations are usually sufficient.

Caveat: the comparison with theoretical noise relies on a proper scaling of the weights of the UV data. This was fine for the previous IRAM correlator, but data exported from CASA is not always correct in this respect. It also remains to be checked with the PolyFIX NOEMA correlator. `UV_PREVIEW` can warn you about potential issues in this respect. The appropriate scaling factor can be specified by variable `SELF_SNOISE`.

7.4.3 Advanced use

Amplitude self calibration The amplitude calibration (`SELFCAL AMPLI`) is a secondary step in the self-calibration process. In general, it should only be attempted if the phase calibration was already excellent, i.e., once you obtained the best solution by adjusting the `SELF_TIMES` parameter for the `SELFCAL PHASE` command. If possible or needed, the amplitude self-calibration should use a *longer timescale* than the phase calibration (typically, `SELF_TIMES`= 120s). `SELFCAL` automatically adjusts the gains so that their mean is 1, in order to avoid changing the flux scale. In practice, it is useless if the expected noise limited dynamic range is less than about 300.

Support restriction, flux threshold Support restriction in the `CLEAN` process may be useful to build a simpler model for very complex, extended sources only. Similarly, limiting the flux per pixel in the model may help.

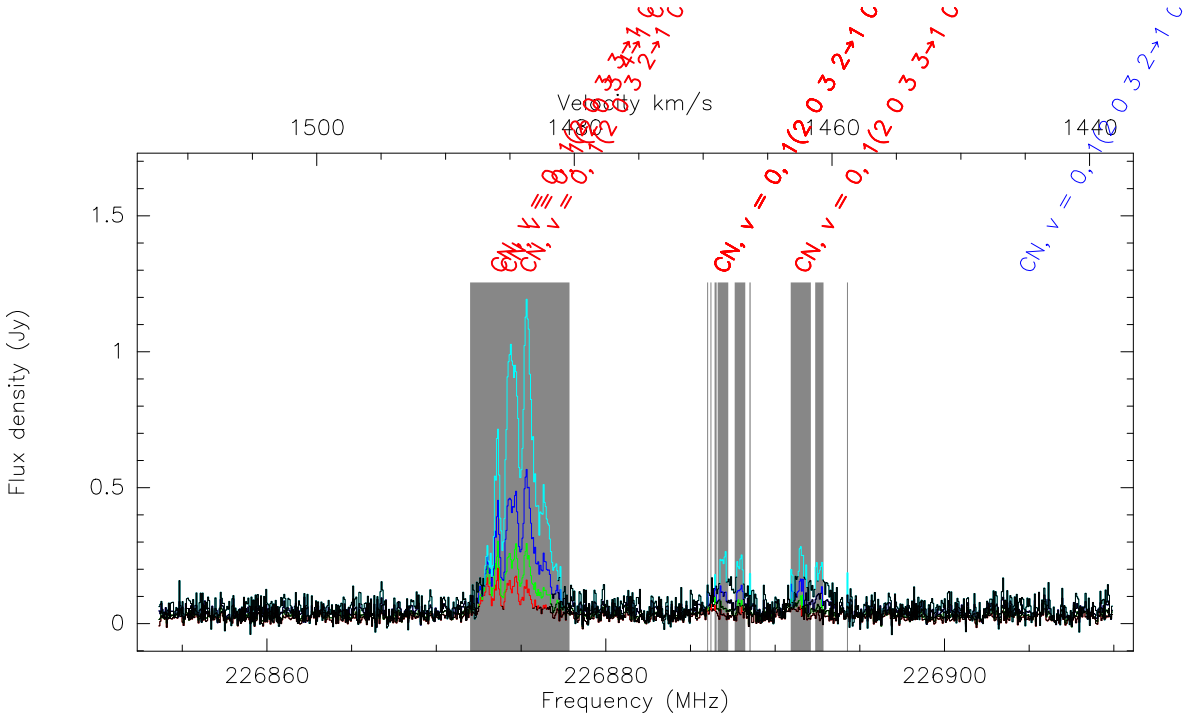

Figure 2: The UV_PREVIEW output

8 Visual Checks and Image Displays

The ultimate (and often sole) way to evaluate the data quality and suitability for scientific analysis is to visualize it. IMAGER offer simple, yet powerfull, tools to do so.

8.1 the UV_PREVIEW command

With large datasets, image can be long. IMAGER offers a simple, fast pre-imaging viewer through command UV_PREVIEW.

This command will display the spectra obtained towards the phase center at several spatial scales (the default is for 4 tapers). It will also attempt to detect spectral features, by analyzing for each spectrum the noise statistics and the outliers. It performs automatic line identifications, using a database in the LINEDB\ format in file $HOME/image.linedb . Potential spectral lines in the band are displayed in blue, and detected ones in red. UV_PREVIEW also warns the user about improper scaling of the data weights. The line emission region is indicated in grey.

The result of this automatic signal detection and line identification is saved in a SIC structure named PREVIEW%, that is automatically used by commands UV_BASELINE /CHANNEL and UV_FILTER /CHANNEL to respectively remove the continuum and filter the line emission to produce a continuum data set. The detected line frequencies are stored in PREVIEW%FOUND%FREQ and their names in PREVIEW%FOUND%LINES. This can be used to reference the velocity scale of the data to one of the detected lines, by using command SPECIFY FREQUENCY'PREVIEW%FOUND%FREQ[1]' for example before further processing.

An example is shown in Fig2

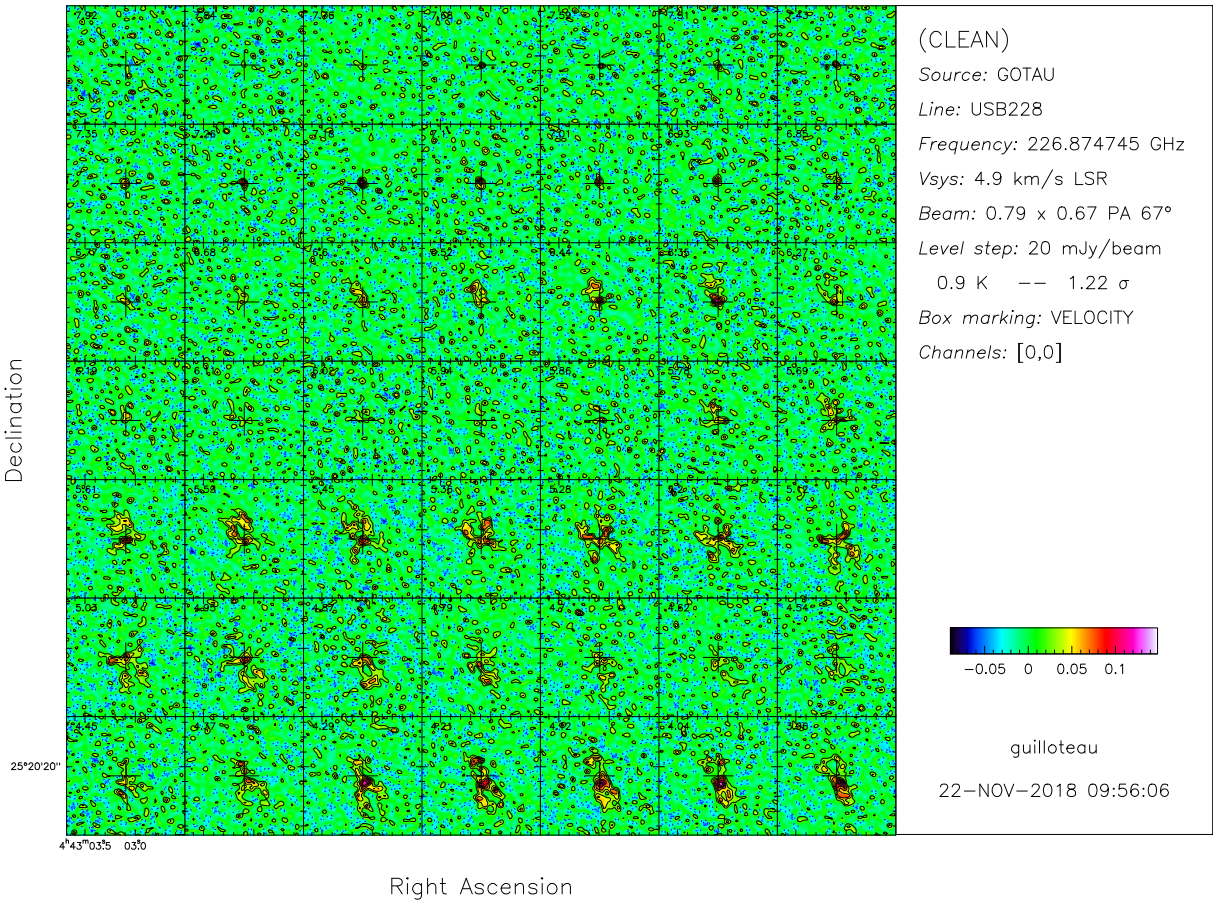

Figure 3: The `SHOW` output.

8.2 the `SHOW` command

In general, command `SHOW` allows a per-plane display of any SIC 3-D image variable, with contours overlaid on bitmap for each plane: e.g.
`SHOW DIRTY 30 -30` will display contour and bitmaps of each channel of the `DIRTY` image, starting for channel 30 and ending 30 channels before the last one. See Fig.3 for an example.

For *uv* data, `SHOW UV` can plot visibility values such as amplitude as a function of time, baseline length, etc..., again on a per channel basis.

`SHOW` can also display more specific issues:

- `SHOW CCT` will display the cumulative flux as a function of number of clean components (Fig.4).
- `SHOW COVERAGE` will display the *uv* coverage (it assumes there is only one, and not one per channel, because the display time is long, see Fig.5)
- `SHOW SELFCAL` behaves as `SELFCAL SHOW`
- `SHOW FIELDS` display the fields of a Mosaic.
- `SHOW NOISE` display the histogram of the intensity distribution for each channel, estimating the noise by fitting a Gaussian in these histograms (see Fig.6.

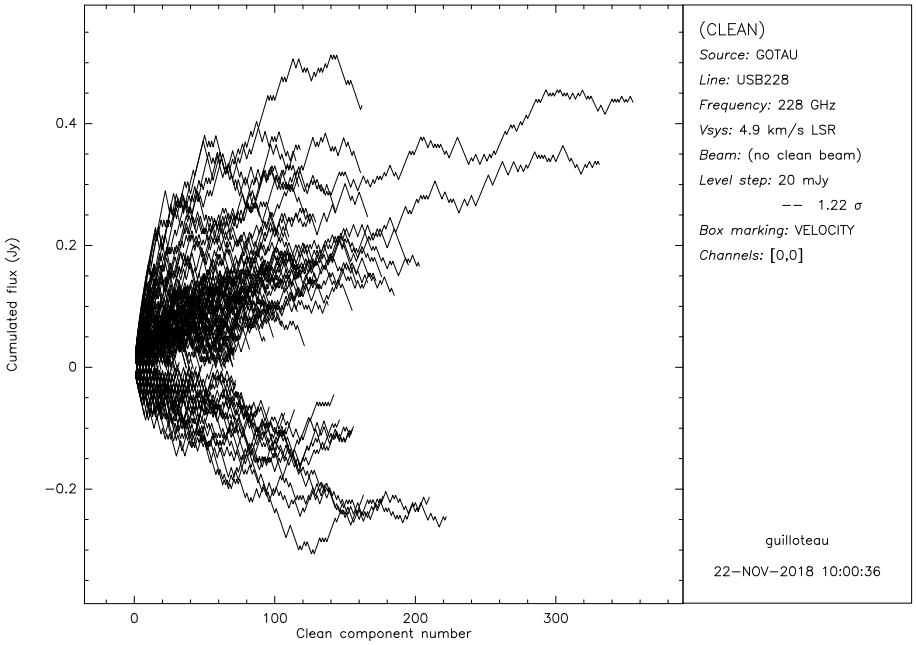

Figure 4: The `SHOW CCT` output.

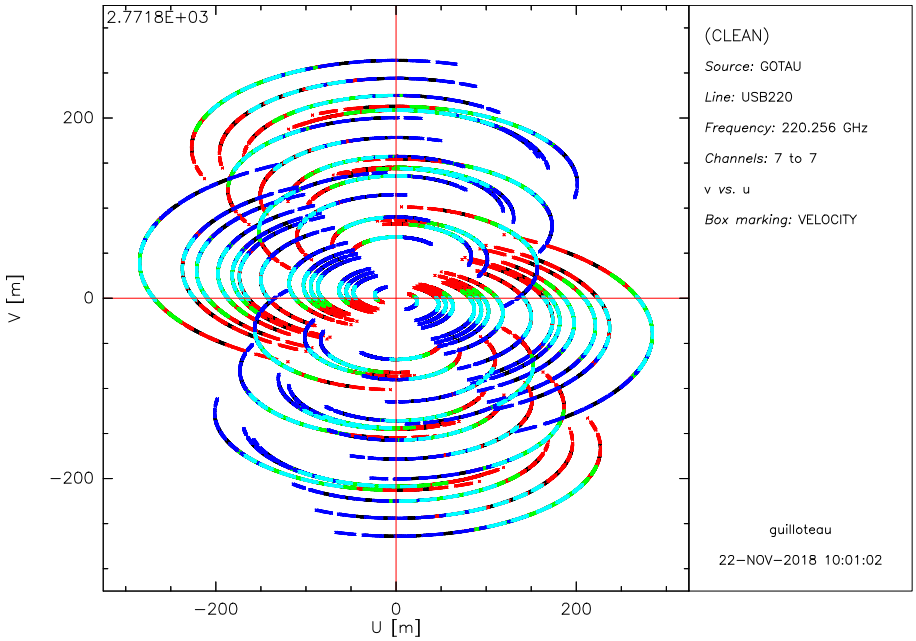

Figure 5: The `SHOW COVERAGE` output. Colors indicate different dates.

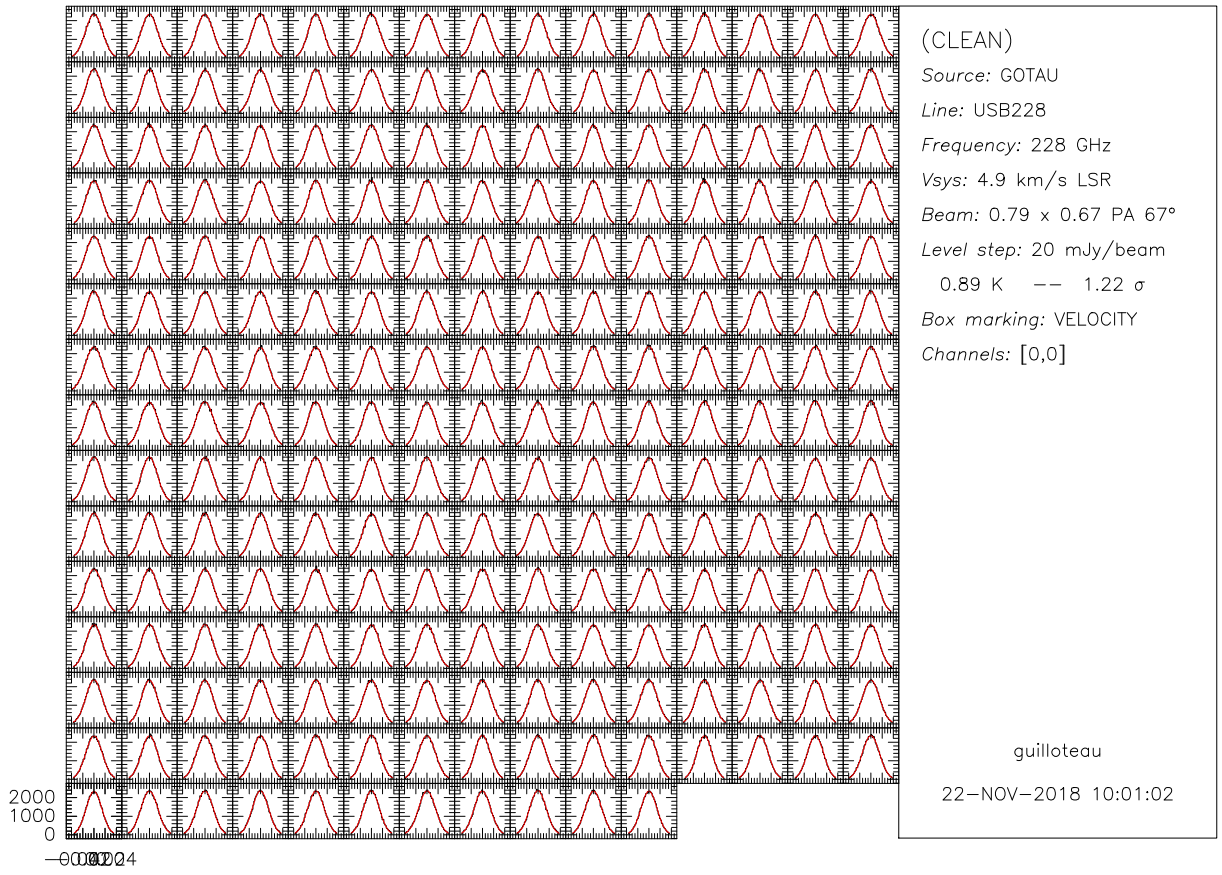

Figure 6: The `SHOW NOISE` output.

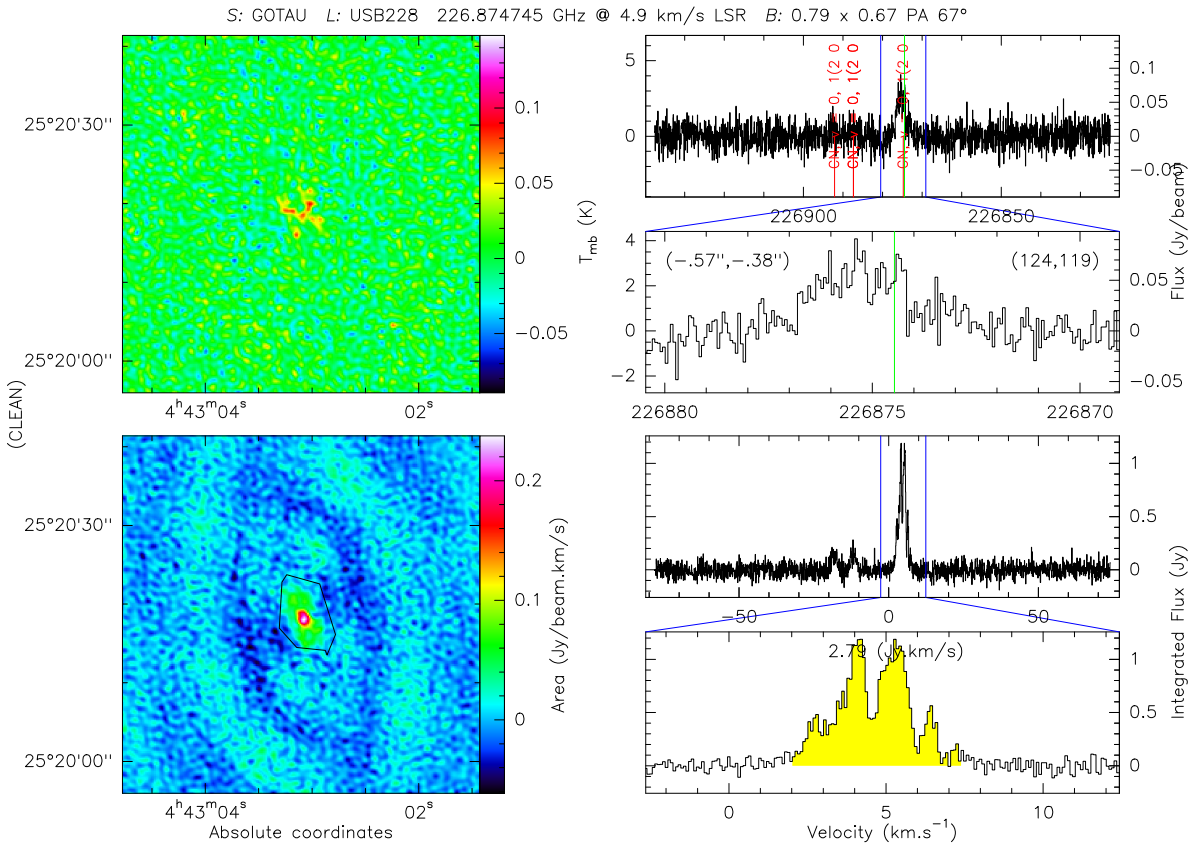

Figure 7: The `VIEW` output.

A direct use on Gildas 3-D image files is also possible: `SHOWFilename.ext` will directly display the file if possible.

8.3 the `VIEW` command

The `VIEW` command is a powerful alternative to `SHOW`, the later being inefficient when the number of spectral line channels is large.

`VIEW` provides a 4 panel display for 3-D data cubes, with the current channel bitmap, the integrated area bitmap, the current spectrum and the integrated intensity spectrum. The spectra can be displayed with 2 simultaneous frequency windows, a broad and a zoomed one, allowing browsing through a large number of channels. Spectral line identification can be

`VIEW CCT` will display the cumulative flux of Clean components for all channels in just one panel, instead of a per-channel panel for `SHOW CCT`

8.4 the `SELFCAL SHOW` command

Verifying the convergence of a self-calibration is important. Figure 8 shows an example, while Fig.9 shows the total phase correction between the original data and the last iteration. The displayed range can be controlled by variables `SELF_PRANGE[2]` for the Phase, `SELF_ARANGE[2]` for the Amplitude, and `SELF_TRANGE[2]` for the Time. Error bars are displayed if `ERROR_BARS` is `YES`, as shown in Fig.10 for amplitude.

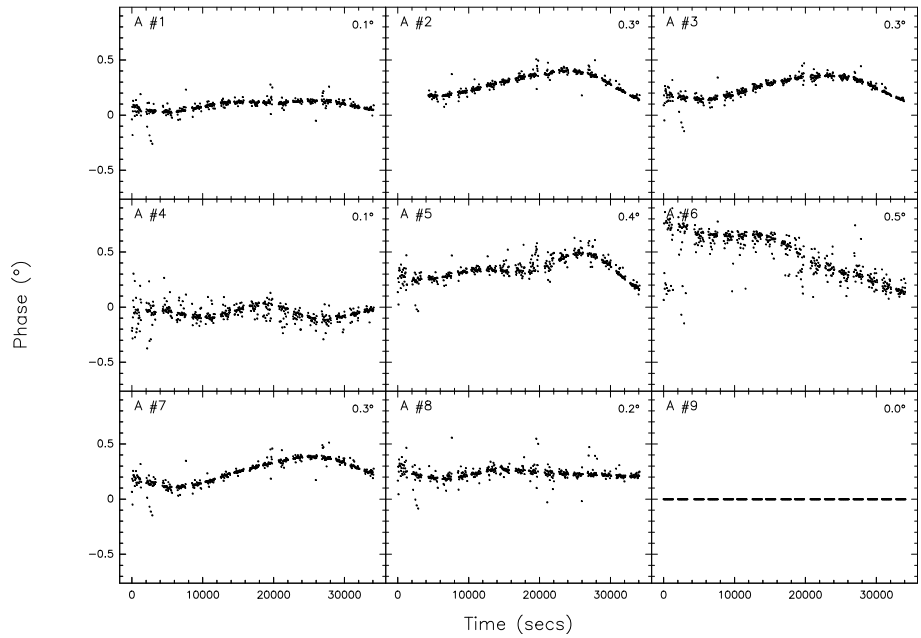

Figure 8: The `SELFCAL SHOW` output after a phase calibration, showing the convergence of the corrections between the last 2 iterations.

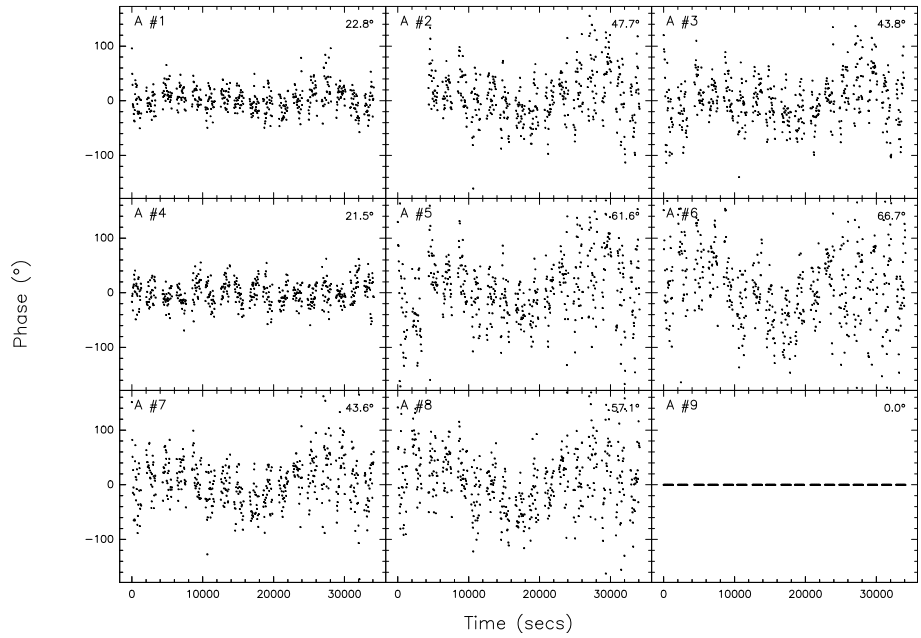

Figure 9: The `SELFCAL SHOW4` output after a phase calibration, showing the difference between iteration 4 and the original data.

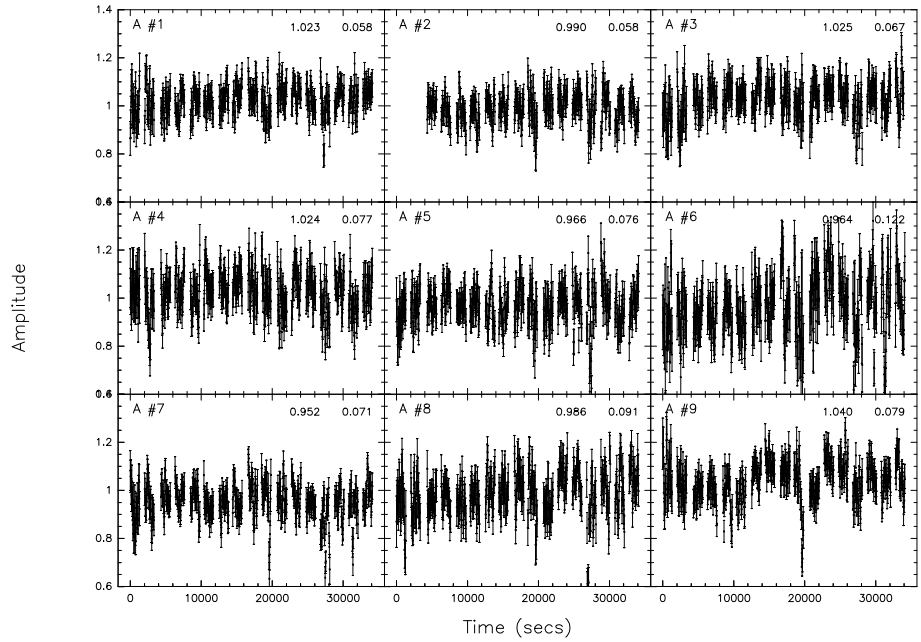

Figure 10: The SELFCAL SHOW4 output after an amplitude self calibration, showing the difference between iteration 4 and the original data, with the error bars.

9 CLEAN Language Internal Help

9.1 Language

```
ALMA          : Joint deconvolution of ALMA and ACA dirty images
CLEAN         : Deconvolve a dirty image using the current METHOD
CLARK         : Deconvolve a dirty image using the CLARK clean algorithm
DUMP          : Dump the control parameters of the deconvolution algorithms
FIT           : Fit the dirty beam
HOGBOM        : Deconvolve a dirty image using the basic clean algorithm
MAP_RESAMPLE  :  Resample a Map on a different Velocity/Frequency scale
MAP_INTEGRATE :  Compute a Moment 0 Map
MAP_COMPRESS  :  Average a Map by several channels
MOSAIC        : Toggle the mosaic mode
MULTI         : Deconvolve a dirty image using the MULTI-SCALE Clean
MRC           : Deconvolve a dirty image using the MULTI-RESOLUTION Clean
MX            : Iteratively image and deconvolve a dirty image
PRIMARY       : Apply primary beam correction
READ          : Read the input files in internal buffers
SDI           : Deconvolve a dirty image using the Steer-Dewdney-Ito Clean
SHOW          : Display (in a plot) some internal buffer result
SPECIFY       : Change the Frequency / Velocity scale or the Telescope
STATISTIC     : Compute statistics on image
STOKES        : Extract one polarization state from a polarized UV table
SUPPORT       : Define the support used to search clean components
UV_BASELINE   : Subtract a continuum baseline from a Line UV data
```

```
UV_CHECK      : Check UV data for null visibilities or per channel flags.
UV_COMPRESS   : Compress a Line UV data into another Line UV data
UV_CONTINUUM  : Compress a Line UV data into a Continuum UV data
UV_FILTER     : Filter out (line) channels
UV_FLAG       : Interactively flag UV data
UV_MAP        : Build the dirty image and beam from a UV table
UV_RESAMPLE   : Resample (in Velocity) the UV data
UV_RESIDUAL   : Subtract Clean Component from the UV Data
UV_RESTORE    : Restore a Clean image from UV data and Clean Components
UV_SHIFT      : Shift a UV table to common phase center
UV_SORT       : Sort and Transpose UV data for plotting
UV_STAT       : Gives beam sizes and noise properties as a function of
                  tapering or robust weighting parameter
UV_TIME       : Time average the current UV data
UV_TRUNCATE   : Truncate the baseline range of the UV data
VIEW          : Show (in a GO VIEW-like plot) some internal buffer result
WRITE         : Save internal buffers or Image variables into output files
```

9.2 ALMA

[CLEAN\]ALMA [FirstPlane [LastPlane]] [/PLOT Clean|Residu] [/FLUX Fmin Fmax] [/QUERY] [/NOISE] [/METHOD]

Joint deconvolution methods specific to ALMA+ACA observations.

The ALMA simulator can be accessed either by typing "@ alma" at the prompt or from the MAPPING main menu.

9.3 CLARK

[CLEAN\]CLARK [FirstPlane [LastPlane]] [/PLOT Clean|Residu] [/FLUX Fmin Fmax] [/QUERY]

A Major-Minor cycles CLEAN method, originally developed by B.Clark, in which clean components are selected using a limited beam patch, and deconvolved through Fourier transform at each major cycle. In mosaic mode (See command MOSAIC), a mosaic clean is performed. Rings and/or stripes may appear on extended sources. Faster than the Hogbom method for single fields but maybe slower for mosaics. The strategy to search for CLEAN components in CLARK does not work properly when the secondary side lobes are too large (e.g. larger than 0.3), or in case of high phase noise.

Clean the specified plane interval (default: planes between variables FIRST and LAST). If only FirstPlane is specified, Clean only that plane.

```
If option /PLOT is given, a display of the CLEAN or RESIDUAL map will be
shown  at  each major cycle, depending on the argument (default: Residu-
al). The user will be prompted for continuation when the  /QUERY  option
is  present.  The  cumulative, already cleaned flux is displayed in real
time in an additional window while cleaning goes on when the  /FLUX  op-
tion  is  present.  Parameters of the /FLUX option are then used to give
the flux limits for this display.

The user can control the algorithm through SIC variables. New values can
be given using "LET VARIABLE value". For ease of use, and whenever it is
possible, a sensible value of each parameter will automatically be  com-
puted from the context if the value of the corresponding variable is set
to its default value, i.e. zero value and empty string. A few  variables
are initialized to "reasonable" values.

    [CLEAN\]CLEAN ?
Will  list  all main CLEAN_* variables controlling the CLEAN parameters.
HELP CLEAN Variables will give a more complete list.
```

9.3.1 CLARK Variables:

```
Basic parameters
CLEAN_GAIN       [       ] Loop gain
CLEAN_NITER      [       ] Maximum number of clean components
CLEAN_FRES       [      %] Maximum value of residual (Fraction of peak)
CLEAN_ARES       [Jy/Beam] Maximum value of residual (Absolute)
CLEAN_POSITIVE   [       ] Minimum number of positive components at start
CLEAN_NKEEP      [       ] Min number of components before convergence

Old names like in MAPPING
BLC              [  pixel] Bottom left corner of cleaning box
TRC              [  pixel] Top right corner of cleaning box
MAJOR            [ arcsec] Clean beam major axis
MINOR            [ arcsec] Clean beam minor axis
ANGLE            [ degree] Position angle of clean beam
BEAM_PATCH       [  pixel] Size of cleaning beam ** not clear **

Method dependent parameters
CLEAN_INFLATE    [      ] Maximum Inflation factor for UV_RESTORE (MuLTISCAL
CLEAN_NCYCLE     [      ] Max number of Major Cycles (SDI & CLARK methods)
CLEAN_NGOAL      [      ] Max number of comp. in Cycles (ALMA method)
CLEAN_RESTORE    [      ] Threshold for restoring a Mosaic (def 0.2)
CLEAN_SEARCH     [      ] Threshold to search Clean Comp. in a Mosaic (def 0
CLEAN_SIDELOBE   [      ] Min threshold to fit the synthesized beam
CLEAN_SMOOTH     [      ] Smoothing ratio (MRC and MULTISCALE)
```

```
CLEAN_SPEEDY     [      ] Speed-up factor (CLARK)
CLEAN_WORRY      [      ] Worry factor (MULTISCALE)
```

9.3.2 CLARK CLEAN_ARES

This is the minimal flux in the dirty map that the program will consider as significant. Alternatively, the threshold can be specified as a fraction of the peak flux using CLEAN_FRES. Once this level has been reached the program stops subtracting, and starts the restoration phase. The unit for this parameter is the map unit (typically Jy/Beam). The parameter should usually be of the order of magnitude of the expected noise in the clean map.

If 0, CLEAN_FRES will be used instead. If all of CLEAN_NITER, CLEAN_ARES and CLEAN_FRES are 0, an absolute residual equal to the noise level will be used for CLEAN_ARES.

Short form is ARES.

9.3.3 CLARK CLEAN_FRES

This is the minimal fraction of the peak flux in the dirty map that the program will consider as significant. Alternatively, an absolute threshold can be specified using CLEAN_ARES. Once this level has been reached the program stops subtracting, and starts the restoration phase. This parameter is normalized to 1 (neither in % nor in db). It should usually be of the order of magnitude of the inverse of the expected dynamic range of the intensity.

If 0, CLEAN_ARES will be used instead. If all of CLEAN_NITER, CLEAN_ARES and CLEAN_FRES are 0, an absolute residual equal to the noise level will be used for CLEAN_ARES.

Short form is FRES.

9.3.4 CLARK CLEAN_GAIN

This is the gain of the subtraction loop. It should typically be chosen in the range 0.05 and 0.3. Higher values give faster convergence, while lower values give a better restitution of the extended structure. A sensible default is 0.2.

Short form is GAIN.

9.3.5 CLARK CLEAN_NITER

This is the maximum number of components the program will accept to subtract. Once it has been reached, the program starts the restoration phase.

If 0, the program will guess a number, based on the image size and maximum signal-to-noise ratio, and specified residual level CLEAN_ARES and/or CLEAN_FRES.

Short form is NITER.

9.3.6 CLARK CLEAN_NKEEP

This is an integer specifying the minimum number of Clean components before testing if Cleaning has converged. The convergence is criterium is a comparison of the cumulative flux evolution separated by CLEAN_NKEEP components. If th

IF CLEAN_NKEEP is 0, CLEAN will ignore this convergence criterium, and continue clean until the CLEAN_NITER, CLEAN_ARES or CLEAN_FRES criteria indicate to stop.

With CLEAN_NKEEP > 0, CLEAN will explore the stability of the total clean flux over the last CLEAN_NKEEP iterations. For a positive (resp. negative) source, if the Clean flux becomes smaller (resp. larger) than the Clean flux CLEAN_NKEEP iterations earlier, CLEAN will stop.

Using CLEAN_NKEEP about 70 is a reasonable value. Some special cases (faint extended sources) may require larger values of CLEAN_NKEEP.

9.3.7 CLARK CLEAN_POSITIVE

The minimum number of positive components before negative ones are selected.

9.3.8 CLARK CLEAN_RESTORE

Fraction of peak response of the primary beams coverage under which the Sky brightness image is blanked in a Mosaic deconvolution.

The default is 0.2.

9.3.9 CLARK CLEAN_SEARCH

Fraction of peak response of the primary beams coverage beyond which no Clean component is searched in a Mosaic deconvolution.

The default is 0.2.

9.3.10 CLARK CLEAN_SIDELOBE

Minimal relative intensity to consider for fitting the syntheized beam to obtain the Clean beam parameters (MAJOR, MINOR and ANGLE) when 0. The default is 0.35.

In case of poor UV coverage, CLEAN_SIDELOBE should be higher than the maximum sidelobe level to perform a good Gaussian fit. Some particularly bad UV coverage may not allow any good fit at all, however.

9.3.11 CLARK CLEAN_NGOAL

Number of clean components to be selected in a Cycle in the ALMA heterogeneous array cleaning method.

9.3.12 CLARK CLEAN_NCYCLE

Maximum number of Major Cycles for the SDI and CLARK methods.

9.3.13 CLARK CLEAN_SMOOTH

Smoothing factor between different scales in the MRC and MULTISCALE methods. The default is sqrt(3).

9.3.14 CLARK CLEAN_SPEEDY

Speed-up factor for the CLARK major cycles. The default is 1.0. Larger values may be used, but at the expense of possible instabilities of the algorithm.

9.3.15 CLARK CLEAN_WORRY

Worry factor in the MULTISCALE method for convergence. It propagates the S/N from one iteration to the other, so that if this S/N degrades, the method stops. Default is 0 (no propagation, and hence no test on S/N). The value should be < 1.0 in all cases.

9.3.16 CLARK CLEAN_INFLATE

Maximum Inflation factor for UV_RESTORE (MULTISCALE method). If the number of true (i.e. pixel based) Clean components found by MULTISCALE is larger than CLEAN_INFLATE times the number of compressed (i.e. those with the smoothing factor information) components, expansion of the compressed components will not be possible, and UV_RESTORE will not be useable.

A default of 50 is in general adequate. Better solutions might be found in the future, and this parameter suppressed. Apart from memory usage, this number has no consequence on the algorithm.

9.3.17 CLARK METHOD

Method used for the deconvolution. Can be HOGBOM, MULTI, MRC, SDI or CLARK.

9.3.18 CLARK Old_Names:

Some of the CLEAN parameters have kept their old names: MAJOR, MINOR, ANGLE (which are also used by command FIT) BLC, TRC and BEAM_PATCH (which are seldom used)

Others have equivalent short names: ARES, FRES, GAIN, NITER for which the CLEAN_ prefix may be omitted.

9.3.19 CLARK BLC

These are the (pixel) coordinates of the Bottom Left Corner of the

cleaning box. The actual cleaning support will be the intersection of the specified window with the inner quarter of the map and with any user defined polygon.

9.3.20 CLARK TRC

These are the (pixel) coordinates of the Top Right Corner of the cleaning box. The actual cleaning window will be the intersection of the specified window with the inner quarter of the map and with any user defined polygon.

9.3.21 CLARK MAJOR

This is the major axis (FWHP) in user coordinates of the Gaussian restoring beam. If 0, the program will fit a Gaussian to the dirty beam. We strongly discourage to change the default value of 0.

9.3.22 CLARK MINOR

This is the minor axis (FWHP) in user coordinates of the Gaussian restoring beam. If 0, the program will fit a Gaussian to the dirty beam. We strongly discourage to change the default value of 0.

9.3.23 CLARK ANGLE

This is the position angle (from North towards East, i.e. anticlockwise) of the major axis of the Gaussian restoring beam (in degrees). If 0, the program will fit a Gaussian to the dirty beam. We strongly discourage to change the default value of 0.

9.3.24 CLARK BEAM_PATCH

The dirty beam patch to be used for the minor cycles in CLARK and MRC method. It should be large enough to avoid doing too many major cycles, but has practically no influence on the result. This size should be specified in pixel units. Reasonable values are between N/8 and N/4, where N is the number of map pixels in the same dimension. If set to N, the CLARK algorithm becomes identical to the HOGBOM algorithm.

9.4 CLEAN

[CLEAN\]CLEAN [FirstPlane [LastPlane]] [/PLOT Clean|Residu] [/FLUX Fmin Fmax] [/QUERY] [/NITER NiterList] [/ARES AresList]

Deconvolve a Mosaic or Single-field using the current METHOD (in SIC variable METHOD). See INPUT CLEAN for the other SIC variables controlling the deconvolution process.

Clean the specified plane interval (default: planes between variables FIRST and LAST). If only FirstPlane is specified, Clean only that plane.

Supported methods are CLARK, HOGBOM, MRC, MULTISCALE and SDI. See help of each of these commands for further details on each algorithm. This command allows a per-plane definition of the convergence criteria CLEAN_NITER and CLEAN_ARES.

The user can control the algorithm through SIC variables. New values can be given using "LET VARIABLE value". For ease of use, and whenever it is possible, a sensible value of each parameter will automatically be computed from the context if the value of the corresponding variable is set to its default value, i.e. zero value and empty string. A few variables are initialized to "reasonable" values.

[CLEAN\]CLEAN ?
Will list all main CLEAN_* variables controlling the CLEAN parameters. HELP CLEAN Variables will give a more complete list.

9.4.1 CLEAN /FLUX

[CLEAN\]CLEAN [FirstPlane[LastPlane]] /FLUX Fmin Fmax [/PLOT Clean|Residu] [/QUERY] [/NITER NiterList] [/ARES AresList]

Display the cumulative Clean flux as Clean progresses. This option is inactive in Parallel mode.

9.4.2 CLEAN /PLOT

[CLEAN\]CLEAN [FirstPlane [LastPlane]] /PLOT Clean|Residu [/FLUX Fmin Fmax] [/QUERY] [/NITER NiterList] [/ARES AresList]

Display the iterated Clean or Residual image for Cleaning methods which have major cycles (CLARK or SDI). This option is inactive in Parallel

mode.

9.4.3 CLEAN /QUERY

[CLEAN\]CLEAN [FirstPlane [LastPlane]] /PLOT Clean|Residu /QUERY [/FLUX Fmin Fmax] [/NITER NiterList] [/ARES AresList]

*** Obsolescent ***

Prompt for continuation when a Major cycle is complete. This option is inactive in Parallel mode.

9.4.4 CLEAN /NITER

[CLEAN\]CLEAN [FirstPlane [LastPlane]] /NITER NiterList [/PLOT Clean|Residu] [/FLUX Fmin Fmax] [/QUERY] [/ARES AresList]

Use a per-plane value for the number of iterations, instead of the global NITER variable. NiterList should be a 1-D integer array of dimension the number of channels.

This option is only available through the CLEAN command, not through the specific command of each method.

9.4.5 CLEAN /ARES

[CLEAN\]CLEAN [FirstPlane [LastPlane]] /ARES AresList [/PLOT Clean|Residu] [/FLUX Fmin Fmax] [/QUERY] [/NITER NiterList]

Use a per-plane value for the absolute residual used to stop cleaning, instead of the global ARES variable. AresList should be a 1-D real array of dimension the number of channels.

This option is only available through the CLEAN command, not through the specific command of each method.

9.4.6 CLEAN Variables:

```
Basic parameters
CLEAN_GAIN       [       ] Loop gain
CLEAN_NITER      [       ] Maximum number of clean components
CLEAN_FRES       [      %] Maximum value of residual (Fraction of peak)
```

```
CLEAN_ARES       [Jy/Beam] Maximum value of residual (Absolute)
CLEAN_POSITIVE   [       ] Minimum number of positive components at start
CLEAN_NKEEP      [       ] Min number of components before convergence

Old names like in MAPPING
BLC              [  pixel] Bottom left corner of cleaning box
TRC              [  pixel] Top right corner of cleaning box
MAJOR            [ arcsec] Clean beam major axis
MINOR            [ arcsec] Clean beam minor axis
ANGLE            [ degree] Position angle of clean beam
BEAM_PATCH       [  pixel] Size of cleaning beam ** not clear **

Method dependent parameters
CLEAN_INFLATE    [      ] Maximum Inflation factor for UV_RESTORE (MuLTISCAL
CLEAN_NCYCLE     [      ] Max number of Major Cycles (SDI & CLARK methods)
CLEAN_NGOAL      [      ] Max number of comp. in Cycles (ALMA method)
CLEAN_RESTORE    [      ] Threshold for restoring a Mosaic (def 0.2)
CLEAN_SEARCH     [      ] Threshold to search Clean Comp. in a Mosaic (def 0
CLEAN_SIDELOBE   [      ] Min threshold to fit the synthesized beam
CLEAN_SMOOTH     [      ] Smoothing ratio (MRC and MULTISCALE)
CLEAN_SPEEDY     [      ] Speed-up factor (CLARK)
CLEAN_WORRY      [      ] Worry factor (MULTISCALE)
```

9.4.7 CLEAN CLEAN_ARES

This is the minimal flux in the dirty map that the program will consider as significant. Alternatively, the threshold can be specified as a fraction of the peak flux using CLEAN_FRES. Once this level has been reached the program stops subtracting, and starts the restoration phase. The unit for this parameter is the map unit (typically Jy/Beam). The parameter should usually be of the order of magnitude of the expected noise in the clean map.

If 0, CLEAN_FRES will be used instead. If all of CLEAN_NITER, CLEAN_ARES and CLEAN_FRES are 0, an absolute residual equal to the noise level will be used for CLEAN_ARES.

Short form is ARES.

9.4.8 CLEAN CLEAN_FRES

This is the minimal fraction of the peak flux in the dirty map that the program will consider as significant. Alternatively, an absolute threshold can be specified using CLEAN_ARES. Once this level has been

reached the program stops subtracting, and starts the restoration phase. This parameter is normalized to 1 (neither in % nor in db). It should usually be of the order of magnitude of the inverse of the expected dynamic range of the intensity.

If 0, CLEAN_ARES will be used instead. If all of CLEAN_NITER, CLEAN_ARES and CLEAN_FRES are 0, an absolute residual equal to the noise level will be used for CLEAN_ARES.

Short form is FRES.

9.4.9 CLEAN CLEAN_GAIN

This is the gain of the subtraction loop. It should typically be chosen in the range 0.05 and 0.3. Higher values give faster convergence, while lower values give a better restitution of the extended structure. A sensible default is 0.2.

Short form is GAIN.

9.4.10 CLEAN CLEAN_NITER

This is the maximum number of components the program will accept to subtract. Once it has been reached, the program starts the restoration phase.

If 0, the program will guess a number, based on the image size and maximum signal-to-noise ratio, and specified residual level CLEAN_ARES and/or CLEAN_FRES.

Short form is NITER.

9.4.11 CLEAN CLEAN_NKEEP

This is an integer specifying the minimum number of Clean components before testing if Cleaning has converged. The convergence is criterium is a comparison of the cumulative flux evolution separated by CLEAN_NKEEP components. If th

IF CLEAN_NKEEP is 0, CLEAN will ignore this convergence criterium, and continue clean until the CLEAN_NITER, CLEAN_ARES or CLEAN_FRES criteria indicate to stop.

With CLEAN_NKEEP > 0, CLEAN will explore the stability of the total clean flux over the last CLEAN_NKEEP iterations. For a positive (resp. negative) source, if the Clean flux becomes smaller (resp. larger) than the Clean flux CLEAN_NKEEP iterations earlier, CLEAN will stop.

Using CLEAN_NKEEP about 70 is a reasonable value. Some special cases (faint extended sources) may require larger values of CLEAN_NKEEP.

9.4.12 CLEAN CLEAN_POSITIVE

The minimum number of positive components before negative ones are selected.

9.4.13 CLEAN CLEAN_RESTORE

Fraction of peak response of the primary beams coverage under which the Sky brightness image is blanked in a Mosaic deconvolution.

The default is 0.2.

9.4.14 CLEAN CLEAN_SEARCH

Fraction of peak response of the primary beams coverage beyond which no Clean component is searched in a Mosaic deconvolution.

The default is 0.2.

9.4.15 CLEAN CLEAN_SIDELOBE

Minimal relative intensity to consider for fitting the syntheized beam to obtain the Clean beam parameters (MAJOR, MINOR and ANGLE) when 0. The default is 0.35.

In case of poor UV coverage, CLEAN_SIDELOBE should be higher than the maximum sidelobe level to perform a good Gaussian fit. Some particularly bad UV coverage may not allow any good fit at all, however.

9.4.16 CLEAN CLEAN_NGOAL

Number of clean components to be selected in a Cycle in the ALMA heterogeneous array cleaning method.

9.4.17 CLEAN CLEAN_NCYCLE

Maximum number of Major Cycles for the SDI and CLARK methods.

9.4.18 CLEAN CLEAN_SMOOTH

Smoothing factor between different scales in the MRC and MULTISCALE methods. The default is sqrt(3).

9.4.19 CLEAN CLEAN_SPEEDY

Speed-up factor for the CLARK major cycles. The default is 1.0. Larger values may be used, but at the expense of possible instabilities of the algorithm.

9.4.20 CLEAN CLEAN_WORRY

Worry factor in the MULTISCALE method for convergence. It propagates the S/N from one iteration to the other, so that if this S/N degrades, the method stops. Default is 0 (no propagation, and hence no test on S/N). The value should be < 1.0 in all cases.

9.4.21 CLEAN CLEAN_INFLATE

Maximum Inflation factor for UV_RESTORE (MULTISCALE method). If the number of true (i.e. pixel based) Clean components found by MULTISCALE is larger than CLEAN_INFLATE times the number of compressed (i.e. those with the smoothing factor information) components, expansion of the compressed components will not be possible, and UV_RESTORE will not be useable.

A default of 50 is in general adequate. Better solutions might be found in the future, and this parameter suppressed. Apart from memory usage, this number has no consequence on the algorithm.

9.4.22 CLEAN METHOD

Method used for the deconvolution. Can be HOGBOM, MULTI, MRC, SDI or CLARK.

9.4.23 CLEAN Old_Names:

Some of the CLEAN parameters have kept their old names: MAJOR, MINOR, ANGLE (which are also used by command FIT) BLC, TRC and BEAM_PATCH (which are seldom used)

Others have equivalent short names: ARES, FRES, GAIN, NITER for which the CLEAN_ prefix may be omitted.

9.4.24 CLEAN BLC

These are the (pixel) coordinates of the Bottom Left Corner of the cleaning box. The actual cleaning support will be the intersection of the specified window with the inner quarter of the map and with any user defined polygon.

9.4.25 CLEAN TRC

These are the (pixel) coordinates of the Top Right Corner of the cleaning box. The actual cleaning window will be the intersection of the specified window with the inner quarter of the map and with any user defined polygon.

9.4.26 CLEAN MAJOR

This is the major axis (FWHP) in user coordinates of the Gaussian restoring beam. If 0, the program will fit a Gaussian to the dirty beam. We strongly discourage to change the default value of 0.

9.4.27 CLEAN MINOR

This is the minor axis (FWHP) in user coordinates of the Gaussian

restoring beam. If 0, the program will fit a Gaussian to the dirty beam. We strongly discourage to change the default value of 0.

9.4.28 CLEAN ANGLE

This is the position angle (from North towards East, i.e. anticlockwise) of the major axis of the Gaussian restoring beam (in degrees). If 0, the program will fit a Gaussian to the dirty beam. We strongly discourage to change the default value of 0.

9.4.29 CLEAN BEAM_PATCH

The dirty beam patch to be used for the minor cycles in CLARK and MRC method. It should be large enough to avoid doing too many major cycles, but has practically no influence on the result. This size should be specified in pixel units. Reasonable values are between N/8 and N/4, where N is the number of map pixels in the same dimension. If set to N, the CLARK algorithm becomes identical to the HOGBOM algorithm.

9.5 DUMP

[CLEAN\]DUMP [U]

Dump on screen the control parameters of the different CLEAN deconvolution algorithms, mainly for debugging purpose. "DUMP U" dumps the parameters as input by the user while "DUMP" dumps the parameters really used and/or modified by the deconvolution algorithm.

9.6 FIT

[CLEAN\]FIT [Field]

Fit the dirty beam to obtain the clean beam parameters. This is done automatically by all CLEAN algorithm when needed. In mosaic mode, you can specify on which field the fit has to be made, using the first argument.

Fixed values: Clean beam parameters can also be specified by the users, by setting the variables MAJOR, MINOR and ANGLE to values other than their default values (0,0,0). In this case, no automatic fit will be performed. We strongly discourage the use of those variables, which may

result in a very improper flux scaling if the beam size is inappropriate. This mode should be reserved to special cases where the beam cannot be properly fitted, or to have a circular beam by taking as beam size the geometrical mean of the fitted major and minor sizes.

9.6.1 FIT CLEAN_SIDELOBE

Minimal relative intensity to consider for fitting the syntheized beam to obtain the Clean beam parameters (MAJOR, MINOR and ANGLE) when 0. The default is 0.30.

In case of poor UV coverage, CLEAN_SIDELOBE should be higher than the maximum sidelobe level to perform a good Gaussian fit. Some particularly bad UV coverage may not allow any good fit at all, however.

9.7 HOGBOM

```
[CLEAN\]HOGBOM [FirstPlane [LastPlane]] [/FLUX Fmin Fmax]
```

See HELP CLEAN for the SIC variables controlling the deeconvolution process.

The simplest CLEAN algorithm, originally developed by Hogbom. In mosaic mode (See command MOSAIC), a mosaic clean is performed. Rings and or strips may appear on extended sources. It is slower than CLARK for a single field but maybe faster for a mosaic. It is extremely robust. Cleaning can be interrupted by pressing C at any time.

Clean the specified plane interval (default: planes between variables FIRST and LAST). If only FirstPlane is specified, Clean only that plane.

The cumulative, already cleaned flux is displayed in real time in an additional window while cleaning goes on when the /FLUX option is present. Parameters of the /FLUX option are then used to give the flux limits for this display.

The user can control the algorithm through SIC variables. New values can be given using "LET VARIABLE value". For ease of use, and whenever it is possible, a sensible value of each parameter will automatically be computed from the context if the value of the corresponding variable is set to its default value, i.e. zero value and empty string. A few variables are initialized to "reasonable" values.

```
[CLEAN\]CLEAN ?
```

Will list all main CLEAN_* variables controlling the CLEAN parameters.

```
HELP CLEAN Variables will give a more complete list.
```

9.7.1 HOGBOM Variables:

```
Basic parameters
CLEAN_GAIN       [       ] Loop gain
CLEAN_NITER      [       ] Maximum number of clean components
CLEAN_FRES       [      %] Maximum value of residual (Fraction of peak)
CLEAN_ARES       [Jy/Beam] Maximum value of residual (Absolute)
CLEAN_POSITIVE   [       ] Minimum number of positive components at start
CLEAN_NKEEP      [       ] Min number of components before convergence

Old names like in MAPPING
BLC              [  pixel] Bottom left corner of cleaning box
TRC              [  pixel] Top right corner of cleaning box
MAJOR            [ arcsec] Clean beam major axis
MINOR            [ arcsec] Clean beam minor axis
ANGLE            [ degree] Position angle of clean beam
BEAM_PATCH       [  pixel] Size of cleaning beam ** not clear **

Method dependent parameters
CLEAN_INFLATE    [      ] Maximum Inflation factor for UV_RESTORE (MuLTISCAL
CLEAN_NCYCLE     [      ] Max number of Major Cycles (SDI & CLARK methods)
CLEAN_NGOAL      [      ] Max number of comp. in Cycles (ALMA method)
CLEAN_RESTORE    [      ] Threshold for restoring a Mosaic (def 0.2)
CLEAN_SEARCH     [      ] Threshold to search Clean Comp. in a Mosaic (def 0
CLEAN_SIDELOBE   [      ] Min threshold to fit the synthesized beam
CLEAN_SMOOTH     [      ] Smoothing ratio (MRC and MULTISCALE)
CLEAN_SPEEDY     [      ] Speed-up factor (CLARK)
CLEAN_WORRY      [      ] Worry factor (MULTISCALE)
```

9.7.2 HOGBOM CLEAN_ARES

```
This is the minimal flux in the dirty map that the program will consider
as significant. Alternatively, the threshold can be specified as a
fraction of the peak flux using CLEAN_FRES. Once this level has been
reached the program stops subtracting, and starts the restoration phase.
The unit for this parameter is the map unit (typically Jy/Beam). The
parameter should usually be of the order of magnitude of the expected
noise in the clean map.

If 0, CLEAN_FRES will be used instead. If all of CLEAN_NITER, CLEAN_ARES
and CLEAN_FRES are 0, an absolute residual equal to the noise level will
be used for CLEAN_ARES.
```

Short form is ARES.

9.7.3 HOGBOM CLEAN_FRES

This is the minimal fraction of the peak flux in the dirty map that the program will consider as significant. Alternatively, an absolute threshold can be specified using CLEAN_ARES. Once this level has been reached the program stops subtracting, and starts the restoration phase. This parameter is normalized to 1 (neither in % nor in db). It should usually be of the order of magnitude of the inverse of the expected dynamic range of the intensity.

If 0, CLEAN_ARES will be used instead. If all of CLEAN_NITER, CLEAN_ARES and CLEAN_FRES are 0, an absolute residual equal to the noise level will be used for CLEAN_ARES.

Short form is FRES.

9.7.4 HOGBOM CLEAN_GAIN

This is the gain of the subtraction loop. It should typically be chosen in the range 0.05 and 0.3. Higher values give faster convergence, while lower values give a better restitution of the extended structure. A sensible default is 0.2.

Short form is GAIN.

9.7.5 HOGBOM CLEAN_NITER

This is the maximum number of components the program will accept to subtract. Once it has been reached, the program starts the restoration phase.

If 0, the program will guess a number, based on the image size and maximum signal-to-noise ratio, and specified residual level CLEAN_ARES and/or CLEAN_FRES.

Short form is NITER.

9.7.6 HOGBOM CLEAN_NKEEP

This is an integer specifying the minimum number of Clean components before testing if Cleaning has converged. The convergence is criterium is a comparison of the cumulative flux evolution separated by CLEAN_NKEEP components. If th

IF CLEAN_NKEEP is 0, CLEAN will ignore this convergence criterium, and continue clean until the CLEAN_NITER, CLEAN_ARES or CLEAN_FRES criteria indicate to stop.

With CLEAN_NKEEP > 0, CLEAN will explore the stability of the total clean flux over the last CLEAN_NKEEP iterations. For a positive (resp. negative) source, if the Clean flux becomes smaller (resp. larger) than the Clean flux CLEAN_NKEEP iterations earlier, CLEAN will stop.

Using CLEAN_NKEEP about 70 is a reasonable value. Some special cases (faint extended sources) may require larger values of CLEAN_NKEEP.

9.7.7 HOGBOM CLEAN_POSITIVE

The minimum number of positive components before negative ones are selected.

9.7.8 HOGBOM CLEAN_RESTORE

Fraction of peak response of the primary beams coverage under which the Sky brightness image is blanked in a Mosaic deconvolution.

The default is 0.2.

9.7.9 HOGBOM CLEAN_SEARCH

Fraction of peak response of the primary beams coverage beyond which no Clean component is searched in a Mosaic deconvolution.

The default is 0.2.

9.7.10 HOGBOM CLEAN_SIDELOBE

Minimal relative intensity to consider for fitting the syntheized beam to obtain the Clean beam parameters (MAJOR, MINOR and ANGLE) when 0.

The default is 0.35.

In case of poor UV coverage, CLEAN_SIDELOBE should be higher than the maximum sidelobe level to perform a good Gaussian fit. Some particularly bad UV coverage may not allow any good fit at all, however.

9.7.11 HOGBOM CLEAN_NGOAL

Number of clean components to be selected in a Cycle in the ALMA heterogeneous array cleaning method.

9.7.12 HOGBOM CLEAN_NCYCLE

Maximum number of Major Cycles for the SDI and CLARK methods.

9.7.13 HOGBOM CLEAN_SMOOTH

Smoothing factor between different scales in the MRC and MULTISCALE methods. The default is sqrt(3).

9.7.14 HOGBOM CLEAN_SPEEDY

Speed-up factor for the CLARK major cycles. The default is 1.0. Larger values may be used, but at the expense of possible instabilities of the algorithm.

9.7.15 HOGBOM CLEAN_WORRY

Worry factor in the MULTISCALE method for convergence. It propagates the S/N from one iteration to the other, so that if this S/N degrades, the method stops. Default is 0 (no propagation, and hence no test on S/N). The value should be < 1.0 in all cases.

9.7.16 HOGBOM CLEAN_INFLATE

Maximum Inflation factor for UV_RESTORE (MULTISCALE method). If the number of true (i.e. pixel based) Clean components found by MULTISCALE is larger than CLEAN_INFLATE times the number of compressed (i.e. those

with the smoothing factor information) components, expansion of the compressed components will not be possible, and UV_RESTORE will not be useable.

A default of 50 is in general adequate. Better solutions might be found in the future, and this parameter suppressed. Apart from memory usage, this number has no consequence on the algorithm.

9.7.17 HOGBOM METHOD

Method used for the deconvolution. Can be HOGBOM, MULTI, MRC, SDI or CLARK.

9.7.18 HOGBOM Old_Names:

Some of the CLEAN parameters have kept their old names: MAJOR, MINOR, ANGLE (which are also used by command FIT) BLC, TRC and BEAM_PATCH (which are seldom used)

Others have equivalent short names: ARES, FRES, GAIN, NITER for which the CLEAN_ prefix may be omitted.

9.7.19 HOGBOM BLC

These are the (pixel) coordinates of the Bottom Left Corner of the cleaning box. The actual cleaning support will be the intersection of the specified window with the inner quarter of the map and with any user defined polygon.

9.7.20 HOGBOM TRC

These are the (pixel) coordinates of the Top Right Corner of the cleaning box. The actual cleaning window will be the intersection of the specified window with the inner quarter of the map and with any user defined polygon.

9.7.21 HOGBOM MAJOR

This is the major axis (FWHP) in user coordinates of the Gaussian restoring beam. If 0, the program will fit a Gaussian to the dirty beam. We strongly discourage to change the default value of 0.

9.7.22 HOGBOM MINOR

This is the minor axis (FWHP) in user coordinates of the Gaussian restoring beam. If 0, the program will fit a Gaussian to the dirty beam. We strongly discourage to change the default value of 0.

9.7.23 HOGBOM ANGLE

This is the position angle (from North towards East, i.e. anticlockwise) of the major axis of the Gaussian restoring beam (in degrees). If 0, the program will fit a Gaussian to the dirty beam. We strongly discourage to change the default value of 0.

9.7.24 HOGBOM BEAM_PATCH

The dirty beam patch to be used for the minor cycles in CLARK and MRC method. It should be large enough to avoid doing too many major cycles, but has practically no influence on the result. This size should be specified in pixel units. Reasonable values are between N/8 and N/4, where N is the number of map pixels in the same dimension. If set to N, the CLARK algorithm becomes identical to the HOGBOM algorithm.

9.8 MAP_COMPRESS

[CLEAN\]MAP_COMPRESS WhichOne Nc

Resample (in frequency/velocity) the images (computed by UV_MAP or CLEAN, or loaded by READ WhichOne) by averaging NC adjacent channels.

WhichOne indicates which image must be compressed (DIRTY, CLEAN, SKY). WHichOne = * can be used to treat all the possible images.

9.9 MAP_INTEGRATE

[CLEAN\]MAP_INTEGRATE WhichOne Min Max Type

Compute the integrated intensity map(s) over the specified range from the current image()s (computed by UV_MAP or CLEAN, or loaded by READ WhichOne). Type can be VELOCITY FREQUENCY or CHANNELS.

WhichOne indicates which image must be compressed (DIRTY, CLEAN, SKY). WhichOne = * can be used to treat all the possible images.

9.10 MAP_RESAMPLE

[CLEAN\]MAP_RESAMPLE WhichOne Nc Ref Val Inc

Resample the images (computed by UV_MAP or CLEAN, or loaded by READ WichOne) on a different velocity scale.

Nc new number of channels
Ref New reference pixel
Val New velocity at reference pixel
Inc Velocity increment

WhichOne indicates which image must be compressed (DIRTY, CLEAN, SKY) WHichOne = * can be used to treat all the possible images.

9.11 SPECIFY

[CLEAN\]SPECIFY FREQUENCY|VELOCITY|TELESCOPE Value

SPECIFY FREQUENCY Value

Modify the rest frequency and recompute the velocity scale accordingly. Value is the new rest frequency in MHz

SPECIFY VELOCITY Value

Modify the source velocity and recompute the rest frequency scale accordingly Value is the new velocity in km/s.

SPECIFY TELESCOPE Name

Add or Replace the telescope section with the parameters (name, size, position) for the specified telescope name, essentially to get the most appropriate beam parameter.

A Telescope section is required for MOSAIC. The beamsize will depend on telescope diameter and frequency, with a telescope dependent factor. The default beam size is 1.13 Lambda/D.

9.12 MOSAIC

[CLEAN\]MOSAIC On|Off

Turn on or off the mosaic mode for deconvolution. Note that a READ PRIMARY command, or a UV_MAP command working on a Mosaic UV Table, automatically switches on the mosaic mode. The program prompt changes to inform the user of the current operating mode for deconvolution.

9.13 MRC

[CLEAN\]MRC [FirstPlane [LastPlane]] [/PLOT Clean|Residu] [/FLUX Fmin Fmax] [/QUERY]

Perform a Multi-Resolution CLEAN on the current dirty image. MRC does not support mosaics for theoretical reasons.

Clean the specified plane interval (default: planes between variables FIRST and LAST). If only FirstPlane is specified, Clean only that plane.

If option /PLOT is given, a display of the CLEAN or RESIDUAL map will be shown at each major cycle, depending on the argument (default: Residual). The user will be prompted for continuation when the /QUERY option is present. The cumulative, already cleaned flux is displayed in real time in an additional window while cleaning goes on when the /FLUX option is present. Parameters of the /FLUX option are then used to give the flux limits for this display. A summary plot with the Difference, Smooth, and total CLEANed maps is also displayed.

The user can control the algorithm through SIC variables. New values can be given using "LET VARIABLE value". For ease of use, and whenever it is possible, a sensible value of each parameter will automatically be computed from the context if the value of the corresponding variable is set to its default value, i.e. zero value and empty string. A few variables are initialized to "reasonable" values.

[CLEAN\]CLEAN ?
Will list all main CLEAN_* variables controlling the CLEAN parameters. HELP CLEAN Variables will give a more complete list.

9.13.1 MRC Variables:

```
Basic parameters
CLEAN_GAIN       [       ] Loop gain
CLEAN_NITER      [       ] Maximum number of clean components
CLEAN_FRES       [      %] Maximum value of residual (Fraction of peak)
CLEAN_ARES       [Jy/Beam] Maximum value of residual (Absolute)
CLEAN_POSITIVE   [       ] Minimum number of positive components at start
```

```
CLEAN_NKEEP      [       ] Min number of components before convergence

Old names like in MAPPING
BLC              [  pixel] Bottom left corner of cleaning box
TRC              [  pixel] Top right corner of cleaning box
MAJOR            [ arcsec] Clean beam major axis
MINOR            [ arcsec] Clean beam minor axis
ANGLE            [ degree] Position angle of clean beam
BEAM_PATCH       [  pixel] Size of cleaning beam ** not clear **

Method dependent parameters
CLEAN_INFLATE    [      ] Maximum Inflation factor for UV_RESTORE (MuLTISCAL
CLEAN_NCYCLE     [      ] Max number of Major Cycles (SDI & CLARK methods)
CLEAN_NGOAL      [      ] Max number of comp. in Cycles (ALMA method)
CLEAN_RESTORE    [      ] Threshold for restoring a Mosaic (def 0.2)
CLEAN_SEARCH     [      ] Threshold to search Clean Comp. in a Mosaic (def 0
CLEAN_SIDELOBE   [      ] Min threshold to fit the synthesized beam
CLEAN_SMOOTH     [      ] Smoothing ratio (MRC and MULTISCALE)
CLEAN_SPEEDY     [      ] Speed-up factor (CLARK)
CLEAN_WORRY      [      ] Worry factor (MULTISCALE)
RATIO      [       ] Smoothing factor (default 0: guess, otherwise must be 2
```

9.13.2 MRC CLEAN_ARES

This is the minimal flux in the dirty map that the program will consider as significant. Alternatively, the threshold can be specified as a fraction of the peak flux using CLEAN_FRES. Once this level has been reached the program stops subtracting, and starts the restoration phase. The unit for this parameter is the map unit (typically Jy/Beam). The parameter should usually be of the order of magnitude of the expected noise in the clean map.

If 0, CLEAN_FRES will be used instead. If all of CLEAN_NITER, CLEAN_ARES and CLEAN_FRES are 0, an absolute residual equal to the noise level will be used for CLEAN_ARES.

Short form is ARES.

9.13.3 MRC CLEAN_FRES

This is the minimal fraction of the peak flux in the dirty map that the program will consider as significant. Alternatively, an absolute threshold can be specified using CLEAN_ARES. Once this level has been reached the program stops subtracting, and starts the restoration phase. This parameter is normalized to 1 (neither in % nor in db). It should

usually be of the order of magnitude of the inverse of the expected dynamic range of the intensity.

If 0, CLEAN_ARES will be used instead. If all of CLEAN_NITER, CLEAN_ARES and CLEAN_FRES are 0, an absolute residual equal to the noise level will be used for CLEAN_ARES.

Short form is FRES.

9.13.4 MRC CLEAN_GAIN

This is the gain of the subtraction loop. It should typically be chosen in the range 0.05 and 0.3. Higher values give faster convergence, while lower values give a better restitution of the extended structure. A sensible default is 0.2.

Short form is GAIN.

9.13.5 MRC CLEAN_NITER

This is the maximum number of components the program will accept to subtract. Once it has been reached, the program starts the restoration phase.

If 0, the program will guess a number, based on the image size and maximum signal-to-noise ratio, and specified residual level CLEAN_ARES and/or CLEAN_FRES.

Short form is NITER.

9.13.6 MRC CLEAN_NKEEP

This is an integer specifying the minimum number of Clean components before testing if Cleaning has converged. The convergence is criterium is a comparison of the cumulative flux evolution separated by CLEAN_NKEEP components. If th

IF CLEAN_NKEEP is 0, CLEAN will ignore this convergence criterium, and continue clean until the CLEAN_NITER, CLEAN_ARES or CLEAN_FRES criteria indicate to stop.

With CLEAN_NKEEP > 0, CLEAN will explore the stability of the total clean flux over the last CLEAN_NKEEP iterations. For a positive (resp.

negative) source, if the Clean flux becomes smaller (resp. larger) than the Clean flux CLEAN_NKEEP iterations earlier, CLEAN will stop.

Using CLEAN_NKEEP about 70 is a reasonable value. Some special cases (faint extended sources) may require larger values of CLEAN_NKEEP.

9.13.7 MRC CLEAN_POSITIVE

The minimum number of positive components before negative ones are selected.

9.13.8 MRC CLEAN_RESTORE

Fraction of peak response of the primary beams coverage under which the Sky brightness image is blanked in a Mosaic deconvolution.

The default is 0.2.

9.13.9 MRC CLEAN_SEARCH

Fraction of peak response of the primary beams coverage beyond which no Clean component is searched in a Mosaic deconvolution.

The default is 0.2.

9.13.10 MRC CLEAN_SIDELOBE

Minimal relative intensity to consider for fitting the syntheized beam to obtain the Clean beam parameters (MAJOR, MINOR and ANGLE) when 0. The default is 0.35.

In case of poor UV coverage, CLEAN_SIDELOBE should be higher than the maximum sidelobe level to perform a good Gaussian fit. Some particularly bad UV coverage may not allow any good fit at all, however.

9.13.11 MRC CLEAN_NGOAL

Number of clean components to be selected in a Cycle in the ALMA het-

erogeneous array cleaning method.

9.13.12 MRC CLEAN_NCYCLE

Maximum number of Major Cycles for the SDI and CLARK methods.

9.13.13 MRC CLEAN_SMOOTH

Smoothing factor between different scales in the MRC and MULTISCALE methods. The default is sqrt(3).

9.13.14 MRC CLEAN_SPEEDY

Speed-up factor for the CLARK major cycles. The default is 1.0. Larger values may be used, but at the expense of possible instabilities of the algorithm.

9.13.15 MRC CLEAN_WORRY

Worry factor in the MULTISCALE method for convergence. It propagates the S/N from one iteration to the other, so that if this S/N degrades, the method stops. Default is 0 (no propagation, and hence no test on S/N). The value should be < 1.0 in all cases.

9.13.16 MRC CLEAN_INFLATE

Maximum Inflation factor for UV_RESTORE (MULTISCALE method). If the number of true (i.e. pixel based) Clean components found by MULTISCALE is larger than CLEAN_INFLATE times the number of compressed (i.e. those with the smoothing factor information) components, expansion of the compressed components will not be possible, and UV_RESTORE will not be useable.

A default of 50 is in general adequate. Better solutions might be found in the future, and this parameter suppressed. Apart from memory usage, this number has no consequence on the algorithm.

9.13.17 MRC METHOD

Method used for the deconvolution. Can be HOGBOM, MULTI, MRC, SDI or CLARK.

9.13.18 MRC Old_Names:

Some of the CLEAN parameters have kept their old names: MAJOR, MINOR, ANGLE (which are also used by command FIT) BLC, TRC and BEAM_PATCH (which are seldom used)

Others have equivalent short names: ARES, FRES, GAIN, NITER for which the CLEAN_ prefix may be omitted.

9.13.19 MRC BLC

These are the (pixel) coordinates of the Bottom Left Corner of the cleaning box. The actual cleaning support will be the intersection of the specified window with the inner quarter of the map and with any user defined polygon.

9.13.20 MRC TRC

These are the (pixel) coordinates of the Top Right Corner of the cleaning box. The actual cleaning window will be the intersection of the specified window with the inner quarter of the map and with any user defined polygon.

9.13.21 MRC MAJOR

This is the major axis (FWHP) in user coordinates of the Gaussian restoring beam. If 0, the program will fit a Gaussian to the dirty beam. We strongly discourage to change the default value of 0.

9.13.22 MRC MINOR

This is the minor axis (FWHP) in user coordinates of the Gaussian

restoring beam. If 0, the program will fit a Gaussian to the dirty beam. We strongly discourage to change the default value of 0.

9.13.23 MRC ANGLE

This is the position angle (from North towards East, i.e. anticlockwise) of the major axis of the Gaussian restoring beam (in degrees). If 0, the program will fit a Gaussian to the dirty beam. We strongly discourage to change the default value of 0.

9.13.24 MRC BEAM_PATCH

The dirty beam patch to be used for the minor cycles in CLARK and MRC method. It should be large enough to avoid doing too many major cycles, but has practically no influence on the result. This size should be specified in pixel units. Reasonable values are between N/8 and N/4, where N is the number of map pixels in the same dimension. If set to N, the CLARK algorithm becomes identical to the HOGBOM algorithm.

9.13.25 MRC RATIO

Used smoothing factor, which must be a power of 2. The default is 0, to indicate that the actual value must be estimated from the image size.

9.14 MULTI

[CLEAN\]MULTI [FirstPlane [LastPlane]] [/FLUX Fmin Fmax]

Multiscale CLEAN algorithm.

Clean the specified plane interval (default: Planes between variables FIRST and LAST). The cumulative, already cleaned flux is displayed in real time in an additional window while cleaning goes on when the /FLUX option is present. Parameters of the /FLUX option are then used to give the flux limits for this display.

MULTI does not yet work on mosaics.

The user can control the algorithm through SIC variables. New values can be given using "LET VARIABLE value". For ease of use, and whenever it is

```
possible,  a sensible value of each parameter will automatically be com-
puted from the context if the value of the corresponding variable is set
to  its default value, i.e. zero value and empty string. A few variables
are initialized to "reasonable" values.

    [CLEAN\]CLEAN ?
Will list all main CLEAN_* variables controlling the  CLEAN  parameters.
HELP CLEAN Variables will give a more complete list.
```

9.14.1 MULTI Variables:

```
Basic parameters
CLEAN_GAIN       [       ] Loop gain
CLEAN_NITER      [       ] Maximum number of clean components
CLEAN_FRES       [      %] Maximum value of residual (Fraction of peak)
CLEAN_ARES       [Jy/Beam] Maximum value of residual (Absolute)
CLEAN_POSITIVE   [       ] Minimum number of positive components at start
CLEAN_NKEEP      [       ] Min number of components before convergence

Old names like in MAPPING
BLC              [  pixel] Bottom left corner of cleaning box
TRC              [  pixel] Top right corner of cleaning box
MAJOR            [ arcsec] Clean beam major axis
MINOR            [ arcsec] Clean beam minor axis
ANGLE            [ degree] Position angle of clean beam
BEAM_PATCH       [  pixel] Size of cleaning beam ** not clear **

Method dependent parameters
CLEAN_INFLATE    [      ] Maximum Inflation factor for UV_RESTORE (MuLTISCAL
CLEAN_NCYCLE     [      ] Max number of Major Cycles (SDI & CLARK methods)
CLEAN_NGOAL      [      ] Max number of comp. in Cycles (ALMA method)
CLEAN_RESTORE    [      ] Threshold for restoring a Mosaic (def 0.2)
CLEAN_SEARCH     [      ] Threshold to search Clean Comp. in a Mosaic (def 0
CLEAN_SIDELOBE   [      ] Min threshold to fit the synthesized beam
CLEAN_SMOOTH     [      ] Smoothing ratio (MRC and MULTISCALE)
CLEAN_SPEEDY     [      ] Speed-up factor (CLARK)
CLEAN_WORRY      [      ] Worry factor (MULTISCALE)
CLEAN_SMOOTH     [       ] Smoothing factor (default sqrt(3))
```

9.14.2 MULTI CLEAN_ARES

```
This is the minimal flux in the dirty map that the program will consider
as significant.  Alternatively, the threshold  can  be  specified  as  a
fraction  of  the  peak flux using CLEAN_FRES.  Once this level has been
reached the program stops subtracting, and starts the restoration phase.
The  unit  for  this parameter is the map unit (typically Jy/Beam).  The
```

parameter should usually be of the order of magnitude of the expected noise in the clean map.

If 0, CLEAN_FRES will be used instead. If all of CLEAN_NITER, CLEAN_ARES and CLEAN_FRES are 0, an absolute residual equal to the noise level will be used for CLEAN_ARES.

Short form is ARES.

9.14.3 MULTI CLEAN_FRES

This is the minimal fraction of the peak flux in the dirty map that the program will consider as significant. Alternatively, an absolute threshold can be specified using CLEAN_ARES. Once this level has been reached the program stops subtracting, and starts the restoration phase. This parameter is normalized to 1 (neither in % nor in db). It should usually be of the order of magnitude of the inverse of the expected dynamic range of the intensity.

If 0, CLEAN_ARES will be used instead. If all of CLEAN_NITER, CLEAN_ARES and CLEAN_FRES are 0, an absolute residual equal to the noise level will be used for CLEAN_ARES.

Short form is FRES.

9.14.4 MULTI CLEAN_GAIN

This is the gain of the subtraction loop. It should typically be chosen in the range 0.05 and 0.3. Higher values give faster convergence, while lower values give a better restitution of the extended structure. A sensible default is 0.2.

Short form is GAIN.

9.14.5 MULTI CLEAN_NITER

This is the maximum number of components the program will accept to subtract. Once it has been reached, the program starts the restoration phase.

If 0, the program will guess a number, based on the image size and maximum signal-to-noise ratio, and specified residual level CLEAN_ARES and/or CLEAN_FRES.

Short form is NITER.

9.14.6 MULTI CLEAN_NKEEP

This is an integer specifying the minimum number of Clean components before testing if Cleaning has converged. The convergence is criterium is a comparison of the cumulative flux evolution separated by CLEAN_NKEEP components. If th

IF CLEAN_NKEEP is 0, CLEAN will ignore this convergence criterium, and continue clean until the CLEAN_NITER, CLEAN_ARES or CLEAN_FRES criteria indicate to stop.

With CLEAN_NKEEP > 0, CLEAN will explore the stability of the total clean flux over the last CLEAN_NKEEP iterations. For a positive (resp. negative) source, if the Clean flux becomes smaller (resp. larger) than the Clean flux CLEAN_NKEEP iterations earlier, CLEAN will stop.

Using CLEAN_NKEEP about 70 is a reasonable value. Some special cases (faint extended sources) may require larger values of CLEAN_NKEEP.

9.14.7 MULTI CLEAN_POSITIVE

The minimum number of positive components before negative ones are selected.

9.14.8 MULTI CLEAN_RESTORE

Fraction of peak response of the primary beams coverage under which the Sky brightness image is blanked in a Mosaic deconvolution.

The default is 0.2.

9.14.9 MULTI CLEAN_SEARCH

Fraction of peak response of the primary beams coverage beyond which no Clean component is searched in a Mosaic deconvolution.

The default is 0.2.

9.14.10 MULTI CLEAN_SIDELOBE

Minimal relative intensity to consider for fitting the syntheized beam to obtain the Clean beam parameters (MAJOR, MINOR and ANGLE) when 0. The default is 0.35.

In case of poor UV coverage, CLEAN_SIDELOBE should be higher than the maximum sidelobe level to perform a good Gaussian fit. Some particularly bad UV coverage may not allow any good fit at all, however.

9.14.11 MULTI CLEAN_NGOAL

Number of clean components to be selected in a Cycle in the ALMA heterogeneous array cleaning method.

9.14.12 MULTI CLEAN_NCYCLE

Maximum number of Major Cycles for the SDI and CLARK methods.

9.14.13 MULTI CLEAN_SMOOTH

Smoothing factor between different scales in the MRC and MULTISCALE methods. The default is sqrt(3).

9.14.14 MULTI CLEAN_SPEEDY

Speed-up factor for the CLARK major cycles. The default is 1.0. Larger values may be used, but at the expense of possible instabilities of the algorithm.

9.14.15 MULTI CLEAN_WORRY

Worry factor in the MULTISCALE method for convergence. It propagates the S/N from one iteration to the other, so that if this S/N degrades, the method stops. Default is 0 (no propagation, and hence no test on S/N). The value should be < 1.0 in all cases.

9.14.16 MULTI CLEAN_INFLATE

Maximum Inflation factor for UV_RESTORE (MULTISCALE method). If the number of true (i.e. pixel based) Clean components found by MULTISCALE is larger than CLEAN_INFLATE times the number of compressed (i.e. those with the smoothing factor information) components, expansion of the compressed components will not be possible, and UV_RESTORE will not be useable.

A default of 50 is in general adequate. Better solutions might be found in the future, and this parameter suppressed. Apart from memory usage, this number has no consequence on the algorithm.

9.14.17 MULTI METHOD

Method used for the deconvolution. Can be HOGBOM, MULTI, MRC, SDI or CLARK.

9.14.18 MULTI Old_Names:

Some of the CLEAN parameters have kept their old names: MAJOR, MINOR, ANGLE (which are also used by command FIT) BLC, TRC and BEAM_PATCH (which are seldom used)

Others have equivalent short names: ARES, FRES, GAIN, NITER for which the CLEAN_ prefix may be omitted.

9.14.19 MULTI BLC

These are the (pixel) coordinates of the Bottom Left Corner of the cleaning box. The actual cleaning support will be the intersection of the specified window with the inner quarter of the map and with any user defined polygon.

9.14.20 MULTI TRC

These are the (pixel) coordinates of the Top Right Corner of the cleaning box. The actual cleaning window will be the intersection of the specified window with the inner quarter of the map and with any user de-

fined polygon.

9.14.21 MULTI MAJOR

This is the major axis (FWHP) in user coordinates of the Gaussian restoring beam. If 0, the program will fit a Gaussian to the dirty beam. We strongly discourage to change the default value of 0.

9.14.22 MULTI MINOR

This is the minor axis (FWHP) in user coordinates of the Gaussian restoring beam. If 0, the program will fit a Gaussian to the dirty beam. We strongly discourage to change the default value of 0.

9.14.23 MULTI ANGLE

This is the position angle (from North towards East, i.e. anticlockwise) of the major axis of the Gaussian restoring beam (in degrees). If 0, the program will fit a Gaussian to the dirty beam. We strongly discourage to change the default value of 0.

9.14.24 MULTI BEAM_PATCH

The dirty beam patch to be used for the minor cycles in CLARK and MRC method. It should be large enough to avoid doing too many major cycles, but has practically no influence on the result. This size should be specified in pixel units. Reasonable values are between N/8 and N/4, where N is the number of map pixels in the same dimension. If set to N, the CLARK algorithm becomes identical to the HOGBOM algorithm.

9.14.25 MULTI SMOOTH

Smoothing ratio between the different scales. The default is sqrt(3), but larger values should be used for large images with wide spatial dynamic range.

9.15 MX

[CLEAN\]MX [FirstPlane [LastPlane]] [/PLOT Clean|Residu] [/FLUX Fmin Fmax] [/QUERY]

Make and deconvolve maps starting from a UV table. It combines UV_MAP and CLEAN in a single step.

The mapping process is identical to UV_MAP. It makes a map from UV data by griding the UV data using a convolving function, and then Fast Fourier Transforming the individual channels. However, MX always produces a single beam for all channels, thus neglecting frequency change between channels. MX enables to shift the map center and rotate the image, by shifting the phase tracking center and rotating the UV coordinates of the input UV table.

The CLEAN algorithm is similar to the CLARK method, but with major cycles operating directly on the ungridded UV table rather than in the image plane. Accordingly, aliasing affects only of the residuals, not the clean components. It is thus more accurate but also slower than CLARK as it asks for the gridding step at each major cycle. MX also shares the same limitation as CLARK on large sidelobes.

The user can control the algorithm through SIC variables. New values can be given using "LET VARIABLE value". For ease of use, and whenever it is possible, a sensible value of each parameter will automatically be computed from the context if the value of the corresponding variable is set to its default value, i.e. zero value and empty string. A few variables are initialized to "reasonable" values.

[CLEAN\]CLEAN ?
Will list all main CLEAN_* variables controlling the CLEAN parameters. HELP CLEAN Variables will give a more complete list.

[CLEAN\]UV_MAP ?
Will list all MAP_* variables controlling the UV_MAP parameters.

The list of control variables is (by alphabetic order, with the old names used by Mapping on the right)

New names	[unit]	-- Description -- % Old Name
MAP_BEAM_STEP	[ ]	Number of channels per single dirty beam
MAP_CELL	[arcsec]	Image pixel size
MAP_CENTER	[string]	RA, Dec of map center, and Position Angle
MAP_CONVOLUTION	[ ]	Convolution function % CONVOLUTION
MAP_FIELD	[arcsec]	Map field of view
MAP_POWER	[ ]	Maximum exponent of 3 and 5 allowed in MAP_SIZE
MAP_PRECIS	[ ]	Fraction of pixel tolerance on beam matching
MAP_ROBUST	[ ]	Robustness factor % UV_CELL[2]

```
MAP_ROUNDING    [       ]  Precision of MAP_SIZE
MAP_SIZE        [       ]  Number of pixels
MAP_TAPEREXPO   [       ]  Taper exponent           % TAPER_EXPO
MAP_TRUNCATE    [      %]  Mosaic truncation level
MAP_UVCELL      [      m]  UV cell size             % UV_CELL[1]
MAP_UVTAPER     [m,m,deg]  Gaussian taper           % UV_TAPER
MAP_VERSION     [       ]  Code version (0 new, -1 old)

NAME is no longer used, and WEIGHT_MODE is obsolete.
MAP_RA          [  hours]  RA of map center
MAP_DEC         [    deg]  Dec of map center
MAP_ANGLE       [    deg]  Map position angle
MAP_SHIFT       [Yes/No ]  Shift phase center
are  obsolescent,  superseded  by MAP_CENTER. They are provided only for
compatibility with older scripts.
```

9.15.1 MX Variables:

```
Basic parameters
CLEAN_GAIN       [       ] Loop gain
CLEAN_NITER      [       ] Maximum number of clean components
CLEAN_FRES       [      %] Maximum value of residual (Fraction of peak)
CLEAN_ARES       [Jy/Beam] Maximum value of residual (Absolute)
CLEAN_POSITIVE   [       ] Minimum number of positive components at start
CLEAN_NKEEP      [       ] Min number of components before convergence

Old names like in MAPPING
BLC              [  pixel] Bottom left corner of cleaning box
TRC              [  pixel] Top right corner of cleaning box
MAJOR            [ arcsec] Clean beam major axis
MINOR            [ arcsec] Clean beam minor axis
ANGLE            [ degree] Position angle of clean beam
BEAM_PATCH       [  pixel] Size of cleaning beam ** not clear **

Method dependent parameters
CLEAN_INFLATE    [      ] Maximum Inflation factor for UV_RESTORE (MuLTISCAL
CLEAN_NCYCLE     [      ] Max number of Major Cycles (SDI & CLARK methods)
CLEAN_NGOAL      [      ] Max number of comp. in Cycles (ALMA method)
CLEAN_RESTORE    [      ] Threshold for restoring a Mosaic (def 0.2)
CLEAN_SEARCH     [      ] Threshold to search Clean Comp. in a Mosaic (def 0
CLEAN_SIDELOBE   [      ] Min threshold to fit the synthesized beam
CLEAN_SMOOTH     [      ] Smoothing ratio (MRC and MULTISCALE)
CLEAN_SPEEDY     [      ] Speed-up factor (CLARK)
CLEAN_WORRY      [      ] Worry factor (MULTISCALE)
```

9.15.2 MX MAP_BEAM_STEP

```
MAP_BEAM_STEP   Integer
```

Number of channels per synthesized beam plane.

Default is 0, meaning only 1 beam plane for all channels. N (>0) indicates N consecutive channels will share the same dirty beam.

A value of -1 can be used to compute the number of channels per beam plane to ensure the angular scale does not deviate more than a fraction of the map cell at the map edge. This fraction is controlled by variable MAP_PRECIS (default 0.1)

9.15.3 MX MAP_CELL

```
MAP_CELL[2]    Real
```

The map pixel size [arcsec]. It is recommended to use identical values in X and Y. A sampling of at least 3 pixel per beam is recommended to ease the deconvolution. Enter 0,0 to let the task find the best values.

9.15.4 MX MAP_CENTER

```
MAP_CENTER     Character String
```

Specify the Map center and orientation in the same way as the arguments of UV_MAP.

9.15.5 MX MAP_CONVOLUTION

```
MAP_CONVOLUTION    Integer
```

Select the desired convolution function for gridding in the UV plane Choices are

```
0    Default (currently 5)
1    Boxcar
2    Gaussian
3    Sin(x)/x
4    Gaussian * Sin(x)/x
5    Spheroidal
```

Spheroidal functions is the optimal choice. So we strongly discourage

use of any other convolution function, which are here for tests only.

9.15.6 MX MAP_FIELD

MAP_FIELD[2] Real

Field of view in X and Y in arcsec. The field of view MAP_FIELD has precedence over the number of pixels MAP_SIZE to define the actual map size when both are non-zero.

9.15.7 MX MAP_POWER

MAP_POWER[2] Integer

Maximum exponent of 3 and 5 allowed in automatic guess of MAP_SIZE. MAP_SIZE is decomposed in 2^k 3^p 5^q, and p and q must be less or equal to MAP_POWER.

Default is 0: MAP_SIZE is just a power of 2. A value of 1 allows approximation of any map size to 20 %, while a value of 2 allows 10 % approximation. Fast Fourier Transform are slightly slower with powers of 3 and 5, but limiting the map size can gain a lot in the Cleaning process (which can scale as MAP_SIZE^4).

9.15.8 MX MAP_PRECIS

MAP_PRECIS Real

Maximum mismatch in pixel at map edge between the true synthesized beam (which would have been computed using the exact channel frequency) and the computed synthesized beam with the mean frequency of the channels sharing the same beam. This is used (with the actual image size) to derive the actual number of channels which can share the same beam, i.e. the effective value of MAP_BEAM_STEP when MAP_BEAM_STEP is -1.

Default is 0.1

9.15.9 MX MAP_ROBUST

MAP_ROBUST Real

Robust weighting factor. A number between 0 and +infty.

Robust weighting gives the natural weight to UV cells whose natural weight is lower than a given threshold. In contrast, if the natural weight of the UV cell is larger than this threshold, the weight is set to this (uniform) threshold. The UV cell size is defined by MAP_UVCELL and the threshold value is in MAP_ROBUST.

0 means natural weighting, which is optimal for point sources. The Robust weighting factor controls the resolution: better resolution is obtained for small values (at the expense of noise), resolution approaching the natural weighting scheme for large values. Larger UV cell size give higher angular resolution (but again more noise).

MAP_ROBUST around .5 to 1 is a good compromise between noise increase and angular resolution.

9.15.10 MX MAP_ROUNDING

MAP_ROUNDING Real

Maximum error between optimal size (MAP_FIELD / MAP_CELL) and rounded (as a power of 2^k 3^p 5^q) MAP_SIZE to round by floor (thus limiting the field of view), instead of ceiling (which guarantees a larger field of view, but leads to bigger images).

Default is 0.05.

9.15.11 MX MAP_SHIFT

MAP_SHIFT Logical

Obsolescent, superseded by MAP_CENTER, or the UV_MAP arguments.

Logical variable indicating whether map center (i.e. phase tracking center) or orientation should be changed.

9.15.12 MX MAP_SIZE

MAP_SIZE[2] Integer

Number of pixels in X and Y. It should preferentially be a power of two, (although this is not strictly required) to speed-up the FFT. MAP_SIZE*MAP_CELL should be at least twice the size of the field-of-view (primary beam size for a single field). Enter 0,0 to let the command find a sensible map size.

MAP_SIZE is not used if MAP_FIELD is non zero.

Odd values are forbidden.

Default is 0,0, i.e. UV_MAP will guess the most appropriate values which depend on MAP_ROUNDING and MAP_POWER.

9.15.13 MX MAP_TAPEREXPO

MAP_TAPEREXPO Real

Taper exponent. The default is 2 (indicating a Gaussian) but smoother or sharper functions can be used. 1 would give an Exponential, 4 would be getting close to square profile...

9.15.14 MX MAP_TRUNCATE

MAP_TRUNCATE Real

Mosaic truncation level in PerCent. Default value is 0.2. Current value can be overriden by option /TRUNCATE in commands UV_MAP or PRIMARY.

9.15.15 MX MAP_UVTAPER

MAP_UVTAPER[3] Real

Parameters of the tapering function (Gaussian if MAP_TAPEREXPO = 2): major axis at 1/e level [m], minor axis at 1/e level [m], and position angle [deg].

9.15.16 MX MAP_UVCELL

MAP_UVCELL Real

UV cell size for robust weighting [m]. Should be of the order of half the dish diameter (7.5 m for PdBI), or smaller or even larger. It controls the beam shape in Robust weighting.

9.15.17 MX MAP_VERSION

MAP_VERSION Integer

[EXPERT Only] Code indicating which version of the UV_MAP and UV_RESTORE algorithm should be used. 0 is optimal. -1 is the "historical" (pre-2016) version. 1 is an intermediate version used during multi-frequency beams development.

9.15.18 MX MCOL

MCOL[2] Integer

First and Last channel to image. Values of 0 mean imaging all the planes.

9.15.19 MX WCOL

WCOL Integer

[Obsolescent] The channel from which the weight should be taken. WCOL set to 0 means using a default channel. WCOL has no real meaning in all cases where more than one beam is computed for all channels.

9.15.20 MX Old_Names:

NAME	[ ]	Label of the dirty image and beam plots
UV_TAPER	[m,m,deg]	UV-apodization by convolution with a Gaussian
WEIGHT_MODE	[ ]	Weighting mode (NA|UN)
UV_CELL	[m, ??]	UV cell size and threshold for Robust weighting
MAP_FIELD	[arcsec]	Map field of view
MAP_CELL	[arcsec]	Map cell size
MAP_SIZE	[pixels]	Map size in pixels (if MAP_FIELD is zero)
MCOL	[ ]	First and Last channel to map
WCOL	[ ]	Channel from which the weights are taken
CONVOLUTION	[ ]	Convolution function (5)
UV_SHIFT	[ ]	Change the map phase center or map orientation?

```
MAP_RA       [       ]  RA of map phase center
MAP_DEC      [       ]  Dec of map phase center
MAP_ANGLE    [    deg]  Map position angle
MAP_BEAM_STEP [     ]  Number of channels per synthesized beam plane
```

9.15.21 MX convolution

```
Older variable name for MAP_CONVOLUTION
```

9.15.22 MX map_angle

```
MAP_ANGLE      Real
```

Position Angle of the direction which will become the apparent North in the map. Used only if UV_SHIFT is YES.

Superseded by MAP_CENTER.

9.15.23 MX map_dec

```
MAP_DEC     Real
```

Dec of map center. Used only if UV_SHIFT is YES.

Superseded by MAP_CENTER.

9.15.24 MX map_ra

```
MAP_RA      Real
```

RA of map center. Used only if UV_SHIFT is YES.

Superseded by MAP_CENTER.

9.15.25 MX uv_cell

```
Older variables for MAP_UVCELL (uv
```

9.15.26 MX uv_shift

Older variable name of MAP_SHIFT (this one is also obsolescent)

9.15.27 MX uv_taper

Older variable name of MAP_UVTAPER

9.15.28 MX taper_expo

Older variable name for MAP_TAPEREXPO

9.15.29 MX weight_mode

weightde Character

Weighting mode: Natural (optimum in terms of sensitivity) or robust (usually lower sidelobes and higher spatial resolution) weighting. This was needed in Mapping to toggle between Natural and Robust weighting, while IMAGER does that based on MAP_ROBUST value.

9.15.30 MX CLEAN_ARES

This is the minimal flux in the dirty map that the program will consider as significant. Alternatively, the threshold can be specified as a fraction of the peak flux using CLEAN_FRES. Once this level has been reached the program stops subtracting, and starts the restoration phase. The unit for this parameter is the map unit (typically Jy/Beam). The parameter should usually be of the order of magnitude of the expected noise in the clean map.

If 0, CLEAN_FRES will be used instead. If all of CLEAN_NITER, CLEAN_ARES and CLEAN_FRES are 0, an absolute residual equal to the noise level will be used for CLEAN_ARES.

Short form is ARES.

9.15.31 MX CLEAN_FRES

This is the minimal fraction of the peak flux in the dirty map that the program will consider as significant. Alternatively, an absolute threshold can be specified using CLEAN_ARES. Once this level has been reached the program stops subtracting, and starts the restoration phase. This parameter is normalized to 1 (neither in % nor in db). It should usually be of the order of magnitude of the inverse of the expected dynamic range of the intensity.

If 0, CLEAN_ARES will be used instead. If all of CLEAN_NITER, CLEAN_ARES and CLEAN_FRES are 0, an absolute residual equal to the noise level will be used for CLEAN_ARES.

Short form is FRES.

9.15.32 MX CLEAN_GAIN

This is the gain of the subtraction loop. It should typically be chosen in the range 0.05 and 0.3. Higher values give faster convergence, while lower values give a better restitution of the extended structure. A sensible default is 0.2.

Short form is GAIN.

9.15.33 MX CLEAN_NITER

This is the maximum number of components the program will accept to subtract. Once it has been reached, the program starts the restoration phase.

If 0, the program will guess a number, based on the image size and maximum signal-to-noise ratio, and specified residual level CLEAN_ARES and/or CLEAN_FRES.

Short form is NITER.

9.15.34 MX CLEAN_NKEEP

This is an integer specifying the minimum number of Clean components before testing if Cleaning has converged. The convergence is criterium is a comparison of the cumulative flux evolution separated by CLEAN_NKEEP components. If th

IF CLEAN_NKEEP is 0, CLEAN will ignore this convergence criterium, and continue clean until the CLEAN_NITER, CLEAN_ARES or CLEAN_FRES criteria indicate to stop.

With CLEAN_NKEEP > 0, CLEAN will explore the stability of the total clean flux over the last CLEAN_NKEEP iterations. For a positive (resp. negative) source, if the Clean flux becomes smaller (resp. larger) than the Clean flux CLEAN_NKEEP iterations earlier, CLEAN will stop.

Using CLEAN_NKEEP about 70 is a reasonable value. Some special cases (faint extended sources) may require larger values of CLEAN_NKEEP.

9.15.35 MX CLEAN_POSITIVE

The minimum number of positive components before negative ones are selected.

9.15.36 MX CLEAN_RESTORE

Fraction of peak response of the primary beams coverage under which the Sky brightness image is blanked in a Mosaic deconvolution.

The default is 0.2.

9.15.37 MX CLEAN_SEARCH

Fraction of peak response of the primary beams coverage beyond which no Clean component is searched in a Mosaic deconvolution.

The default is 0.2.

9.15.38 MX CLEAN_SIDELOBE

Minimal relative intensity to consider for fitting the syntheized beam to obtain the Clean beam parameters (MAJOR, MINOR and ANGLE) when 0.

The default is 0.35.

In case of poor UV coverage, CLEAN_SIDELOBE should be higher than the maximum sidelobe level to perform a good Gaussian fit. Some particularly bad UV coverage may not allow any good fit at all, however.

9.15.39 MX CLEAN_NGOAL

Number of clean components to be selected in a Cycle in the ALMA heterogeneous array cleaning method.

9.15.40 MX CLEAN_NCYCLE

Maximum number of Major Cycles for the SDI and CLARK methods.

9.15.41 MX CLEAN_SMOOTH

Smoothing factor between different scales in the MRC and MULTISCALE methods. The default is sqrt(3).

9.15.42 MX CLEAN_SPEEDY

Speed-up factor for the CLARK major cycles. The default is 1.0. Larger values may be used, but at the expense of possible instabilities of the algorithm.

9.15.43 MX CLEAN_WORRY

Worry factor in the MULTISCALE method for convergence. It propagates the S/N from one iteration to the other, so that if this S/N degrades, the method stops. Default is 0 (no propagation, and hence no test on S/N). The value should be < 1.0 in all cases.

9.15.44 MX CLEAN_INFLATE

Maximum Inflation factor for UV_RESTORE (MULTISCALE method). If the number of true (i.e. pixel based) Clean components found by MULTISCALE is larger than CLEAN_INFLATE times the number of compressed (i.e. those

with the smoothing factor information) components, expansion of the compressed components will not be possible, and UV_RESTORE will not be useable.

A default of 50 is in general adequate. Better solutions might be found in the future, and this parameter suppressed. Apart from memory usage, this number has no consequence on the algorithm.

9.15.45 MX METHOD

Method used for the deconvolution. Can be HOGBOM, MULTI, MRC, SDI or CLARK.

9.15.46 MX Old_Names:

Some of the CLEAN parameters have kept their old names: MAJOR, MINOR, ANGLE (which are also used by command FIT) BLC, TRC and BEAM_PATCH (which are seldom used)

Others have equivalent short names: ARES, FRES, GAIN, NITER for which the CLEAN_ prefix may be omitted.

9.15.47 MX BLC

These are the (pixel) coordinates of the Bottom Left Corner of the cleaning box. The actual cleaning support will be the intersection of the specified window with the inner quarter of the map and with any user defined polygon.

9.15.48 MX TRC

These are the (pixel) coordinates of the Top Right Corner of the cleaning box. The actual cleaning window will be the intersection of the specified window with the inner quarter of the map and with any user defined polygon.

9.15.49 MX MAJOR

This is the major axis (FWHP) in user coordinates of the Gaussian restoring beam. If 0, the program will fit a Gaussian to the dirty beam. We strongly discourage to change the default value of 0.

9.15.50 MX MINOR

This is the minor axis (FWHP) in user coordinates of the Gaussian restoring beam. If 0, the program will fit a Gaussian to the dirty beam. We strongly discourage to change the default value of 0.

9.15.51 MX ANGLE

This is the position angle (from North towards East, i.e. anticlockwise) of the major axis of the Gaussian restoring beam (in degrees). If 0, the program will fit a Gaussian to the dirty beam. We strongly discourage to change the default value of 0.

9.15.52 MX BEAM_PATCH

The dirty beam patch to be used for the minor cycles in CLARK and MRC method. It should be large enough to avoid doing too many major cycles, but has practically no influence on the result. This size should be specified in pixel units. Reasonable values are between N/8 and N/4, where N is the number of map pixels in the same dimension. If set to N, the CLARK algorithm becomes identical to the HOGBOM algorithm.

9.16 PRIMARY

[CLEAN\]PRIMARY [BeamSize] [/TRUNCATE Percent]

Apply approximate primary beam correction to a single field deconvolved (CLEAN) image, in order to create the sky brightness image (named SKY). The primary beam model is a simple Gaussian.

If BeamSize (in radian) is specified, uses the corresponding half-power beam size to determine the Gaussian beam.

If not, the parameters are taken from the telescope parameters, as found in the telescope section of the CLEAN image and the observing frequency from the CLEAN image (e.g. for ALMA it uses 1.13 Lambda/D).

The SKY image is written with extension .lmv-sky by command WRITE.

9.16.1 PRIMARY /TRUNCATE

[CLEAN\]PRIMARY [BeamSize] /TRUNCATE Percent

Specify the truncation level. Default (in % of peak beam response) is given by the MAP_TRUNCATE variable. Values are blanked beyond this.

9.17 READ

[CLEAN\]READ Buffer File [/COMPACT] |/FREQUENCY RestFreq] [/RANGE Min Max Type] [/NOTRAIL]

Read the specified internal buffer (BEAM, CCT, CGAINS, CLEAN, DIRTY, FIELDS, MASK, MODEL, PRIMARY, RESIDUAL, SKY, SUPPORT, SINGLEDISH, UV) from input File. Default extension is .uvt for UV, CGAINS and MODEL, and for the others, in order, .beam, .cct, .lmv-clean, .lmv, .msk, .lobe, .lmv-res, be indicated through the /RANGE option, even for UV tables.

The corresponding buffer is available as a SIC image-like variable of the same name, so that one can use e.g. HEADER DIRTY command.

READ * Name will attempt to read all existing files of same Name with the standard file types corresponding to the respective buffers.

The MODEL UV table is for use in conjunction with commands in the CALIBRATE\ language.

The /COMPACT option is used to load the ACA-specific internal buffer used in the ALMA joint deconvolution method.

The /NOTRAIL allows to ignore trailing columns (normally not recommended. Used for debug only).

9.17.1 READ Optimisation

Reading can be lengthy, especially for ALMA data. Mapping attempts to minimize read operations by checking if anything has changed since the last command. This capability is enable if the SIC variable MAPPING_OPTIMIZE is non zero, disabled otherwise.

Currently, the READ command always display a message about what read-

ing may have been skipped, and whether this possible optimization has been overridden by user choice.

9.17.2 READ /COMPACT

[CLEAN\]READ Buffer File /COMPACT [/RANGE Min Max Type]

Read the specified internal buffer (UV, MODEL, BEAM, PRIMARY, DIRTY, CLEAN, MASK, CCT) from input File to the "compact array" data area.

9.17.3 READ /FREQUENCY

[CLEAN\]READ Buffer File /FREQUENCY RestFreq [/RANGE Min Max Type]

Read the specified internal buffer and reset the velocity scale to the corresponding rest frequencies. Velocities specified in the /RANGE Min Max VELOCITY option would then refer to this new frequency.

9.17.4 READ /NOTRAIL

[CLEAN\]READ Buffer File /NOTRAIL [/FREQUENCY Freq] [/RANGE Min Max Type]

When reading UV Tables, ignores any trailing column. Trailing columns normally appear for mosaics. However, ALMA sometimes uses known proper motions to shift (by small amounts) phase centers between two observing periods, yielding pseudo-mosaics with tiny (in general insignificant) displacements. The /NOTRAIL option allows to ignore these details.

9.17.5 READ /PLANES

[CLEAN\]READ Buffer File /PLANES First Last

** OBSOLESCENT ** Use /RANGE for a simpler interface.

Read only "channels" between First and Last. For UV tables with more than 1 Stokes parameter, which are **NOT** fully supported by IMAGER, the meaning of "channel" is ambiguous.

9.17.6 READ /RANGE

[CLEAN\]READ Buffer File /RANGE Min Max Type

Load only the channels between the First and Last defined by Min Max and

Type. Type can be CHANNEL, VELOCITY or FREQUENCY.

For type CHANNEL, Min and Max indicate offsets from Channel 1 and Channel Nchan (the number of channels in the data set). Thus Max can be negative: it then indicates Last = Nchan-Max. Also Min=0 and Max=0 implies loading all the channels.

9.17.7 READ SINGLE

[CLEAN\]READ SINGLE File[.ext] [/RANGE Min Max Type]

Read the "Single Dish" data set. It can be a Class table (.tab), or a 3-D data cube (.lmv).

The /RANGE option only works for a .lmv data cube so far.

9.18 SDI

[CLEAN\]SDI [FirstPlane [LastPlane]] [/PLOT Clean|Residu] [/FLUX Fmin Fmax] [/QUERY]

Perform a Steer-Dewdney-Ito CLEAN. This clean method selects an ensemble of clean components and remove them at once using FFTs. It works best for extended sources and UV coverages with short spacings. In such a case, it may avoid the "ringing" features which appear using the CLARK or HOGBOM techniques. In mosaic mode (see command MOSAIC), a mosaic clean is performed.

Clean the specified plane interval (default: planes between variables FIRST and LAST). If only FirstPlane is specified, Clean only that plane.

If option /PLOT is given, a display of the CLEAN or RESIDUAL map will be shown at each major cycle, depending on the argument (default: Residual). The user will be prompted for continuation when the /QUERY option is present. The cumulative, already cleaned flux is displayed in real time in an additional window while cleaning goes on when the /FLUX option is present. Parameters of the /FLUX option are then used to give the flux limits for this display.

The user can control the algorithm through SIC variables. New values can be given using "LET VARIABLE value". For ease of use, and whenever it is possible, a sensible value of each parameter will automatically be computed from the context if the value of the corresponding variable is set to its default value, i.e. zero value and empty string. A few variables are initialized to "reasonable" values.

```
    [CLEAN\]CLEAN ?
Will  list  all main CLEAN_* variables controlling the CLEAN parameters.
HELP CLEAN Variables will give a more complete list.
```

9.18.1 SDI Variables:

```
Basic parameters
CLEAN_GAIN       [       ] Loop gain
CLEAN_NITER      [       ] Maximum number of clean components
CLEAN_FRES       [      %] Maximum value of residual (Fraction of peak)
CLEAN_ARES       [Jy/Beam] Maximum value of residual (Absolute)
CLEAN_POSITIVE   [       ] Minimum number of positive components at start
CLEAN_NKEEP      [       ] Min number of components before convergence

Old names like in MAPPING
BLC              [  pixel] Bottom left corner of cleaning box
TRC              [  pixel] Top right corner of cleaning box
MAJOR            [ arcsec] Clean beam major axis
MINOR            [ arcsec] Clean beam minor axis
ANGLE            [ degree] Position angle of clean beam
BEAM_PATCH       [  pixel] Size of cleaning beam ** not clear **

Method dependent parameters
CLEAN_INFLATE    [      ] Maximum Inflation factor for UV_RESTORE (MuLTISCAL
CLEAN_NCYCLE     [      ] Max number of Major Cycles (SDI & CLARK methods)
CLEAN_NGOAL      [      ] Max number of comp. in Cycles (ALMA method)
CLEAN_RESTORE    [      ] Threshold for restoring a Mosaic (def 0.2)
CLEAN_SEARCH     [      ] Threshold to search Clean Comp. in a Mosaic (def 0
CLEAN_SIDELOBE   [      ] Min threshold to fit the synthesized beam
CLEAN_SMOOTH     [      ] Smoothing ratio (MRC and MULTISCALE)
CLEAN_SPEEDY     [      ] Speed-up factor (CLARK)
CLEAN_WORRY      [      ] Worry factor (MULTISCALE)
```

9.18.2 SDI CLEAN_ARES

```
This is the minimal flux in the dirty map that the program will consider
as  significant.  Alternatively,  the  threshold  can be specified as a
fraction of the peak flux using CLEAN_FRES.  Once this  level  has  been
reached the program stops subtracting, and starts the restoration phase.
The unit for this parameter is the map unit  (typically  Jy/Beam).  The
parameter  should  usually  be of the order of magnitude of the expected
noise in the clean map.

If 0, CLEAN_FRES will be used instead. If all of CLEAN_NITER, CLEAN_ARES
and CLEAN_FRES are 0, an absolute residual equal to the noise level will
be used for CLEAN_ARES.
```

Short form is ARES.

9.18.3 SDI CLEAN_FRES

This is the minimal fraction of the peak flux in the dirty map that the program will consider as significant. Alternatively, an absolute threshold can be specified using CLEAN_ARES. Once this level has been reached the program stops subtracting, and starts the restoration phase. This parameter is normalized to 1 (neither in % nor in db). It should usually be of the order of magnitude of the inverse of the expected dynamic range of the intensity.

If 0, CLEAN_ARES will be used instead. If all of CLEAN_NITER, CLEAN_ARES and CLEAN_FRES are 0, an absolute residual equal to the noise level will be used for CLEAN_ARES.

Short form is FRES.

9.18.4 SDI CLEAN_GAIN

This is the gain of the subtraction loop. It should typically be chosen in the range 0.05 and 0.3. Higher values give faster convergence, while lower values give a better restitution of the extended structure. A sensible default is 0.2.

Short form is GAIN.

9.18.5 SDI CLEAN_NITER

This is the maximum number of components the program will accept to subtract. Once it has been reached, the program starts the restoration phase.

If 0, the program will guess a number, based on the image size and maximum signal-to-noise ratio, and specified residual level CLEAN_ARES and/or CLEAN_FRES.

Short form is NITER.

9.18.6 SDI CLEAN_NKEEP

This is an integer specifying the minimum number of Clean components before testing if Cleaning has converged. The convergence is criterium is a comparison of the cumulative flux evolution separated by CLEAN_NKEEP components. If th

IF CLEAN_NKEEP is 0, CLEAN will ignore this convergence criterium, and continue clean until the CLEAN_NITER, CLEAN_ARES or CLEAN_FRES criteria indicate to stop.

With CLEAN_NKEEP > 0, CLEAN will explore the stability of the total clean flux over the last CLEAN_NKEEP iterations. For a positive (resp. negative) source, if the Clean flux becomes smaller (resp. larger) than the Clean flux CLEAN_NKEEP iterations earlier, CLEAN will stop.

Using CLEAN_NKEEP about 70 is a reasonable value. Some special cases (faint extended sources) may require larger values of CLEAN_NKEEP.

9.18.7 SDI CLEAN_POSITIVE

The minimum number of positive components before negative ones are selected.

9.18.8 SDI CLEAN_RESTORE

Fraction of peak response of the primary beams coverage under which the Sky brightness image is blanked in a Mosaic deconvolution.

The default is 0.2.

9.18.9 SDI CLEAN_SEARCH

Fraction of peak response of the primary beams coverage beyond which no Clean component is searched in a Mosaic deconvolution.

The default is 0.2.

9.18.10 SDI CLEAN_SIDELOBE

Minimal relative intensity to consider for fitting the syntheized beam to obtain the Clean beam parameters (MAJOR, MINOR and ANGLE) when 0. The default is 0.35.

In case of poor UV coverage, CLEAN_SIDELOBE should be higher than the maximum sidelobe level to perform a good Gaussian fit. Some particularly bad UV coverage may not allow any good fit at all, however.

9.18.11 SDI CLEAN_NGOAL

Number of clean components to be selected in a Cycle in the ALMA heterogeneous array cleaning method.

9.18.12 SDI CLEAN_NCYCLE

Maximum number of Major Cycles for the SDI and CLARK methods.

9.18.13 SDI CLEAN_SMOOTH

Smoothing factor between different scales in the MRC and MULTISCALE methods. The default is sqrt(3).

9.18.14 SDI CLEAN_SPEEDY

Speed-up factor for the CLARK major cycles. The default is 1.0. Larger values may be used, but at the expense of possible instabilities of the algorithm.

9.18.15 SDI CLEAN_WORRY

Worry factor in the MULTISCALE method for convergence. It propagates the S/N from one iteration to the other, so that if this S/N degrades, the method stops. Default is 0 (no propagation, and hence no test on S/N). The value should be < 1.0 in all cases.

9.18.16 SDI CLEAN_INFLATE

Maximum Inflation factor for UV_RESTORE (MULTISCALE method). If the

number of true (i.e. pixel based) Clean components found by MULTISCALE is larger than CLEAN_INFLATE times the number of compressed (i.e. those with the smoothing factor information) components, expansion of the compressed components will not be possible, and UV_RESTORE will not be useable.

A default of 50 is in general adequate. Better solutions might be found in the future, and this parameter suppressed. Apart from memory usage, this number has no consequence on the algorithm.

9.18.17 SDI METHOD

Method used for the deconvolution. Can be HOGBOM, MULTI, MRC, SDI or CLARK.

9.18.18 SDI Old_Names:

Some of the CLEAN parameters have kept their old names: MAJOR, MINOR, ANGLE (which are also used by command FIT) BLC, TRC and BEAM_PATCH (which are seldom used)

Others have equivalent short names: ARES, FRES, GAIN, NITER for which the CLEAN_ prefix may be omitted.

9.18.19 SDI BLC

These are the (pixel) coordinates of the Bottom Left Corner of the cleaning box. The actual cleaning support will be the intersection of the specified window with the inner quarter of the map and with any user defined polygon.

9.18.20 SDI TRC

These are the (pixel) coordinates of the Top Right Corner of the cleaning box. The actual cleaning window will be the intersection of the specified window with the inner quarter of the map and with any user defined polygon.

9.18.21 SDI MAJOR

This is the major axis (FWHP) in user coordinates of the Gaussian restoring beam. If 0, the program will fit a Gaussian to the dirty beam. We strongly discourage to change the default value of 0.

9.18.22 SDI MINOR

This is the minor axis (FWHP) in user coordinates of the Gaussian restoring beam. If 0, the program will fit a Gaussian to the dirty beam. We strongly discourage to change the default value of 0.

9.18.23 SDI ANGLE

This is the position angle (from North towards East, i.e. anticlockwise) of the major axis of the Gaussian restoring beam (in degrees). If 0, the program will fit a Gaussian to the dirty beam. We strongly discourage to change the default value of 0.

9.18.24 SDI BEAM_PATCH

The dirty beam patch to be used for the minor cycles in CLARK and MRC method. It should be large enough to avoid doing too many major cycles, but has practically no influence on the result. This size should be specified in pixel units. Reasonable values are between N/8 and N/4, where N is the number of map pixels in the same dimension. If set to N, the CLARK algorithm becomes identical to the HOGBOM algorithm.

9.19 SHOW

```
    [CLEAN\]SHOW Variable [Arg1 [Arg2]]
```

where
Variable
    is an internal buffer to be plotted (BEAM, CCT, CLEAN, DIRTY, etc..) or a SIC image variable.

```
    [CLEAN\]SHOW ?
```

will list the names of recognized keywords. If Variable is not one of the recognized keywords, but an existing Image variable, this image will be displayed.

Except for UV data where they have a different meaning,
Arg1 and Arg2
are optional arguments to restrict the range of channels to be plotted. They default to FIRST and LAST variable values respectively, and Arg2 defaults to Arg1 if only Arg1 is specified.

The buffer is displayed using the appropriate procedure
p_show_map for data cubes
p_show_cct for Clean Component Tables
p_uvshow_sub for UV data
Command VIEW offers a different style of display for cubes.

The UV data in the internal buffer is the one loaded by command READ UV File and optionally resampled by UV_RESAMPLE, UV_COMPRESS or transformed by UV_CONT. Flagged UV data will appear in a different color.

Note and Caution:
GO PLOT (or its variants GO BIT, GO NICE and GO MAP) and GO UVSHOW offer similar features, but take data from files or SIC image variables, depending on variables NAME and TYPE.

9.19.1 SHOW COVERAGE

SHOW COVERAGE [Ant [Date]]

Displays the UV coverage. Ant and Date are optional arguments indicating which Antenna is to be highlighted, and for which date. Date is a sequential number from 1 to the number of observing dates.

For spectral line UV data, SHOW COVERAGE will only show one UV coverage if FIRST and LAST are set to zero. It will show one per channel otherwise.

9.19.2 SHOW UV

SHOW UV [Ant [Date]]

Displays the UV data. Items to be displayed are controlled by XTYPE and YTYPE variables. Ant and Date are optional arguments indicating which Antenna is to be highlighted, and for which date. Date is a sequential number from 1 to the number of observing dates.

Structure uvshow% controls some additional options, such as coloring of various dates, of data flagging, etc...

9.19.3 SHOW GO_PLOT

GO PLOT provides a somewhat different plotting mechanism, using the disk files specified by NAME and TYPE (or the SIC variable specified by NAME if TYPE is empty)

GO PLOT Variable [First [Last]]

which will plot channels of the specified data set. First (default: 0) and Last (default: First) are optional arguments indicating the first and last channel to be displayed. If not specified the variables FIRST and LAST are used.

Most aspects of the display are controlled through the same SIC variables as the ones used by the "GO BIT|MAP|LMV" commands. Detailed information about those variables is found through the "INPUT LMV" command.

9.19.4 SHOW GO_UVSHOW

GO UVSHOW

Display UV data from the 'name'.uvt file. Control parameters are available as SIC variables. Use command INPUT UVSHOW to check them.

9.20 STATISTIC

[CLEAN\]STATISTIC [Clean|Dirty|Residu] [Plane] [/WHOLE]

Compute some basic statistics (min, max, mean, rms,...) on the specified buffer (default: Clean) and plane (default: variable FIRST). The current polygon (see command SUPPORT) is used to define the area on which the pixels are to be taken into account, except when option /WHOLE is present: The whole image is then considered.

9.21 STOKES

[CLEAN\]STOKES Key UVin [UVout]

Derive a single polarization UV table (UVout) with the polarization state specified by Key from a multi-polarization UV table (Uvin). If UVout is not specified, UVin is overwritten. Key is any of the following: NONE, I, Q, U, V, LL, RR, HH, VV.

A typical use is after command FITS on CASA data:

```
FITS Fits.uvfits TO UVin.uvt
STOKES NONE UVin
```

9.22 SUPPORT

```
  [CLEAN\]SUPPORT [Polygon] [/CURSOR] [/MASK] [/PLOT] [/RESET]
[/VARIABLE] [/THRESHOLD [Raw Smooth Length Guard]]
```

Define and/or plot the support inside which to search for CLEAN components. The support can be defined through a mask or a polygon, depending on the selected options. selected.

A polygon stored in a file, or in a Sic variable (/VARIABLE option), can be loaded as support. The /PLOT option can then be used to plot it.

9.22.1 SUPPORT /CURSOR

```
[CLEAN\]SUPPORT /CURSOR
```

With option /CURSOR, it calls the interactive cursor to define the polygon summits. Type any key to go to next summit, D to correct the last one and type E to end the polygon definition. The last polygon side will then appear. The polygon definition may be aborted by typing Q. For graphical displays, you may use the mouse buttons for the commands. The left mouse button draws a vertex, the middle mouse button deletes the last vertex, and the right mouse button ends the polygon definition.

The resulting support is available in the Sic structure SUPPORT:
- SUPPORT%NXY [Integer] Number of summits
- SUPPORT%X [Double] X coordinates
- SUPPORT%Y [Double] Y coordinates

9.22.2 SUPPORT /RESET

```
[CLEAN\]SUPPORT /RESET
```

Reset any support to default. This deletes the current polygon support. This does not unload any mask defined by READ MASK: it can be re-instated by SUPPORT /MASK.

9.22.3 SUPPORT /MASK

```
[CLEAN\]SUPPORT /MASK
```

Use the mask defined by READ MASK as the clean support. This does not suppress the current polygon: it can be re-instated by SUPPORT /PLOT. The Mask can be a 3-D array: CLEAN will find out match which mask plane must be used for each spectral channel.

Caution: READ MASK does not perform an implicit SUPPORT /MASK command.

9.22.4 SUPPORT /PLOT

[CLEAN\]SUPPORT [Name] /PLOT

Plot the current (or specified) polygon, or plot the current mask (if it is 2-D only).

9.22.5 SUPPORT /THRESHOLD

[CLEAN\]SUPPORT /THRESHOLD [Raw Smooth [Length [Guard]]

Define a Mask from thresholding the CLEAN image. If the CLEAN image is 3-D, the Mask will be 3-D. ‘

The method involves a first thresholding, followed by a smoothing to extend the support and a second thresholding.

The algorithm to define the mask is controlled by 4 parameters.
Raw
Initial threshold (in units of CLEAN image noise) under which the mask is set to 0. Default is 6 sigma. The noise is taken from the computed Clean noise (clean%gil%rms) if defined, or from the theoretical noise (dirty%gil%noise) if not.
Smooth
Threshold under which the Mask is set to 0 after smoothing. The default is 2 sigma.
Length
FWHM of the smoothing gaussian to derive the smooth mask from the initial mask. The default is the Clean beam major axis.
Guard
Size of the guard band at edges where the mask is set to zero, in units of image size. The default is 0.18, i.e. the mask can extends a little more than the inner quarter. This is to avoid the edges where sidelobes aliasing occurs.

The computed mask can be displayed by SHOW MASK and saved by WRITE MASK for further use.

9.22.6 SUPPORT /VARIABLE

```
[CLEAN\]SUPPORT VarName /VARIABLE
```

Load the support from the Sic variable VarName. VarName can be an array of the form:

```
VarName[NXY,2]  Real or Double
```

or a structure of the form (i.e. same as output):

```
VarName%NXY              Integer or Long
VarName%X[VarName%NXY]   Real or Double
VarName%Y[VarName%NXY]   Real or Double
```

9.23 UV_BASELINE

```
[CLEAN\]UV_BASELINE [Degree] [/CHANNELS Channel_List]
[/FREQUENCY List Of Frequencies] [/RANGE Min Max [TYPE]]
[/VELOCITY List of Velocities] [/WIDTH Width [TYPE]]
```

Subtract a continuum from a line UV data set, by fitting a baseline for each visibility. The channels to be ignored in this process (i.e. the ones including the line emission) can be specified either by the /FREQUENCY or /VELOCITY options in combination with the /WIDTH option, or by a channel list with /CHANNELS or a range (of channels, frequencies or velocities) with /RANGE.

9.23.1 UV_BASELINE /CHANNELS

```
[CLEAN\]UV_BASELINE /CHANNELS Channel_List
```

Channel_List must be a 1-D SIC variable containing the list of channels to filter out.

9.23.2 UV_BASELINE /FREQUENCY

```
[CLEAN\]UV_BASELINE /FREQUENCY F1 [... [Fn]] [/WIDTH Width [TYPE]]
```

Specify around which frequencies the line emission should be filtered. Frequencies F1 to Fn must be in MHz. The full width of the filtering window around every frequency can be set by option /WIDTH. The optional argument TYPE indicates the type of width: FREQUENCY (in MHz), VELOCITY (in km/s) or CHANNEL (no unit), the default being FREQUENCY. The default width is the current channel width.

Tip: it can be convenient to have a list of SIC variables containing the

frequencies of the most intense spectral lines, e.g.
HCO10 = 89188.52

9.23.3 UV_BASELINE /RANGE

[CLEAN\]UV_BASELINE /FREQUENCY F1 [... [Fn]] /RANGE Min Max [TYPE]

Indicate that channels between the First and Last defined by Min Max and Type contain line emission and should be ignored in the baseline fitting. Type can be CHANNEL, VELOCITY or FREQUENCY.

For type CHANNEL, Min and Max indicate offsets from Channel 1 and Channel Nchan (the number of channels in the data set). Thus Max can be negative: it then indicates Last = Nchan-Max. Also Min=0 and Max=0 implies loading all the channels.

9.23.4 UV_BASELINE /VELOCITY

[CLEAN\]UV_BASELINE /VELOCITY V1 [... [Vn]] [/WIDTH Width [TYPE]]

Specify around which velocities the line emission should be filtered. Velocities V1 to Vn must be in km/s. The full width of the filtering window around every frequency can be set by option /WIDTH. The optional argument TYPE indicates the type of width: FREQUENCY (in MHz), VELOCITY (in km/s) or CHANNEL (no unit), the default being FREQUENCY. The default width is the current channel width.

9.23.5 UV_BASELINE /WIDTH

[CLEAN\]UV_BASELINE /FREQUENCY F1 [... [Fn]] /WIDTH Width [TYPE]

Specify the full width of the window around every frequency given in the /FREQUENCY option. The optional argument TYPE indicates the type of width: FREQUENCY (in MHz), VELOCITY (in km/s) or CHANNEL (no unit), the default being FREQUENCY. The default width is the current channel width. The default width is the current channel width.

9.24 UV_CHECK

[CLEAN\]UV_CHECK Beams|Nulls

Check UV data for weight consistency.

BEAMS
List which channel range can be processed with the same synthesized

beam.

NULLS
Check if there are null visibilities with non-zero weights, and flag them if found.

9.25 UV_COMPRESS

[CLEAN\]UV_COMPRESS Nc

Resample the UV data loaded by READ UV by averaging NC adjacent channels. All other UV commands except UV_RESAMPLE work on the "Resampled" UV table.

The "Resampled" UV table is a simple copy of the original one after a READ UV command, or after a UV_RESAMPLE or UV_COMPRESS commands without arguments.

9.26 UV_CONTINUUM

[CLEAN\]UV_CONTINUUM Naver [First Last]

Transform the (presumably spectral line) UV data set loaded by READ UV into a "continuum" data set.

The transformation selects line channels from First to Last, average them by groups of Naver contiguous channels, and concatenate the resulting visibilities into a "continuum" UV table.

For each of the (First-Last+1)/Naver channels, and for all visibilities, the U and V coordinates are rescaled to the mean observing frequency, and the resulting (single-channel) visibilities are concatenated into the "continuum" UV data set. The continuum dataset becomes the current UV data. Flagged channels are ignored: this allows to mask channels containing spectral lines (see UV_FILTER).

Default for First and Last is 0 0, meaning all channels are selected. Negative value for Last indicate an offset from end channel (i.e. -20 means ignore the last 20 channels).

9.27 UV_FILTER

[CLEAN\]UV_FILTER [/ZERO] [/CHANNELS Channel_List]

```
[/FREQUENCY List Of Frequencies] [/RANGE Min Max [TYPE]]
[/VELOCITY List of Velocities] [/WIDTH Width [TYPE]]
```

"Filter" line emission, by flagging the corresponding channels. The channels can be specified either by the /FREQUENCY or /VELOCITY options in combination with the /WIDTH option, or by a channel list with /CHANNELS or a range (of channels, frequencies or velocities) with /RANGE.

By default, channels to be filtered are flagged (weights becoming negative). Option /ZERO can be used to erase them: weights and visibilities are set to zero, see HELP UV_FILTER /ZERO.

9.27.1 UV_FILTER /CHANNELS

```
[CLEAN\]UV_FILTER [/ZERO] /CHANNELS Channel_List
```

Channel_List must be a 1-D SIC variable containing the list of channels to filter out.

9.27.2 UV_FILTER /FREQUENCY

```
[CLEAN\]UV_FILTER [/ZERO] /FREQUENCY F1 [... [Fn]] [/WIDTH Width]
```

Specify around which frequencies the line emission should be filtered. Frequencies F1 to Fn must be in MHz. The full width of the filtering window around every frequency can be set by option /WIDTH. The optional argument TYPE indicates the type of width: FREQUENCY (in MHz), VELOCITY (in km/s) or CHANNEL (no unit), the default being FREQUENCY. The default width is the current channel width.

Tip: it can be convenient to have a list of SIC variables containing the frequencies of the most intense spectral lines, e.g.

```
HCO10 = 89188.52
```

9.27.3 UV_FILTER /RANGE

```
[CLEAN\]UV_FILTER [/ZERO] /RANGE Min Max [TYPE]
```

Indicate that channels between the First and Last defined by Min Max and Type contain line emission and should be filtered out. Type can be CHANNEL, VELOCITY or FREQUENCY.

For type CHANNEL, Min and Max indicate offsets from Channel 1 and Channel Nchan (the number of channels in the data set). Thus Max can be negative: it then indicates Last = Nchan-Max. Also Min=0 and Max=0 implies loading all the channels.

9.27.4 UV_FILTER /VELOCITY

```
[CLEAN\]UV_FILTER /VELOCITY V1 [... [Vn]] [/WIDTH Width [TYPE]]
```

Specify around which velocities the line emission should be filtered. Velocities V1 to Vn must be in km/s. The full width of the filtering window around every frequency can be set by option /WIDTH. The optional argument TYPE indicates the type of width: FREQUENCY (in MHz), VELOCITY (in km/s) or CHANNEL (no unit), the default being FREQUENCY. The default width is the current channel width.

9.27.5 UV_FILTER /WIDTH

```
[CLEAN\]UV_FILTER [/ZERO] /FREQUENCY F1 [... [Fn]] /WIDTH Width
[TYPE]

[CLEAN\]UV_FILTER [/ZERO] /VELOCITY V1 [... [Vn]] /WIDTH Width
[TYPE]
```

Specify the full width of the filtering window around every frequency given in the /FREQUENCY option or any velocity given in the /VELOCITY option. The optional argument TYPE indicates the type of width: FREQUENCY (in MHz), VELOCITY (in km/s) or CHANNEL (no unit), the default being FREQUENCY. The default width is the current channel width.

9.27.6 UV_FILTER /ZERO

```
[CLEAN\]UV_FILTER /ZERO [/CHANNELS Channel_List]
[/FREQUENCY F1 [... [Fn]] [/VELOCITY V1 [... [Vn]]
[/RANGE Min Max [TYPE]] [/WIDTH Width [TYPE]]
```

Erase filtered channels (set weight and visibilities to zero) rather than simply flagging them (weight set to negative value). This can be more convenient for further display, but is not reversible.

By default, channels to be filtered are flagged (weights becoming negative, and data can be unflagged by UV_FLAG). Flagged channels are ignored in the averaging process, such as UV_RESAMPLE or UV_CONT. However, as UV_MAP (in general) uses only one channel to define the weights for all others, these flagged channels will nevertheless appear in the imaged data cube.

See UV_CHECK for information about handling flagged channels and different beams in a single UV table.

9.28 UV_FLAG

```
[CLEAN\]UV_FLAG [/RESET]
```

Display UV data and calls the cursor to interactively select a region where UV data will be flagged (sign of weight is reversed). The /RESET option is used to unflag the data. UV data flagged using command UV_FLAG can be saved on file with command WRITE UV File. Detailed information about the display control of the UV data may be found in the UV_SHOW help.

The user can control the algorithm through variables. Values of those variables can be checked using "EXAMINE VARIABLE". New values can be given using "LET VARIABLE value".

```
DATE_START  [    ] The date of the first data to be flagged
UT_START    [    ] The UT time of the first data to be flagged
DATE_END    [    ] The date of the last data to be flagged
UT_END      [    ] The time of the last data to be flagged
BASELINE    [    ] Baseline to flagged
CHANNEL     [    ] Channel range to be flagged/unflagged
FLAG        [    ] Flag or unflag the specified time range
```

Subsequent mapping with UV_MAP will ignore flagged data. However, for multi-channel imaging, the weight column is (by default) taken from one channel, so that channel-based flagging will not be recognized in the default mode, but only in the "One Beam per Channel" mode.

9.28.1 UV_FLAG DATE_START

The date (DD-MMM-YYYY) of the first data to be flagged.

9.28.2 UV_FLAG UT_START

The UT time (hh:mm:ss.ss) of the first data to be flagged.

9.28.3 UV_FLAG DATE_END

The date (DD-MMM-YYYY) of the last data to be flagged.

9.28.4 UV_FLAG UT_END

The time (hh:mm:ss.ss) of the last data to be flagged.

9.28.5 UV_FLAG BASELINE

Baseline (ALL, 12, 13, 23, ...) to flagged.

9.28.6 UV_FLAG FLAG

Flag (T) or unflag (F) the specified time range.

9.28.7 UV_FLAG CHANNEL

Channel range to be flagged/unflagged.

9.29 UV_MAP

[CLEAN\]UV_MAP [CenterX CenterY UNIT [Angle]] [/FIELDS FieldList] [/TRUNCATE Percent]

Compute a dirty map and beam from a UV data. UV data must have been loaded from a UV table by command "READ UV File". UV_MAP processes single fields as well as Mosaics.

The user can control the algorithm through SIC variables. New values can be given using "LET VARIABLE value". For ease of use, and whenever it is possible, a sensible value of each parameter will automatically be computed from the context if the value of the corresponding variable is set to its default value, i.e. zero value and empty string. A few variables are initialized to "reasonable" values.

[CLEAN\]UV_MAP ?
Will list all MAP_* variables controlling the UV_MAP parameters

9.29.1 UV_MAP Mosaics

[CLEAN\]UV_MAP [CenterX CenterY UNIT [Angle]] /FIELDS FieldList [/TRUNCATE Percent]

The UV data can be a Mosaic UV table. In this case, UV_MAP will image

the Mosaic, using appropriate primary beam size and truncation level.

By default, the primary beam size is taken from the telescope parameters, either as found in the telescope section of the UV table and the observing frequency (e.g. for ALMA it uses 1.13 Lambda/D). If absent, the telescope section information can be added by command SPECIFY TELESCOPE.

The truncation level is taken from variable MAP_TRUNCATE or from the /TRUNCATE option argument.

9.29.2 UV_MAP /FIELDS

[CLEAN\]UV_MAP [CenterX CenterY UNIT [Angle]] /FIELDS FieldList [/TRUNCATE Percent]

For a Mosaic, only image the fields specified in a 1-D Integer variable FieldList. SHOW FIELDS will highlight the list of fields selected by this command.

9.29.3 UV_MAP /TRUNCATE

[CLEAN\]UV_MAP [CenterX CenterY UNIT [Angle]] /FIELDS FieldList /TRUNCATE Percent

For a Mosaic, truncate the primary beam to the specified level (in percent). SHOW FIELDS will use this level to show the beams. The default is to use MAP_TRUNCATE.

9.29.4 UV_MAP Variables:

[CLEAN\]UV_MAP ?
Will list all MAP_* variables controlling the UV_MAP parameters.

The list of control variables is (by alphabetic order, with the old names used by Mapping on the right)

New names	[unit]	-- Description -- % Old Name
MAP_BEAM_STEP	[ ]	Number of channels per single dirty beam
MAP_CELL	[arcsec]	Image pixel size
MAP_CENTER	[string]	RA, Dec of map center, and Position Angle
MAP_CONVOLUTION	[ ]	Convolution function % CONVOLUTION
MAP_FIELD	[arcsec]	Map field of view
MAP_POWER	[ ]	Maximum exponent of 3 and 5 allowed in MAP_SIZE
MAP_PRECIS	[ ]	Fraction of pixel tolerance on beam matching
MAP_ROBUST	[ ]	Robustness factor % UV_CELL[2]

```
MAP_ROUNDING    [       ]  Precision of MAP_SIZE
MAP_SIZE        [       ]  Number of pixels
MAP_TAPEREXPO   [       ]  Taper exponent           % TAPER_EXPO
MAP_TRUNCATE    [      %]  Mosaic truncation level
MAP_UVCELL      [      m]  UV cell size             % UV_CELL[1]
MAP_UVTAPER     [m,m,deg]  Gaussian taper           % UV_TAPER
MAP_VERSION     [       ]  Code version (0 new, -1 old)

NAME is no longer used, and WEIGHT_MODE is obsolete.
MAP_RA          [  hours]  RA of map center
MAP_DEC         [    deg]  Dec of map center
MAP_ANGLE       [    deg]  Map position angle
MAP_SHIFT       [Yes/No ]  Shift phase center
```

are obsolescent, superseded by MAP_CENTER. They are provided only for compatibility with older scripts.

9.29.5 UV_MAP MAP_BEAM_STEP

```
MAP_BEAM_STEP   Integer
```

Number of channels per synthesized beam plane.

Default is 0, meaning only 1 beam plane for all channels. N (>0) indicates N consecutive channels will share the same dirty beam.

A value of -1 can be used to compute the number of channels per beam plane to ensure the angular scale does not deviate more than a fraction of the map cell at the map edge. This fraction is controlled by variable MAP_PRECIS (default 0.1)

9.29.6 UV_MAP MAP_CELL

```
MAP_CELL[2]    Real
```

The map pixel size [arcsec]. It is recommended to use identical values in X and Y. A sampling of at least 3 pixel per beam is recommended to ease the deconvolution. Enter 0,0 to let the task find the best values.

9.29.7 UV_MAP MAP_CENTER

```
MAP_CENTER     Character String
```

Specify the Map center and orientation in the same way as the arguments

of UV_MAP.

9.29.8 UV_MAP MAP_CONVOLUTION

MAP_CONVOLUTION Integer

Select the desired convolution function for gridding in the UV plane Choices are

0 Default (currently 5)
1 Boxcar
2 Gaussian
3 Sin(x)/x
4 Gaussian * Sin(x)/x
5 Spheroidal

Spheroidal functions is the optimal choice. So we strongly discourage use of any other convolution function, which are here for tests only.

9.29.9 UV_MAP MAP_FIELD

MAP_FIELD[2] Real

Field of view in X and Y in arcsec. The field of view MAP_FIELD has precedence over the number of pixels MAP_SIZE to define the actual map size when both are non-zero.

9.29.10 UV_MAP MAP_POWER

MAP_POWER[2] Integer

Maximum exponent of 3 and 5 allowed in automatic guess of MAP_SIZE. MAP_SIZE is decomposed in 2^k 3^p 5^q, and p and q must be less or equal to MAP_POWER.

Default is 0: MAP_SIZE is just a power of 2. A value of 1 allows approximation of any map size to 20 %, while a value of 2 allows 10 % approximation. Fast Fourier Transform are slightly slower with powers of 3 and 5, but limiting the map size can gain a lot in the Cleaning process (which can scale as MAP_SIZE^4).

9.29.11 UV_MAP MAP_PRECIS

MAP_PRECIS Real

Maximum mismatch in pixel at map edge between the true synthesized beam (which would have been computed using the exact channel frequency) and the computed synthesized beam with the mean frequency of the channels sharing the same beam. This is used (with the actual image size) to derive the actual number of channels which can share the same beam, i.e. the effective value of MAP_BEAM_STEP when MAP_BEAM_STEP is -1.

Default is 0.1

9.29.12 UV_MAP MAP_ROBUST

MAP_ROBUST Real

Robust weighting factor. A number between 0 and +infty.

Robust weighting gives the natural weight to UV cells whose natural weight is lower than a given threshold. In contrast, if the natural weight of the UV cell is larger than this threshold, the weight is set to this (uniform) threshold. The UV cell size is defined by MAP_UVCELL and the threshold value is in MAP_ROBUST.

0 means natural weighting, which is optimal for point sources. The Robust weighting factor controls the resolution: better resolution is obtained for small values (at the expense of noise), resolution approaching the natural weighting scheme for large values. Larger UV cell size give higher angular resolution (but again more noise).

MAP_ROBUST around .5 to 1 is a good compromise between noise increase and angular resolution.

9.29.13 UV_MAP MAP_ROUNDING

MAP_ROUNDING Real

Maximum error between optimal size (MAP_FIELD / MAP_CELL) and rounded (as a power of 2^k 3^p 5^q) MAP_SIZE to round by floor (thus limiting the field of view), instead of ceiling (which guarantees a larger field of view, but leads to bigger images).

Default is 0.05.

9.29.14 UV_MAP MAP_SHIFT

```
MAP_SHIFT        Logical
```

Obsolescent, superseded by MAP_CENTER, or the UV_MAP arguments.

Logical variable indicating whether map center (i.e. phase tracking center) or orientation should be changed.

9.29.15 UV_MAP MAP_SIZE

```
MAP_SIZE[2]      Integer
```

Number of pixels in X and Y. It should preferentially be a power of two, (although this is not strictly required) to speed-up the FFT. MAP_SIZE*MAP_CELL should be at least twice the size of the field-of-view (primary beam size for a single field). Enter 0,0 to let the command find a sensible map size.

MAP_SIZE is not used if MAP_FIELD is non zero.

Odd values are forbidden.

Default is 0,0, i.e. UV_MAP will guess the most appropriate values which depend on MAP_ROUNDING and MAP_POWER.

9.29.16 UV_MAP MAP_TAPEREXPO

```
MAP_TAPEREXPO    Real
```

Taper exponent. The default is 2 (indicating a Gaussian) but smoother or sharper functions can be used. 1 would give an Exponential, 4 would be getting close to square profile...

9.29.17 UV_MAP MAP_TRUNCATE

```
MAP_TRUNCATE    Real
```

Mosaic truncation level in PerCent. Default value is 0.2. Current value can be overriden by option /TRUNCATE in commands UV_MAP or PRIMARY.

9.29.18 UV_MAP MAP_UVTAPER

MAP_UVTAPER[3] Real

Parameters of the tapering function (Gaussian if MAP_TAPEREXPO = 2): major axis at 1/e level [m], minor axis at 1/e level [m], and position angle [deg].

9.29.19 UV_MAP MAP_UVCELL

MAP_UVCELL Real

UV cell size for robust weighting [m]. Should be of the order of half the dish diameter (7.5 m for PdBI), or smaller or even larger. It controls the beam shape in Robust weighting.

9.29.20 UV_MAP MAP_VERSION

MAP_VERSION Integer

[EXPERT Only] Code indicating which version of the UV_MAP and UV_RESTORE algorithm should be used. 0 is optimal. -1 is the "historical" (pre-2016) version. 1 is an intermediate version used during multi-frequency beams development.

9.29.21 UV_MAP MCOL

MCOL[2] Integer

First and Last channel to image. Values of 0 mean imaging all the planes.

9.29.22 UV_MAP WCOL

WCOL Integer

[Obsolescent] The channel from which the weight should be taken. WCOL set to 0 means using a default channel. WCOL has no real meaning in all cases where more than one beam is computed for all channels.

9.29.23 UV_MAP Old_Names:

```
NAME         [       ]  Label of the dirty image and beam plots
UV_TAPER     [m,m,deg]  UV-apodization by convolution with a Gaussian
WEIGHT_MODE  [       ]  Weighting mode (NA|UN)
UV_CELL      [m, ??  ]  UV cell size and threshold for Robust weighting
MAP_FIELD    [ arcsec]  Map field of view
MAP_CELL     [ arcsec]  Map cell size
MAP_SIZE     [ pixels]  Map size in pixels (if MAP_FIELD is zero)
MCOL         [       ]  First and Last channel to map
WCOL         [       ]  Channel from which the weights are taken
CONVOLUTION  [       ]  Convolution function (5)
UV_SHIFT     [       ]  Change the map phase center or map orientation?
MAP_RA       [       ]  RA of map phase center
MAP_DEC      [       ]  Dec of map phase center
MAP_ANGLE    [    deg]  Map position angle
MAP_BEAM_STEP [     ]  Number of channels per synthesized beam plane
```

9.29.24 UV_MAP convolution

```
Older variable name for MAP_CONVOLUTION
```

9.29.25 UV_MAP map_angle

```
MAP_ANGLE      Real
```

Position Angle of the direction which will become the apparent North in the map. Used only if UV_SHIFT is YES.

Superseded by MAP_CENTER.

9.29.26 UV_MAP map_dec

```
MAP_DEC     Real
```

Dec of map center. Used only if UV_SHIFT is YES.

Superseded by MAP_CENTER.

9.29.27 UV_MAP map_ra

MAP_RA Real

RA of map center. Used only if UV_SHIFT is YES.

Superseded by MAP_CENTER.

9.29.28 UV_MAP uv_cell

Older variables for MAP_UVCELL (uv

9.29.29 UV_MAP uv_shift

Older variable name of MAP_SHIFT (this one is also obsolescent)

9.29.30 UV_MAP uv_taper

Older variable name of MAP_UVTAPER

9.29.31 UV_MAP taper_expo

Older variable name for MAP_TAPEREXPO

9.29.32 UV_MAP weight_mode

weightde Character

Weighting mode: Natural (optimum in terms of sensitivity) or robust (usually lower sidelobes and higher spatial resolution) weighting. This was needed in Mapping to toggle between Natural and Robust weighting, while IMAGER does that based on MAP_ROBUST value.

9.30 UV_RESAMPLE

```
[CLEAN\]UV_RESAMPLE [Nc Ref Val Inc]
```

Resample the UV data loaded by READ UV on a different velocity scale. All other UV commands except UV_COMPRESS work on the "Resampled" UV table.

```
Nc   new number of channels
Ref  New reference pixel
Val  New velocity at reference pixel
Inc  Velocity increment
```

Any argument can be set to * for an automatic determination based on the values of the other arguments. The automatic determination of NC preserves the velocity coverage.

The "Resampled" UV table is a simple copy of the original one after a READ UV command, or after a UV_RESAMPLE or UV_COMPRESS command without arguments.

9.31 UV_RESIDUAL

```
[CLEAN\]UV_RESIDUAL [Niter]
```

Subtract the Niter first (default all) Clean Components from the UV data. The residual UV data can be written by WRITE UV, and imaged by UV_MAP.

9.32 UV_RESTORE

```
[CLEAN\]UV_RESTORE
```

Create a Clean image from the current UV data set and the Clean Component list. The Clean Components are subtracted from the UV data set, and these residuals are gridded and Fourier transformed to compute the Residual image. This Residual image is added to the Gaussian beam convolved image of the sum of Clean components. The results are similar to those of MX, since only the residual are aliased.

This command can be used after HOGBOM, CLARK, MULTI, SDI or MX (although it is pointless after MX), but not MRC which has no notion of Clean Components.

9.33 UV_REWEIGHT

```
[CLEAN\]UV_REWEIGHT Scale
```

Scale the weights of the current UV data with the defined Scale factor. Can be used to patch e.g. JVLA data files which may have only relative weights, not absolute values indicating the noise.

9.34 UV_SHIFT

```
[CLEAN\]UV_SHIFT [CenterX CenterY UNIT [Angle]]
```

Shift the current UV table (single field or mosaic) to a common phase center. If no argument is specified, the string contained in MAP_CENTER is used instead.

Although UV_MAP also provides a direct possibility to re-center the image on a specified projection (phase) center, the modified visibilities cannot be saved on file. It is sometimes required to do this for specific use. UV_SHIFT provides this possibility, and the shifted UV table can be written using command WRITE UV.

UV_DEPROJECT includes the UV_SHIFT capabilities, but also has additional functions.

9.35 UV_SORT

```
[CLEAN\]UV_SORT TIME|BASE
```

Sort and transpose the UV data set, loaded by command READ UV File. Order is either TIME for Time-Baseline ordering, BASE for Baseline-Time ordering. The sorted UV data is then available in variable UVS for further plotting.

NOTE: This is only done in an internal buffer. WRITE UV will **NOT** write this sorted, transposed, buffer.

9.36 UV_STAT

```
[CLEAN\]UV_STAT CELL|HEADER|SETUP|TAPER|WEIGHT [Step Start]
```

UV_STAT allows the astronomer to select the best weighting and imaging parameters according to its personal trade off between angular resolution, sensitivity and field of view.

Default is HEADER+SETUP

9.36.1 UV_STAT CELL

[CLEAN\]UV_STAT CELL [Step Start]

Predict the synthesized beam, expected noise level, and recommended pixel size for different values of the uv cell size for current robust weighting parameters.

9.36.2 UV_STAT HEADER

[CLEAN\]UV_STAT HEADER

Display a brief summary of the UV data: number of antennas, observing dates, baseline ranges, spectroscopic information.

9.36.3 UV_STAT SETUP

[CLEAN\]UV_STAT SETUP

Display recommended values for the imaging: image size, pixel size, field of view, largest angular scale, etc...

9.36.4 UV_STAT TAPER

[CLEAN\]UV_STAT TAPER [Step Start]

For TAPER, beam sizes and noise level (in flux and brightness) will be computed for 9 different tapers (from Start to Start*Step^9). Default value for Step is sqrt(2), Default value for Start is 50 m. Weighting mode, UV cell size and "robust" parameter are taken from variable UV_CELL (i.e. one can combine Robust weighting and Tapering).

9.36.5 UV_STAT WEIGHT

[CLEAN\]UV_STAT WEIGHT [Step Start]

For WEIGHT, beam sizes and noise level (in flux and brightness) will be computed for 9 different "robust" weighting parameters (from Start to Start*Step^9). Default value for Step is sqrt(10), and default value for Start is derived to center the "robust" parameter values around 1. UV cell size is taken from variable UV_CELL[1], and Taper is taken from variable UV_TAPER.

9.37 UV_TIME

```
[CLEAN]UV_TIME [Time] [/Weight Wcol]
```

Average in time the current UV data set to reduce the number of visibilities. Time must be in seconds. If not specified, an automatic guess is performed based on antenna size and baseline lengths.

9.37.1 UV_TIME /WEIGHT

```
[CLEAN]UV_TIME Time /WEIGHT Wcol
```

Select the weight column. Default is 0.

9.38 UV_TRUNCATE

```
[CLEAN]UV_TRUNCATE Max [Min]
```

Truncate the UV data, by removing baselines out of the specified range Min (default 0) and Max (in meter).

9.39 VIEW

```
[CLEAN\]VIEW Variable [FirstPlane [LastPlane]]
```

where
is the internal buffer to be plotted (BEAM, CCT, CLEAN, DIRTY, FIELDS, MASK, etc..), or any Sic Image variable.

```
[CLEAN\]VIEW ?
```

will list the names of recognized keywords. If Variable is not one of the recognized keywords, but an existing Image variable, this image will be displayed.

FirstPlane and LastPlane
are optional arguments to restrict the range of channels to be plotted (default: planes between variables FIRST and LAST). If only FirstPlane is specified, SHOW only that plane.

The plot is done by procedure p_view_cct for Clean Components CCT and p_view_map for data cubes. For data cubes, the behaviour is the same as GO VIEW with NAME set to the appropriate buffer name.

VIEW is also controlled by a set of variables, available in the View%

```
global structure
```

9.39.1 VIEW Variables

```
User control variables
view%expand       0.8 Character and Tick expansion factor
view%contour      NO  Contour the current channel map
view%movie        0   Elapsed time in seconds for a movie (0 means guess)
view%side         YES Display values in side Window
view%status%rima  NO  Relative coordinates in Images
view%status%rspe  NO  Relative coordinates in Spectra

Returned values are found in view%current structure. The ZOOM action
produces a sub-cube named EXTRACTED. The SLICE action produces a 2-D
data set named SLICE. As any other SIC Image variable, EXTRACTED and
SLICE can be written on disk using command WRITE.
```

9.39.2 VIEW Keys

```
Position-independent actions:
  Press E key: EXIT loop
  Press H key: HELP display
  Press M key: MOVIE
  Press P key: PRINT plot in "ha" subdirectory
  Press Q key: QUIT loop
  Press X key: EXTRACT on disk current zoomed region
  Press N key: Toggle  Narrow - Wide mode

Cursor on images:
  Left  clic: Display spectrum at pointed position
  Right clic: Define a polygon
  Press C key: COORDINATES toggled from absolute to relative and back
  Press S key: SLICE definition (velocity-position)
  Press K key: KILL   pointed pixel
  Press U key: UNKILL pointed pixel
  Press Z key: ZOOM defined spatial region
  Press B key: BACK to full field of view
  Press V key: Display map coordinates at current position (the associated b
  Press W key: WRITE Integrated Area image
Cursor on spectra:
  Left clic in Selected   Spectrum: Display selected velocity channel
  Left clic in Integrated Spectrum: Define velocity range
  Press C key: COORDINATES toggled from freq/velo to channels and back
  Press Z key: ZOOM defined velocity region
```

```
  Press B key: BACK to full velocity range
  Press W key: WRITE Integrated (and current) Spectrum
Cursor outside plots:
  Press B key: BACK to full velocity range AND full field of view
```

ZOOM action produces a sub-cube named EXTRACTED. SLICE action produces a 2-D data set named SLICE. As any other SIC Image variable, EXTRACTED and SLICE can be written on disk using command WRITE.

9.40 WRITE

```
    [CLEAN\]WRITE Name File [/APPEND] [/RANGE Start End Kind] [/REPLACE]
[/TRIM]
```

WRITE the buffer specified by Name (UV, CGAINS, and BEAM, PRIMARY, DIRTY, CLEAN, RESIDUAL, SUPPORT, CCT, SKY) onto a File. Default extensions are .uvt for UV and CGAINS and .beam, .lmv, .lobe, .lmv-clean, .lmv-res, .pol, .cct, and lmv-sky respectively for the next ones. If Name does not refer to a known buffer, but to a SIC Image variable, this variable is written. The default extension is then .gdf.

For UV data, the flagged data are written by default. They may be omitted using the /TRIM option.

WRITE * File can be used to write all modified image-like buffers (not the UV tables) under a common File name. This typically include .beam, .lmv and .lmv-clean, but also the .lmv-sky file if the PRIMARY command has been used after deconvolution.

9.40.1 WRITE /RANGE

```
    [CLEAN\]WRITE Buffer File /RANGE Start End Kind [/REPLACE]
```

A range of image planes can be specified through /RANGE option. Kind is the unit of the Start-End values: CHANNEL, VELOCITY, or FREQUENCY.

Restrictions:
- The option is still experimental
- The Buffer name must be specified (* is not valid here)
- This does not apply to UV data.

9.40.2 WRITE /APPEND

```
    [CLEAN\]WRITE Buffer File /APPEND [/RANGE Start End Kind]
```

EXPERIMENTAL: The selected channels are appended to an existing file.

9.40.3 WRITE /REPLACE

[CLEAN\]WRITE Buffer File /REPLACE [/RANGE Start End Kind]

EXPERIMENTAL: The selected channels are replaced in an existing file.

9.40.4 WRITE /TRIM

[CLEAN\]WRITE UV File /TRIM

Remove the flagged visibilities while writing.

10 CALIBRATE Language Internal Help

10.1 Language

```
APPLY         : Apply gain solution to current UV Data
FLUX_SCALE    : Adjust flux scale on a daily basis
MODEL         : Compute a UV model from the Clean Components Table
SOLVE         : Solve for complex gains using the UV model
UV_SELF       : Build the Self Calibration UV Table and dirty image
```

10.2 APPLY

```
[CALIBRATE\]APPLY [AMPLI|PHASE [gain]] [/FLAG]
```

Apply gain solution computed by MODEL and SOLVE (which are called implicitely by SELFCAL) or obtained by READ CGAINS to the current UV data. The optional arguments indicate whether this should be an AMPLITUDE or PHASE gain, and what gain value is used (in range 0 to 1).

If no argument is given, the current SELF_MODE (see HELP SELFCAL) is used, and the gain is 1.0.

The /FLAG option controls whether data without a valid gain solution are kept unchanged or flagged.

10.2.1 APPLY /FLAG

```
[CALIBRATE\]APPLY [AMPLI|PHASE [gain]] /FLAG
```

Apply gain solution (in AMPLITUDE or PHASE) and flag data without a corresponding valid gain solution.

10.3 FLUX_SCALE

```
[CALIBRATE\]FLUX Find|Apply|List|Calibrate [VarName]
```

A set of commands to check flux calibration on a day to day basis. A supervising procedure, check_flux.map, allows self-calibration of the flux by using as intermediate model a table generated by the Clean Component list.

FLUX_SCALE FIND DateTolerance
determines, by linear regression, the best scaling factor to match date by date the UV data set with the MODEL data set. Dates are assumed iden-

tical if separated by less than DateTolerance

FLUX_SCALE APPLY VarName
apply previously determined flux scale factors to the MODEL data set, previously read by READ MODEL. This is in general used only in an iterative search way, e.g. by procedure check_flux. The resulting UV data set is loaded into the specified VarName SIC variable.

FLUX_SCALE LIST
print out dates, baselines and determined flux factors

FLUX_SCALE CALIBRATE
apply previously determined flux scale factors to the UV data set. This may then be written using command WRITE UV .

10.4 MODEL

[CALIBRATE\]MODEL [MaxIter] [/MINVAL Value [Unit]]

Compute visibilities on the current UV sampling using a source model made of the MaxIter first Clean Components, or of all pixel values above the given Value if /MINFLUX is present.

10.4.1 MODEL /MINVAL

[CALIBRATE\]MODEL [MaxIter] /MINVAL Value [Unit]

Construct the source model using all Clean Components until MaxIter (all if MaxIter is 0 or not specified). These components are stacked on a grid, and then all pixels above the given Value are taken as source model to derive visibilities.

Unit can be Jy, mJy, K or sigma. The default value is Jy.

10.5 SOLVE

[CALIBRATE\]SOLVE Time SNR [Reference]
/MODE [Phase|Amplitude] [Antenna|Baseline] [Flag|Keep]

Solve the baseline or antenna based gains using the current UV data and current MODEL.

Time is the integration time for the solution. SNR is the minimum Signal to Noise Ratio required to find a solution.

10.5.1 SOLVE /MODE

```
[CALIBRATE\]SOLVE Time SNR [Reference]
  /MODE [Phase|Amplitude] [Antenna|Baseline] [Flag|Keep]
```

Dependin on the /MODE arguments, the gains can be antenna-based or baseline-based, and include Phase or Amplitude, and data without solutions either KEEPed or FLAGged,

10.6 UV_SELF

```
  [CALIBRATE\]UV_SELF [CenterX CenterY UNIT [Angle]]
[/RANGE [Min Max Type]] [/RESTORE]
```

Use (and if specified and/or needed create) the "Self Calibrated" UV dataset to make a dirty image, instead of using the current UV table.

UV_SELF utilizes UV_MAP for imaging. See HELP UV_MAP for parameters.

10.6.1 UV_SELF /RANGE

[CALIBRATE\]UV_SELF [CenterX CenterY UNIT [Angle]] /RANGE [Min Max Type]

Create and image the "Self Calibrated" UV data.

The "Self Calibrated" UV dataset is created from the current UV data set by extracting the range of channels specified by the /RANGE arguments Min Max Type. Type can be CHANNEL, VELOCITY or FREQUENCY. If /RANGE has no argument, all channels are averaged together.

It is then updated by command SOLVE at each self-calibration loop. See SOLVE and CLEAN\SELFCAL for details.

10.6.2 UV_SELF /RESTORE

```
[CALIBRATE\]UV_SELF /RESTORE
```

As UV_RESTORE but for the self-calibrated UV table.

Restores the Clean image from the Clean Component Table by removing the components from the Self-calibrated UV data and imaging the residuals before adding them to the convolved Clean components.

```
See UV_RESTORE for details.
```

11 NEWSTUFF Language Internal Help

11.1 Language

```
EXTRACT       : Extract a subset from a data cube
MAP_CONTINUUM : Determine the continuum image from a spectral cube
MFS           : Multi Frequency Synthesis (under development - not functiona
SLICE         : Extract a Slice of the specified data cube
UV_DEPROJECT  : Deproject UV data
UV_PREVIEW    : Quick look at the spectral aspects of the UV data
UV_RADIAL     : Deproject and compute radial average of UV data
UV_SHORT      : Compute and add short spacings to UV data
```

11.2 EXTRACT

```
[NEWSTUFF\]EXTRACT VarName blc1 blc2 blc3 trc1 trc2 trc3
```

Extract the subset VarName[blc1:trc1,blc2:trc2,blc3:trc3] of the image variable VarName and put it in the EXTRACTED image variable.

11.3 MAP_CONTINUUM

```
[NEWSTUFF\]MAP_CONTINUUM [DIRTY|CLEAN [Threshold]]
```

Compute a continuum image from the Dirty or Clean data set. The derivation of the continuum follows the same algorithm than in UV_CONTINUUM, using the integrated spectrum over the image.

To be added: using the Support, and/or the display Size to derive the integrated spectrum

11.4 MFS

```
[NEWSTUFF\]MFS
```

** NOT OPERATIONAL -- UNDER DEVELOPMENT **

Multi-frequency synthesis, allowing to account for spectral index changes across the observed field of view for very wide bandwidths continuum imaging.

11.5 SLICE

```
[NEWSTUFF\]SLICE VarName Start1 Start2 End1 End2 UNIT
```

Cut a slice through the 3-D image variable VarName with the specified edges and put it in the SLICE image variable. UNIT is a keyword indicating in which units the edges are specified. It can be PIXELS, DEGREE, MINUTE, SECOND, RADIAN, of ABSOLUTE. For ABSOLUTE, Start1 and End1 are assumed to be in hours and Start2 End2 in degrees, with sexagesimal notation allowed.

11.6 SELFCAL

SELFCAL [?|AMPLI|APPLY|PHASE|SUMMARY|SHOW [Last [First]] [/WIDGET]

Command to perform Self Calibration (even on spectral line data). The solution is computed, saved and/or applied.

The arguments control the action.

SELFCAL ?	Lists the self calibration parameters
SELFCAL AMPLI	Compute an Amplitude only self calibration
SELFCAL APPLY	Apply the computed solution
SELFCAL PHASE	Compute a Phase only self calibration
SELFCAL SHOW []	Show the computed corrections
SELFCAL SUMMARY	Display the improvements in Noise & Dynamic range and calibration status

11.6.1 SELFCAL /WIDGET

SELFCAL /WIDGET

Activates the widget interface to perform self calibration (even on spectral line data set). A continuum data set is extracted from the specified channel range, with all selected channels averaged to provide improved sensitivity to find a solution.

The buttons control the action.

AMPLI	Perform an Amplitude only self calibration
PHASE	Perform a Phase only self calibration
Continue	Continue execution when the script is in Pause
APPLY	Apply the solution
INPUT	List parameters (as in SELFCAL ?)
SAVE	Save the parameters in selfcal.last
SHOW	Show the computed corrections
SUMMARY	Display the improvements in Noise & Dynamic range

11.6.2 SELFCAL Arguments:

```
  The arguments control the action.
SELFCAL ?         Lists the self calibration parameters
SELFCAL AMPLI     Perform an Amplitude only self calibration
SELFCAL APPLY     Apply the solution
SELFCAL PHASE     Perform a Phase only self calibration
SELFCAL SAVE      Save the parameters in selfcal.last
SELFCAL SHOW []   Show the computed corrections
SELFCAL SUMMARY   Display the improvements in Noise & Dynamic range
```

11.6.3 SELFCAL AMPLITUDE

```
SELFCAL AMPLI
```

Compute an Amplitude only self-calibration. The integration times, SELF_TIME, should in general be significantly larger than for a Phase only self-calibration.

SELFCAL automatically adjusts the gains so that their mean is 1.0, to avoid changing the flux scale.

11.6.4 SELFCAL APPLY

```
SELFCAL APPLY
```

Apply the self calibration solution. This is done only if the return status from the previous computation, SELF_STATUS, is greater than 0. SELFCAL APPLY automatically saves the parameters and results in the "selfcal.last" file.

11.6.5 SELFCAL INPUT

```
SELFCAL INPUT  or SELFCAL ?
```

Display SELFCAL control variables

11.6.6 SELFCAL PHASE

```
SELFCAL PHASE
```

Compute a Phase-only self calibration. The integration time should be short enough to correct for atmospheric errors, but large enough to obtain significant signal to noise for most antennas.

11.6.7 SELFCAL SAVE

```
SELFCAL SAVE
```

Save the parameters and results in the "selfcal.last" file.

11.6.8 SELFCAL SHOW

```
SELFCAL SHOW [Last [First]]
```

Shows the correction computed by self calibration. By default, the difference between the last two iterations is displayed: phase should be around 0, and amplitude around 1 if the solution is converged.

If Last and First are present, it shows the difference between these two specified iteration. If Last only is present, it shows the total correction between the original data and that iteration.

11.6.9 SELFCAL SUMMARY

```
SELFCAL SUMMARY
```

Display a summary of the Self-calibration process: type of calibration, rms and dynamic range at each iteration, as well as the number of flagged or uncalibrated visibilities.

11.6.10 SELFCAL Results:

SELFCAL returns results in several variables, creates a CGAINS array containing the Gain values, and one file to display the computed correction. The file name is specified by SELF_SNAME.

The result variables are:

SELF_APPLIED
  Indicates whether the solution has been applied
SELF_STATUS
  Indicates if a solution has been computed
SELF_DYNAMIC[Self_Loop+1]
  The dynamic range at each iteration
SELF_RMSCLEAN[Self_Loop+1]
  The rms noise at each iteration

11.6.11 SELFCAL SELF_APPLIED

Variable SELF_APLPLIED indicates whether a solution has been applied (#0) or not (0). The value indicates the type and quality of solution, as for SELF_STATUS

11.6.12 SELFCAL SELF_STATUS

Variable SELF_STATUS indicates if a solution has been computed
0 no solution
+/- 1 Phase solution
+/- 2 Gain solution
>0 means a good solution, <0 a poor one.

11.6.13 SELFCAL SELF_DYNAMIC

SELF_DYNAMIC is a variable length array of size Self_Loop+1, containing the dynamic range at each iteration.

11.6.14 SELFCAL SELF_RMSCLEAN

SELF_RMSCLEAN is a variable length array of size Self_Loop+1, containing the Clean map rms noise at each iteration.

11.6.15 SELFCAL Variables:

The SELFCAL behaviour can be adjusted through control variables named SELF_... (see EXA SELF_ for a list), in addition to the control variables of UV_MAP (MAP_...) and CLEAN (CLEAN_...)

The most important variable is SELF_TIMES, a variable length array which controls the integration time at each loop. SELF_NITER and SELF_MINFLUX also control the number of Clean components and minimum flux retained in each loop.

11.6.16 SELFCAL SELF_CHANNEL

SELF_CHANNEL[2]

First and last channel to define the range to compute the UV table for the self-calibration solution. For example, if you have used UV\PLUGC to plug a continuum data into the last channel, this could be set to the

number of channels. 0 0 means all channels are averaged to compute the "continuum" image.

11.6.17 SELFCAL SELF_LOOP

Number of self-iteration loops. It is a ReadOnly variable that is automatically computed from the size of the SELF_TIMES, SELF_NITER and SELF_MINFLUX arrays. Constant arrays are assumed of arbitrary length in this determination. If all 3 arrays have constant values, SELF_TIMES determines the number of loops.

11.6.18 SELFCAL SELF_NITER

Variable length array specifying the number of Clean components retained for each loop of the self-calibration process. Default is 0, meaning all Clean components found by CLEAN.

For simple structures and Phase calibration, 10 may be enough. For more complex ones, be sure to include enough Clean components in the model. More components can be taken at each step, although the better knowledge of phase errors often allows the source to be represented with a smaller number of components after self-calibration than before. The default is a reasonably good compromise, although faster convergence may be obtained with smaller number of components.

11.6.19 SELFCAL SELF_TIMES

Variable length array specifying the integration time (in seconds) for each loop of the self-calibration process.

At NOEMA, 45 sec is the normal minimum integration time. Depending on Signal to Noise, you may need to have this longer, e.g. 120 sec. For several loops, start with a longer value, and decrease only at the end.

At ALMA, the minimum integration time is 6 sec. At VLA, this may be as small as 1 sec.

It is recommended to use the same integration time for the last two iterations, to simplify the interpretation of the results and of the SELFCAL SHOW display.

11.6.20 SELFCAL SELF_MINFLUX

Variable length array specifying the minimum flux density (in Jy/beam) to be considered in the Clean image for each loop of the self-calibration process. Note that this is the brightness of a pixel, not the flux of a Clean component.

11.6.21 SELFCAL SELF_REFANT

The reference antenna number. If 0, the program will peak the one with the shortest average baselines.

11.6.22 SELFCAL SELF_FLUX

Maximum value in the FLUX window. If 0, the FlUX window of Clean is not displayed

11.6.23 SELFCAL SELF_PRECISION

Tolerance to test for self-calibration convergence. The default is 0.01. SELFCAL SUMMARY will write a message when no more selfcal iteration improves the solution (noise and dynamic range) at this precision level.

11.6.24 SELFCAL SELF_RESTORE

Use UV_SELF /RESTORE after Cleaning. This avoids signal aliasing at image edges, and leads to a better estimate of the noise level.

It also allows to use smaller images, in practice,

11.6.25 SELFCAL SELF_DISPLAY

If YES, display Clean image before each calibration loop, and prompt for user input. If YES, Cleaning at each step will use the number of iterations specified by NITER, while if set to NO, Cleaning will stop at SELF_NITER, saving time.

The dynamic range progress report is accurate only if SELF_DISPLAY is set.

11.6.26 SELFCAL SELF_FLAG

If SELF_FLAG is YES, SELFCAL will flag data with no solution. If NO, it will keep the data as it was before.

11.6.27 SELFCAL SELF_SNR

Minimum Signal to Noise ratio on the antenna gain to consider a solution to be valid for an antenna. 6 is a good value, 3 a lower limit. Beware that this SNR makes sense only if the a priori estimate of the noise from the UV weights is correct: see SELF_SNOISE.

11.6.28 SELFCAL SELF_SNOISE

Noise scaling factor. This should be 1, but some noise estimates need corrections. Continuum data from ALMA often requires sqrt(2) instead, for example. Command UV_PREVIEW may help you checking the noise scale.

11.6.29 SELFCAL CLEAN_ARES

Maximum absolute residual to stop Cleaning

11.6.30 SELFCAL CLEAN_FRES

Maximum fractional residual to stop Cleaning

11.6.31 SELFCAL CLEAN_NITER

Maximum number of Clean components. If all of CLEAN_ARES, CLEAN_FRES and CLEAN_NITER are 0, Clean stops by checking the stability of the cleaned flux over CLEAN_NKEEP iterations. See HELP CLEAN for details.

11.7 UV_DEPROJECT

```
[NEWSTUFF\]UV_DEPROJECT x0 y0 Rota Incli
```

Deproject a UV table for inclination and orientation (Incli, Rota, in Degrees) around a specified center (x0,y0 in Radians). This can be useful for almost planar or cylindrical objects (e.g. galaxies, or protoplanetary disks)

The result becomes the current UV table.

11.8 UV_PREVIEW

```
[NEWSTUFF\]UV_PREVIEW [Ntapers [Threshold [NHist]]]
```

A fast previewer to figure out if there is signal and what is its spectral shape. The command attempts to find out the line free regions and to estimate a continuum image.

Optional arguments are

Ntapers:	number of scale sizes (default 4)
Threshold:	Truncation level in Sigma (default 3.5)
Nhist:	Histogram size (default is variable)

11.8.1 UV_PREVIEW Algorithm

UV_PREVIEW computes for Ntapers different tapers the spectrum towards the phase center. The taper ranges are determined from the available baseline lengths and telescope diameter.

For each spectrum, UV_PREVIEW then attempts to figure out if there is line emission and the line-free channels to define the continuum level. This is based on the histogram of the intensity distribution of all channels. The most likely value and the noise level is derived from this histogram. An iterative scheme, blanking out of range (presumably line emission) channels, is used for this to converge towards a Gaussian histogram, which normally represents the noise distribution around the continuum level.

As a last step, blanked channels are accumulated in a list of channels, which thus contain possible line emission at any of the Ntapers scales.

11.8.2 UV_PREVIEW Limitations

UV_PREVIEW cannot identify lines if there are two few channels. It will only display the spectra in this case. A minimum of 32 channels is required, but confused spectra may also prevent a proper line recognition.

11.8.3 UV_PREVIEW Output

UV_PREVIEW returns the list of possible line channels through variable PREVIEW%CHANNELS. If no line emission was identified at any scale, the list is empty and the variable does not exist.

With this list, the user can compute the continuum image, using commands UV_FILTER /CHANNEL PREVIEW%CHANNELS, then UV_CONTINUUM and the usual UV_MAP and CLEAN. Alternatively, the user can filter out the continuum emission using UV_BASELINE /CHANNEL PREVIEW%CHANNELS.

UV_PREVIEW also creates two other variables, PREVIEW%EDGES and PREVIEW%FREQUENCIES which contains the start and end channels (for %EDGES, resp. frequencies for %FREQUENCIES) of each contiguous range of channels in PREVIOUS%CHANNELS. These variables are used for line identification and imaging in the

```
  @ preview_lines
  @ image_lines
```

scripts.

11.9 UV_RADIAL

[NEWSTUFF\]UV_RADIAL x0 y0 Rota Incli [/SAMPLING QSTEP [QMIN QMAX]] [/U_ONLY] [/ZERO [Flux]]

Compute a UV table containing the radial distribution of the azimutal average of the visibilities after deprojection for inclination and orientation (Incli, Rota, in Degrees) around a specified center (x0,y0 in Radians).

The result becomes the current UV table.

If not specified, x0 y0 Rota and Incli default to 0.

11.9.1 UV_RADIAL /SAMPLING

[NEWSTUFF\]UV_RADIAL x0 y0 Rota Incli /SAMPLING QSTEP [QMIN QMAX] [/U_ONLY] [/ZERO [Flux]]

Specify the sampling of the UV distances: Qstep is the step, Qmin and

Qmax the min and max. Distances are in meter. If not present, an automatic guess is made from the minimum and maximum baselines and the dish diameter.

11.9.2 UV_RADIAL /U_ONLY

[NEWSTUFF\]UV_RADIAL x0 y0 Rota Incli /U_ONLY [/SAMPLING QSTEP [QMIN QMAX]] [/ZERO [Flux]]

Indicate that the resulting UV table should have all V values equal to zero. This is convenient to display the azimutal average of the visibilities as a function of UV distance, but cannot be used for further imaging.

If not present, the resulting UV table has a (u,v) coverage which is extended by rotation, so that it can be used to image the (rotationally symmetric) 2-D radial distribution using standard commands like UV_MAP and CLEAN. The radial profile of the brightness distribution can then be recovered by using any radial cut through this image.

11.9.3 UV_RADIAL /ZERO

[NEWSTUFF\]UV_RADIAL x0 y0 Rota Incli /ZERO [Flux] [/SAMPLING QSTEP [QMIN QMAX]] [/U_ONLY]

Add the zero spacing flux to the azimutal average. If no value is given, the zero spacing flux is extrapolated from the shortest baselines using a parabolic interpolation centered on (u,v)=(0,0).

11.10 UV_SHORT

[NEWSTUFF\]UV_SHORT [Arg]

Compute the Short Spacings from the current Single Dish dataset (read by READ SINGLE) and merge it to the current UV data.

UV_SHORT takes sensible default guesses for most parameters. UV_SHORT ? lists the essential parameter values, UV_SHORT ?? some additional ones, and UV_SHORT ??? even the debugging control variables.

The current values can be overriden by the user, who need to set (and if needed to define first) the corresponding SHORT_whatever variable. SHORT_SD_FACTOR is the main one which may need to be specified by the user, as the Single Dish data is rarely in the appropriate

unit.

The resulting UV table becomes the current UV data, and can be imaged, written, etc...

11.10.1 UV_SHORT /REMOVE

UV_SHORT /REMOVE

Removes any short spacing from the current UV data set.

11.10.2 UV_SHORT Algorithm

UV_SHORT task computes pseudo-visibilities for short or zero spacings from a single dish table of spectra (Class table) or LMV data cube. These pseudo visibilities are appended to the current (presumably a Mosaic) UV table.

Short spacings are computed when the Interferometer dish diameter is smaller than the Single-dish diameter, Zero spacings otherwise (see HELP UV_SHORT Zero_Spacing for this case)

For short spacings, the command performs two steps
(1) Creation of a "well behaved" map from the spectra.
(2) Extraction of UV visibilities from this map.

See HELP UV_SHORT Step_i for detailed explanations of the method steps.

With recent UV tables and Single Dish CLASS table, most parameters are automatically determined. The only parameter to be specified remains SHORT_SD_FACTOR (although that one may also be determined automatically if the input single dish data set is in main-beam brightness temperature).

A parameter set to 0 value indicates the appropriate default should be used.

11.10.3 UV_SHORT Zero_Spacing

Zero spacings are computed when the single dish diameter is the same as the interferometer dish diameter. Zero spacing extraction proceeds differently for Class data tables and 3-D data cubes.

In the data cube case, the nearest pixel matching the direction of each field is taken as the Zero spacing for this field. If there is no point close enough, according to the specified position tolerance SHORT_TOLE, an error occurs.

In the Class data table case, all spectra within the specified position tolerance of a field center are averaged together to produce the Zero spacing. If none is found, an error occcurs.

11.10.4 UV_SHORT Step_1

Step (1) Creation of a "well behaved" map from the spectra.

Step (1) only occurs if the input single-dish data set (read by READ SINGLE) is a table of spectra. The table format is described in the CLASS\GRID command of CLASS.
The identification of the input single-dish data set as a table of spectra is currently only based on the specified file extension: if that is .tab, it is assumed to be a table of spectra.

It is recommended that this input table is a collection of single-dish, Nyquist sampled spectra covering twice the interferometric field of view of interest. However, UV_SHORT does *NOT* make any assumption. It thus tries to compute a "well behaved" map by linear operations (convolutions) from the original spectra, in an optimum way from signal to noise point of view. The map is extrapolated smoothly towards zero at the map edge in order to avoid further aliasing in the Fourier transform operations required in Step (2). This extrapolation has a scale length of twice the single-dish beam, in order to avoid spurious Fourier components.

In detail, UV_SHORT performs the following operations:

- Resampling (in space) of the original spectra on a regular grid by convolution with a small (typically 1/4 of the single-dish beam) gaussian convolving kernel. In this process, the weights of individual spectra is carried on a weight map.
- Extrapolation by zero outside the convex hull of the mapped region.
- Convolution of the result by a gaussian twice as wide as the single-dish beam. Within the convex hull of the mapped region, the smoothed map is replaced by the original map.

11.10.5 UV_SHORT Step_2

```
      Step (2) Extraction of UV visibilities from this map.
```

From the given input data cube, or the "well behaved" data cube created by Step (1), UV_SHORT computes the visibilities in the following way:

- Fourier transform of the single dish map;
- Division by the Fourier transform of the single dish beam, up to a maximum spacing (SHORT_SD_DIAM, in meters);
- Inverse Fourier transform to the image plane and then for each pointing center;
- Multiplication of the image by the primary beam of the interferometer elements;
- Fourier transform back to the UV plane;
- Creation of the visibilities, with a given weight SHORT_SD_WEIGHT and an appropriate calibration factor to Janskys SHORT_SD_FACTOR.

Both the single-dish and the interferometer antennas are assumed to have gaussian beams (SHORT_SD_BEAM and SHORT_IP_BEAM, in radians).

11.10.6 UV_SHORT Variables:

Control variables for UV_SHORT are not predefined, except for the 3 main ones: SHORT_SD_FACTOR, SHORT_SD_WEIGHT and SHORT_UV_TRUNC.

All others should be defined by the user in case the default value is not appropriate, with their appropriate (Real, Char or Logical) types.

11.10.7 UV_SHORT SHORT_DO_SINGLE

Logical value, should be YES except for test purposes.

11.10.8 UV_SHORT SHORT_DO_PRIMARY

Logical value, should be YES except for test purposes.

11.10.9 UV_SHORT SHORT_IP_BEAM

Half-power beam width of the interferometer antennas, in radians. The beam is assumed to be gaussian.

Default value is 0, meaning that the beam is taken from the Telescope section if present.

11.10.10 UV_SHORT SHORT_IP_DIAM

Interferometer diameter for which UV_SHORT will compute short spacing visibilities.

Default value is 0, meaning that the diameter is taken from the Telescope section if present.

11.10.11 UV_SHORT SHORT_MCOL

*** Obsolescent ***

See READ SINGLE ClassTable.tab /RANGE command for an equivalent method of selecting the appropriate channel range.

The first and last column to be mapped. For tables produced by GRID command of CLASS, SHORT_MCOL[1] should be 4 and SHORT_MCOL can be set to 0 to process all channels.

Default value: 4 0, appropriate for tables coming from CLASS\GRID command.

11.10.12 UV_SHORT SHORT_MIN_WEIGHT

The minimum weights under which a given point in the map should be filled by the smooth map rather than by the gridded (original] map.

Default value: 0.01

11.10.13 UV_SHORT SHORT_MODE

This is an integer code used for backward compatibility with an older version of the UV_SHORT task, and also for test purpose.

Allowed values are :

-1 indicates to create a single UV table with columns for the Phase center offsets only
-2 indicates to create a UV table with columns for the Pointing center offsets
-3 indicates to create a UV table with the additional columns type being Pointing or Phase, as in the original UV_TABLE$
+1 indicates to append to the initial UV table the short spacings with Phase center offsets (which must thus match the initial UV table shape)
+2 indicates to append to the initial UV table the short spacings with Pointing center offsets (which must thus match the initial UV table shape)
+3 indicates to append to the initial UV table the short spacings (The extra column type being determined automatically).

The default value is 3, i.e. automatic merging with the current UV table.

11.10.14 UV_SHORT SHORT_SD_BEAM

Half-power beam width of the single dish antenna, in radians. The beam is assumed to be gaussian.

Default value is 0, meaning that the beam is taken from the Telescope section if present.

11.10.15 UV_SHORT SHORT_SD_DIAM

Single dish diameter used to produce the input spectra, in meters.

Default value is 0, meaning that the diameter is taken from the Telescope section if present.

11.10.16 UV_SHORT SHORT_SD_FACTOR

Multiplicative calibration factor; it is used to convert from the single dish map units (e.g., main-beam brightness temperature) to janskys.

A default value of 0 can be used if the original data file is in unit of Tmb, the main beam brightness temperature, because in this case the conversion factor can be derived from the beam size.

11.10.17 UV_SHORT SHORT_SD_WEIGHT

Weight scaling factor for the generated visibilities.

It is a relative scaling factor in the weights compared to a supposedly optimal weighting to give the best combined synthesized beam. That optimal weighting essentially gives the same weight density par unit area in the UV plane than the shortest baselines measured with the interferometer only. However, if the single-dish data has not been observed long enough, or has baselines problems for example, this weight may add noise to the overall data set, so could be down-weighted.

Default: 1.0

11.10.18 UV_SHORT SHORT_TOLE

The tolerance in position (in radians). The behaviour differ for Short and Zero spacings and Table or 3-D cubes as Single-Dish data.

If the Single-Dish data is a table of spectra, Spectra differing by less than this amount will be added together prior to gridding. A recommended value is below 1/10th of the Single Dish primary beam. This is valid for Short Spacings and Zero Spacing cases.

If the Single-Dish data is 3-D data cube, SHORT_TOLE is used only for Zero Spacings. If no pixel is within SHORT_TOLE of an Interferometer pointing center, no short spacing is added for this field and an error occur.

Default value is 0, meaning using 1/16th of the Single Dish primary beam.

11.10.19 UV_SHORT SHORT_UV_TRUNC

No visibility at spacings higher than SHORT_UV_TRUNC is generated. Theoretical consideration on the method used in this task implies that SHORT_UV_TRUNC should be at most (SHORT_SD_DIAM-SHORT_IP_DIAM). Smaller values may need to be applied if, for example, the pointing accuracy of the Single Dish is insufficient.

Default value is 0, meaning to use SHORT_SD_DIAM-SHORT_IP_DIAM

11.10.20 UV_SHORT SHORT_WCOL

```
For  tests only: The column of the spectra table containing the weights.

  Default value: 0=3, appropriate for tables coming from CLASS\GRID com-
mand.
```

11.10.21 UV_SHORT SHORT_WEIGHT_MODE

```
The  weighting mode (NATURAL, UNIFORM or GRIDDED).  It is advised to use
'NA' for Natural weighting.
```

11.10.22 UV_SHORT SHORT_XCOL

```
    For tests only: The column of the spectra table  containing  X  off-
sets.

  Default value: 0=1, appropriate for tables coming from CLASS\GRID com-
mand.
```

11.10.23 UV_SHORT SHORT_YCOL

```
    For tests only: The column of the spectra table  containing  Y  off-
sets.

  Default value: 0=2, appropriate for tables coming from CLASS\GRID com-
mand.
```

A Properties of the Fourier Transform

The Fourier Transform of a product of two functions is the convolution of the Fourier Transforms of the functions.

References

Clark, B. G. 1980, A&A, 89, 377

Cornwell, T. & Holdaway, M. 200X

Guilloteau, S. 2000, in IRAM Millimeter Interferometry Summer School, ed. A. Dutrey

Högbom, J. A. 1974, A&A suppl., 15, 417

Pety, J., Gueth, F., & Guilloteau, S. 2001a, ALMA+ACA Simulation Results, Alma memo 387, IRAM

Pety, J., Gueth, F., & Guilloteau, S. 2001b, ALMA+ACA Simulation Tools, Alma memo 386, IRAM

Pety, J., Gueth, F., & Guilloteau, S. 2001c, Impact of ACA on the Wide-Field Imaging Capabilities of ALMA, Alma memo 398, IRAM

Schwab, F. R. 1984, AJ, 89, 1076

Steer, D. G., Dewdney, P. E., & Ito, M. R. 1984, A&A, 137, 159

Wakker, B. P. & Schwarz, U. J. 1988, A&A, 200, 312

Index