# MAPPING

A GILDAS software

March 29th, 2007

Version 2.0

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: J. Pety[1,2].
Active developers: N. Rodriguez–Fernandez[1], S. Guilloteau[3].
Main past contributors: R. Lucas, K. Bouyoucef.

**Software**
In charge: J. Pety[1,2].
Active developers: S. Guilloteau[3], F. Gueth[1], N. Rodriguez–Fernandez[1].
Main past contributors: R. Lucas, K. Bouyoucef.

1. IRAM
2. Observatoire de Paris
3. Observatoire 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** **1**

**2 Basic concepts** **3**
2.1 Overview of the data reduction and analysis . . . 3
2.2 The structure of the MAPPING program . . . 4
2.2.1 Structure and recommendations . . . 4
2.2.2 A bit of history . . . 5
2.3 Cookbook for the impatient ones . . . 5
2.3.1 Imaging and deconvolution pipeline . . . 5
2.3.2 Finely tuned imaging and deconvolution . . . 6
2.3.3 Noise estimation and plotting . . . 7

**3 Interacting with *uv* tables** **9**
3.1 Structure . . . 9
3.1.1 *uv* table . . . 9
3.1.2 *uv* header . . . 10
3.2 Interacting with *uv* table . . . 11
3.2.1 Basics . . . 11
3.2.2 Advanced . . . 12
3.3 Visualization (`GO UVALL` and `GO UVCOV`) . . . 12
3.4 Flagging (`UV_FLAG`, `RUN UV_MFLAG`, `RUN UV_CLIP`) . . . 13
3.5 Fitting (`GO UVFIT` and `GO PLOTFIT`) . . . 14
3.5.1 General considerations . . . 14
3.5.2 Implementation . . . 14
3.5.3 Practical advices . . . 16
3.6 The widget `Operations on uv tables` . . . 16

**4 Single-field imaging and deconvolution** **17**
4.1 Measurement equation and other definitions . . . 17
4.2 Imaging . . . 18
4.2.1 Image size, pixel size, robust weighting and tapering . . . 18
4.2.2 Implementation (`READ UV`, `UV_MAP` and `UV_STAT`) . . . 19
4.2.3 Typical imaging session . . . 21
4.3 Deconvolution . . . 22
4.3.1 Philosophy . . . 22
4.3.2 The family of `CLEAN` algorithms (`HOGBOM`, `CLARK`, `MX`, `SDI`, `MRC`, `MULTI`) . . . 23
4.3.3 Implementation and typical use . . . 26

4.3.4 Comparison and practical advices . . . . . 28

**5 Wide-field imaging and deconvolution 31**
5.1 General considerations about wide-field imaging . . . . . 31
5.2 Mosaicing . . . . . 32
5.2.1 Observations and processing . . . . . 32
5.2.2 Imaging (`UV_MAP` and `RUN MAKE_MOSAIC`) . . . . . 33
5.2.3 Deconvolution (`MOSAIC`, `HOGBOM`, `CLARK` and `SDI`) . . . . . 34
5.2.4 Implementation and typical use . . . . . 34
5.3 Short spacings (`RUN UV_SHORT` and `RUN UV_ZERO`) . . . . . 36
5.3.1 Algorithms to merge single-dish and interferometer information . . . . . 37
5.3.2 Practical considerations . . . . . 40

**6 Internal Helps 43**
6.1 CLEAN Language Internal Help . . . . . 43
6.2 Description of Associated Tasks . . . . . 43
6.2.1 mapping . . . . . 43
6.2.2 UV_AVERAGE . . . . . 44
6.2.3 UV_CASA . . . . . 44
6.2.4 UV_CENTER . . . . . 44
6.2.5 UV_CHECKBEAM . . . . . 44
6.2.6 UV_CIRCLE . . . . . 45
6.2.7 UV_CLIP . . . . . 45
6.2.8 UV_CTIME . . . . . 45
6.2.9 UV_CUTS . . . . . 45
6.2.10 UV_DFT . . . . . 46
6.2.11 UV_FIT . . . . . 46
6.2.12 UV_FITC . . . . . 46
6.2.13 UV_FIT-PROPER . . . . . 47
6.2.14 UV_GETINTERVAL . . . . . 47
6.2.15 UV_HANNING . . . . . 47
6.2.16 UV_INVERT . . . . . 48
6.2.17 UV_LIST . . . . . 48
6.2.18 UV_MERGE . . . . . 48
6.2.19 UV_MFLAG . . . . . 49
6.2.20 UV_MOSAIC . . . . . 49
6.2.21 UV_MULT . . . . . 49
6.2.22 UV_NOISE . . . . . 49
6.2.23 UV_PROPER_MOTION . . . . . 49
6.2.24 UV_REF_GAUSS . . . . . 50
6.2.25 UV_REF_MODEL . . . . . 50
6.2.26 UV_REF_POINT . . . . . 51
6.2.27 UV_ROBUST . . . . . 51
6.2.28 UV_SHORT . . . . . 51
6.2.29 UV_SPLITPOLAR . . . . . 52
6.2.30 UV_SUBTRACT . . . . . 52
6.2.31 UV_STOKES . . . . . 52
6.2.32 UV_TEMPLATE . . . . . 52

6.2.33 UV_TRIM . . . . . 52
6.2.34 UV_UPDATE_FIELDS . . . . . 52
6.2.35 UV_SPLITPOLAR . . . . . 53
6.2.36 UV_VLA_REORDER . . . . . 53
6.2.37 UV_ZERO . . . . . 53
6.2.38 MAP_REFINE . . . . . 53
6.2.39 UV_CAL . . . . . 54
6.2.40 UV_GAIN . . . . . 54
6.2.41 UV_CAL . . . . . 55
6.2.42 UV_GAIN . . . . . 55

**A Properties of the Fourier Transform** **57**

# Chapter 1

# Warning

Rewriting this documentation from scratch made us think about the MAPPING user interface. Since then, we have thus rewritten this interface mostly from scratch. This implies that this documentation is already lagging. We will work to update it. In the meantime, we start to distribute this version because we think it is anyway better (even though far from perfect) than the old one.

# Chapter 2

# Basic concepts

## 2.1 Overview of the data reduction and analysis

Once the data has been acquired by an interferometer such as the IRAM Plateau de Bure Interferometer (PdBI) data, 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.

GILDAS mainly implements the first approach, as the program and the data format used for each step is different. The calibration of PdBI data is done inside CLIC on the PdBI 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 MAPPING on the calibrated *uv* table and deliver mainly an `lmv` spectra cube (2 axes of coordinates and 1 axis of velocity/frequency). Finally, the GREG program implements several tools to visualize and analyze an `lmv` spectra cube as those functionalities are not specific to an interferometer (*e.g.* the user can use them as well on 30m spectra cube). It happens that those visualization and analysis functionalities are also shipped inside the MAPPING program so that the user do not have to change program between the deconvolution and analysis.

The choice of clearly separating calibration and imaging+deconvolution was taken at start of the PdBI, when the limiting number of antenna forbid the use of self-calibration. This choice turned out to be useful for several reasons: 1) It simplified the structure of each program by keeping at the minimum the complexity of the data format required at each step. 2) It ensured clear responsibility between developers. 3) It enabled well-defined break points in data reduction and analysis in steps which are more and more generic: While many points of the calibration algorithms inside CLIC are specific to PdBI 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 MAPPING for imaging and deconvolution and in GREG for visualization and analysis. But the reverse is also true: While calibration of PdBI data should be done inside CLIC, imaging+deconvolution and visualization+analysis can be done in other softwares (*e.g.* MIRIAD, AIPS, AIPS++ newly called CASA for the imaging and deconvolution and KARMA for the visualization and analysis).

With the improvements of PdBI (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. We are now exploring this possibility in MAPPING.

## 2.2 The structure of the MAPPING program

### 2.2.1 Structure and recommendations

The MAPPING 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 from 30m data into PdBI data;
- The visualization and analysis of spectra cubes, mainly by importation of the GREG capabilities;
- A simulator of ALMA continuum observations.

Those steps can be done in the following different ways

- A collection of tasks activated via the `RUN` command;
- A collection of SIC procedures (the `GO` and `INPUT` families) which either wraps the calling of tasks to ease the specifications of their input parameters (*e.g.* `GO CLEAN`) or code a complex series of native MAPPING commands to achieve a well-defined goal (*e.g.* `GO UVALL`);
- A collection of commands, either dedicated to image and deconvolution (the `CLEAN\` language) or implementing basic functionalities (the `SIC\` and `GREG\` families of languages);
- A collection of widgets grouped in the MAPPING main menu to interface most of the above possibilities.

We recommend the following general strategy to select the "right" way of doing one operation

- If a command of the `CLEAN` language implements this operation, always use it even though this possibility may also be implemented as a procedure and/or a task (*e.g.* `CLEAN\UV_MAP`, `GO UV_MAP` and `RUN UV_MAP`). This is because our minimum goal is the maintenance of the `CLEAN` language, not of the tasks (as long as the tasks do not share the same code as the `CLEAN` language).
- If a `GO` procedure or a widget and a task implement this operation, always prefer the use of the procedure or the widget over the use of the task as procedure or the widget were designed to ease the interaction with the task, *i.e.* they wrap the call of the task anyhow.
- If this operation is coded only as a task or a procedure, avoid reinvent the wheel, just use it.

There may be exceptions to those recommendations: Always follow the detailed indications of the manual.

### 2.2.2 A bit of history

Implementation of imaging and deconvolution algorithms inside GILDAS started in the early nighties. The first implementation was made as a collection of independent programs, called tasks in the GILDAS environment, and activated through the `RUN` command in the GRAPHIC program. The main advantage was the ease of programing, the main drawback was the lack of user-friendliness. To tackle this drawback, two different approaches were used.

- First, the calling of tasks were hidden through the call to SIC procedures (the `GO` and `INPUT` families of scripts) whose behaviors was modified by global variables. The activation of the tasks and the development of the SIC procedures happened in a preexisting program, called GRAPHIC which also contained the tools to visualize and analyze the spectra cubes.

- Second, a single, big program, called MAPPING, was developed with flexibility in mind, *e.g.* the possibility to interactively define supports where to search for clean components. The support for deconvolution of mosaics was built only in MAPPING. The same procedure names were used in MAPPING and GRAPHIC to obtain the same look-and-feel.

Up to 2003, there thus were two GILDAS program, *i.e.* MAPPING and GRAPHIC, which offered slightly different services with procedures sharing the same names. The status of the GRAPHIC program was difficult to understand as it shared similarities with the GREG program (which defines all the drawing commands of GILDAS) and with the MAPPING program. We thus decided during the 2003 change of GILDAS architecture to transfer the visualization and analysis capacities of GRAPHIC in GREG and the imaging and deconvolution capacities of GRAPHIC in MAPPING. The GRAPHIC program was deprecated and we made the GREG and MAPPING program able to understand the `graphic` extension of procedure files for backward compatibility. The major drawback of this decision is the fact that we currently have in the same program (*i.e.* MAPPING) both tasks, procedures and commands to do similar but slightly different things. Not much happened following this step due to manpower shortage in the GILDAS team. Our goal in the coming years is to clean this situation by ensuring that both tasks and commands use the same FORTRAN code. Our first main step is to update the documentation.

## 2.3 Cookbook for the impatient ones

This section presents typical deconvolution sessions on a real example available in the GILDAS distribution.

### 2.3.1 Imaging and deconvolution pipeline

```
 1 sic copy gag_demo:demo.uvt 1mm.uvt
 2 let name 1mm
 3 go uvcov
 4 go uvall
 5 input uvall
 6 let xtype weight
 7 go uvall
 8 go image
 9 let type beam
10 go bit
```

```
11 let type lmv
12 go bit
13 let type lmv-res
14 go bit
15 let type lmv-clean
16 go bit
17 exit
```

Comments:

**Step 1** Copy a demonstration *uv* table in the GILDAS distribution in the current directory.

**Steps 2-7** Visualization of different aspects of *uv* data directly from the file. Step 3 shows the *uv* coverage. Step 4 displays the scatter plots of the amplitude vs spatial frequency of the *uv* visibilities (default of the `GO UVALL` procedure). Step 5 displays the value of the parameters which modify the behavior of the `GO UVALL` procedure. Steps 6 and 7 display the scatter plots of the amplitude vs weight of the *uv* visibilities.

**Step 8** Imaging and deconvolution pipeline with visualization of the deconvolved image using default parameters. The result is clearly not optimal in this case (see next section). All the results are directly written as files on disk with the following file name conventions: dirty beam: `1mm.beam`, dirty image: `1mm.lmv`, clean image: `1mm.lmv-clean`, clean residuals: `1mm.lmv-res`.

**Step 9-14** Successive visualization of the dirty beam (steps 9 and 10), dirty image (steps 11 and 12), clean residuals (steps 13 and 14) and clean image (steps 15 and 16).

### 2.3.2 Finely tuned imaging and deconvolution

```
 1 lut rainbow3
 2 read uv 1mm
 3 uv_show
 4 uv_stat weight
 5 input uv_map
 6 let weight_mode UN
 7 let uv_cell 7.5 1
 8 uv_map
 9 show beam
10 show dirty
11 input clean
12 let niter 1000
13 hogbom /flux 0 0.6
14 show residual
15 let niter 2000
16 hogbom /flux 0 0.6
17 show residual
18 let niter 4000
19 hogbom /flux 0 0.6
20 show residual
21 show clean
```

```
22 support
23 hogbom /flux 0 0.6
24 show residual
25 show clean
26 write beam 1mm
27 write dirty 1mm
28 write clean 1mm
29 write residual 1mm
30 write cct 1mm
31 exit
```

Comments:

**Step 1** Select a color lookup table which nicely displays the features of the studied source.

**Step 2** Read *uv* data from the `1mm.uvt` file to an internal MAPPING buffer.

**Step 3** Displays the scatter plot of the amplitude vs spatial frequency of the *uv* visibilities. This commands is similar to the `GO UVALL` command (see previous section), except that it works only on the data previously loaded in the internal buffer.

**Step 4** Predicts the synthesized beam, expected noise level, and recommended pixel size for different values of the robust weighting threshold. This helps the user to select the threshold used in the imaging steps (2nd parameter of the `uv_cell` variable).

**Steps 5-10** Compute a tailored dirty beam and dirty image. Step 5 displays the SIC variables that customizes the behavior of the `UV_MAP` command. Step 6 and 7 selects robust weighting (instead of the default natural weighting) and the associated threshold. Step 8 actually computes the results which are stored in internal buffers and visualized in steps 9 and 10.

**Steps 11-14** First deconvolution on internal buffers. Resulting clean residuals, clean image and clean component tables are also stored in internal buffers. Step 11 displays the SIC variables that customizes the behavior of all the clean deconvolution algorithms. Steps 12 select the stopping criterion by enabling a maximum of 1000 clean components. Step 13 launches the deconvolution using the simplest `CLEAN` algorithm with simultaneous plot of the cumulative flux as a function of the number of found clean components. Step 14 displays the residuals, *i.e.* remaining undeconvolved signal.

**Steps 15-24** Successive tries of the deconvolution to ensure deep enough cleaning, just by changing the stopping criterion (here the total number of clean components).

**Step 25** Displays the resulting clean image.

**Steps 26-30** Write the results (dirty beam, dirty image, clean image, clean residuals and clean component table) on disk files for use in future sessions.

### 2.3.3 Noise estimation and plotting

```
1 let name 1mm
2 let type lmv-clean
3 go cct
```

```
4 go noise
5 let spacing '3*noise'
6 go bit
7 hardcopy 1mm-clean
8 exit
```

Comments:

**Steps 1-3** Visualize the cumulative flux as a function of the number of found clean components to get an idea of the cleaning convergence.

**Step 4** Computes an experimental noise value which takes into account possible remaining side lobes after deconvolution.

**Step 5** Sets the spacing between contour levels to 3 times the experimental noise value (the `noise` SIC variable was defined by the `GO NOISE` procedure).

**Step 6** Visualizes the clean image with the right values for the contour levels.

**Step 7** Makes a color Post-Script file, named `1mm-clean.eps`, of the plot.

# Chapter 3

# Interacting with $uv$ tables

## 3.1 Structure

A $uv$ table is a file which usually contains calibrated visibilities ready for imaging and deconvolution. It thus is the main input of the MAPPING program. The $uv$ table format is a particular example of the Gildas Data Format (GDF[1]). It contains a header plus a table, *i.e.* an array of dimension two composed of lines and columns.

### 3.1.1 $uv$ table

In a $uv$ table, each line describe one and only one visibility. The number of lines of the table is thus the number of visibilities described in the table. Each column of the table stores a particular property of the visibilities, namely:

**Line 1** U in meters;

**Line 2** V in meters;

**Line 3** Scan number;

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

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

**Line 6** Number of the first antenna used to measure the visibility;

**Line 7** Number of the second antenna used to measure the visibility;

**Line 8** Real part for the first frequency channel;

**Line 9** Imaginary part for the first frequency channel;

**Line 10** Weight for the first frequency channel;

[1] A binary format made of a header plus an array whose dimension is between 1 and 4

[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...

**Lines 11-13** Same as column 8-10 but for the second frequency channel;

**...**

If a *uv* table describes `nvis` visibility spectra composed of `nchan` frequency channels, the size of the table will thus be: `nvis` lines of `7+3*nchan` columns.

### 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. The `HEADER` is the standard way inside GILDAS to display in a human readable way the header of GDF file. For instance, `MAPPING> header 12cs21.uvt` would display

```
 1  W-GDF_RHSEC,  Absent section NOISE
 2  File : 12cs21-1.uvt                                              REAL*4
 3  Size        Reference Pixel           Value                  Increment
 4     154   25.50000000000000       97980.02890200325     -6.5365856842222472E-02
 5    1320   0.000000000000000       0.000000000000000       1.000000000000000
 6       1   0.000000000000000       0.000000000000000       0.000000000000000
 7       1   0.000000000000000       0.000000000000000       0.000000000000000
 8  Blanking value and tolerance      1.23455997E+34   0.0000000
 9  Source name         HORSEHEAD
10  Map unit            Jy
11  Axis type           UV-RAW       RANDOM       UNKNOWN      UNKNOWN
12  Coordinate system   EQUATORIAL
13  Right Ascension     05:40:53.903        Declination        -02:28:22.00
14  Lii        206.9592680349889            Bii      -16.80061668685155
15  Epoch              2000.0000
16  Projection type     AZIMUTHAL           Angle     0.000000000000000
17  Axis 0     A0       05:40:54.270        Axis 0     D0      -02:28:00.00
18  Minimum            0.0000000      at     0     0     0     0
19  Maximum            0.0000000      at     0     0     0     0
20  Axis 1 Line Name    12CS(2-1)           Rest Frequency   97980.95299999999
21  Resolution in Velocity   0.20000000     in Frequency   -6.5365856842222472E-02
22  Offset in Velocity        10.500000     Image Frequency  0.000000000000000
23  Noise
24  Beam                     2.494E-04   0.00       0.00
```

Comments:

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

**Lines 3-7** Display the dimensions of the associated array. Here it is a rank 2 array of dimension 1320 columns times 154 lines, *i.e.* 1320 visibility spectra of 49 frequency channels (see previous section). 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 49 frequency channels of width -65.36 kHz, the frequency of the reference pixel 25.50 corresponding to 97980.02890200325 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 10** Indicates the unit of the real and imaginary parts of the visibilities. *** Weights unit? JP ***

**Line 11** Indicates that this is *uv* table (`UV-RAW` and `RANDOM`).

**Lines 12-15** Describe the coordinate system.

**Lines 16-17** 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 18-19** Unused for *uv* tables.

**Lines 20-22** Describe additional information about the frequency axis of the visibility spectra. In particular, the rest frequency (here 97980.953 MHz) corresponding to a velocity of 10.5 km/s in the velocity frame (in general LSR).

**Line 23** Unused for *uv* tables.

**Line 24** Indicates the primary beam size of the interferometer in radian (This is an indirect way to pass the size of the interferometer antennas).

## 3.2 Interacting with *uv* table

### 3.2.1 Basics

A *uv* table being a particular type of the GDF format, all the SIC and GREG commands can be used on *uv* tables. Here is a typical example:

```
 1 transpose 12cs21.uvt 12cs21.tuv 21
 2 column x 1 y 2 /table 12cs21.tuv
 3 limits
 4 box
 5 points
 6 define image tuv 12cs21.tuv read
 7 limits /var tuv[1] tuv[2]
 8 box
 9 points uv[1] uv[2]
10 define double mjd /like tuv[1]
11 let mjd tuv[4]+tuv[5]/24.
12 delete /var tuv
13 define image tuv 12cs21.tuv write
14 for ichan 1 to (tuv%dim[2]-7)/3
15    let tuv['7+3*ichan'] -tuv['7+3*ichan'] /where tuv['5+3*ichan',].ge.100.0
16 next ichan
17 delete /var tuv
```

Comments:

**Step 1** Transposes columns and lines of the *uv* table. Indeed, CLIC naturally produces the *uv* tables in visibility order, which is not well suited for display where it is more useful to have all information about a visibility in a line than in a column. By convention, the extension of transposed *uv* tables is `tuv`.

**Steps 2-5** Read and plot the $u$ and $v$ coordinate to give an idea of the *uv* coverage of the data inside the table.

**Steps 6-12** Steps 6 to 9 gives the same result as steps 2 to 5 except than here all the table is red and stored in the sic `tuv` variable. This enables computation of complex quantities from several table column, *e.g.* the computation of a continuous variable of time (mjd) in steps 10 and 11.

**Steps 13-17** Change the sign of the weight columns for all visibilities whose real amplitude is greater or equal to 100.0. Changing the sign of the weight is a reversible way to flag out a visibility. Indeed, it is always possible to change again the sign. This operation is done in place, *i.e.* the input file is overwritten in step 17 as announced by the `write` option of command `define` in step 13.

### 3.2.2 Advanced

Finally, it is sometimes useful to order the *uv* table either in ascending, negative V values or in ascending time-baseline order. This can be done through the `UV_SORT` task.

## 3.3 Visualization (`GO UVALL` and `GO UVCOV`)

Although images are a wonderful way to display results, problems may be more obvious in scatter plots of visibilities. For instance, large strips in the dirty image often originates from a single outlier visibility. `GO UVALL` is a procedure to explore the content of a *uv* table through visualization of scatter plots. This procedure is controlled by several input parameters, in particular the following global variables

`NAME` The filename without extension of the input *uv* table;

`XTYPE` **and** `YTYPE` The types of data to be plotted along the X and Y axes. These two variables can take the following values

`u` **or** `v` Fourier coordinates [m];

`radius` *uv* distance ($\sqrt{U^2+V^2}$) [m];

`angle` *uv* position angle (`atan2(V,U)`) [deg];

`time` UT time of observation [hrs];

`date` Date of observation [days];

`scan` Scan number;

`number` Visibility number in *uv* table;

`amp` Amplitude of visibility [Jy];

`phase` Phase of visibility [deg];

`real` Real part of visibility [Jy];

`image` Imaginary part of visibility [Jy];

`weight` Weights of visibility [*** ? JP ***].

The default is to plot the amplitude vs the radius of the visibilities.

`FIRST` **and** `LAST`, *i.e.* the first and last channel to plot. By default, all channels are plotted.

Other input parameters are available. Please type `input uvall` to obtain the most recent available help. For instance, to obtain the scatter plots of amplitude vs weight of the visibility (which should look like an hyperbole), you would type:

```
1 let name 12cs21
2 let xtype weight
3 let ytype radius
4 go uvall
```

The plot of the *uv* coverage of the data is such an important subset of the `GO UVALL` possibility that we created a dedicated procedure for it, namely `GO UVCOV`.

## 3.4 Flagging (UV_FLAG, RUN UV_MFLAG, RUN UV_CLIP)

The experience shows that flagging out PdBI data inside MAPPING rarely improve the results. This is the consequence of several factors: 1) The on-line PdBI system quite efficiently catch possible observations problems (*e.g.* phase unlock); 2) A particular care has been taken in the computation of the relative weights so that data taken in poor observing conditions will almost not affect the result. Finally, it is extremely difficult to flag data based on the data itself and a *uv* table is an extremely compact data format which has lost most of the information needed to make intelligent flagging. This is why we recommend the use of tools like the "quality assessment tool" in CLIC which selects good on-source data based on observing conditions (*e.g.* weather, tracking, ...). This said, MAPPING offers several tools to enable data flagging in the few cases where it can be useful like the presence of a single clearly outlier visibility.

`UV_FLAG` **command** Display *uv* data (controled by similar variables than `GO UVALL`) and calls the cursor to interactively select a region where *uv* data will be flagged. The data are flagged by changing the sign of the weight. Subsequent mapping with `UV_MAP` will ignore flagged data. The flagged UV table can be saved on a file with command `WRITE UV` File. The `/RESET` option can be used to unflag the data (all the flagged data, not interactive).

`UV_FLAG` **task** Flag or unflag a specified time range in a *uv* data set, for one or all baselines and a specified channel range. *HOW CAN WE SEE EASILY THE TIME ?*

`UV_CLIP` **task** Clip *uv* data with amplitudes higher than a given threshold. For spectroscopic data, all channels are flagged if any channel between the first and last considered has an amplitude greater than the trheshold. Clipped channels are set to zero amplitude and zero weight. **It doesn't seem to work. First due to old data format, but after using uvt_convert it does not work neither**

`UV_MFLAG` **task** Compare the PdBI uv table with an uv table calculated from a model source and flag the data if the REAL part of the visibility differs from the model in more that a given tolerance. Data are flagged setting weights to 0. *NOT TESTED*

## 3.5 Fitting (`GO UVFIT` and `GO PLOTFIT`)

### 3.5.1 General considerations

The first output of a millimeter interferometer is a set of calibrated visibilities in the *uv* plane. Scientific analysis of this kind of data is possible in two different ways

**Analysis in the image plane** , *i.e.* analysis on images which are made from the *uv* visibilities. This is the most common kind of analysis but it implies (complex) mathematical transforms (*i.e.* imaging and deconvolution) which may bring some artifacts in the results. This is the subject of the next chapter.

**Analysis in the *uv* plane** , *i.e.* fitting of a source model through the *uv* visibilities. When possible, it is the best analysis mode because it avoids the imaging and deconvolution steps. For instance, this enables the precise determination of the properties of proto-planetary disks (geometry, Keplerian rotation, etc...). However, the fit process implies the use of an underlying model of the source. This limits the use of this analysis mode to the "simplest" objects. It must also be used with much critical sense as the use of a wrong fitting model may largely bias the result: *e.g.* trying to fit an circular Gaussian through an elliptical one will give a biased full width at half maximum. In practice, a really important use of this analysis mode is the fit of simple models (point, Gaussian, etc...) through unresolved (or slightly resolved) sources, in particular at low signal-to-noise ratio[3]. MAPPING offers tools dedicated to this special goal.

### 3.5.2 Implementation

The MAPPING fitting capabilities are available through the `GO UVFIT` and `GO PLOTFIT` procedures and/or the associated widgets (inside the `UV interferometric operations` widget of the MAPPING menu). These procedures allow you to fit models directly through the visibilities of a *uv* table whose filename (without extension) is stored into `NAME` sic variable. The models are either simple functions or linear combinations of simple functions. The residual *uv* visibilities will be written in a *uv* table named `'NAME'"-res.uvt"`. The results (*e.g.* values of the fitted parameters) will be saved in an output binary file (in detail, a `GDF` table), named `'NAME'".uvfit"`. The velocity/frequency channels to be fitted are selected through the `FIRST` and `LAST` sic variables.

All other control parameters of the fit and of the plot of the fitted parameters are stored in the `UVFIT` sic structure

**Fit parameters** are

`UVFIT%RANGE` The range (in meters) of spatial scales to be used in the fitting. `"0 1700"` is the correct default as of today (2023) for NOEMA.

`UVFIT%NF` Number of simple fitted functions. Both the task and the procedure enable the fit of the linear combinations of up to four simple functions while the widget gives support only for the combination of two simple functions. Indeed, the larger the number of simultaneously fitted function , the more difficult the convergence... Each of the following parameters is suffixed with the number of the fitted function it contributes to describe (*e.g.* 01, 02, 03, 04).

[3]Indeed, low signal-to-noise ratio is a condition where imaging and deconvolution steps are very delicate.

UVFIT%FUNCT01 Currently supported functions are

| | | |
|---|---|---|
| POINT | Point source | X and Y offset, flux |
| E_GAUSS | Elliptical Gaussian | X and Y offset, flux, major and minor FWHM, pos. ang. |
| C_GAUSS | Circular Gaussian | X and Y offset, flux, FWHM |
| C_DISK | Circular disk | X and Y offset, flux, diameter |
| E_DISK | Elliptical (inclined) disk | X and Y offset, flux, axes (major and minor), pos. ang. |
| RING | Annulus | X and Y offset, flux, inner and outer diameter |
| U_RING | *** ? JP *** | X and Y offset, flux, *** ? JP *** |
| EXPO | Exponential brightness | X and Y offset, flux, FWHM axis |
| POWER-2 | $B = 1/r^2$ | X and Y offset, flux, FWHM axis |
| POWER-3 | $B = 1/r^3$ | X and Y offset, flux, FWHM axis |

The number and description of the fitted parameters depends on the kind of fitted functions. As can be seen, at least 3 parameters are always fitted and the first 3 parameters are common to all fitted functions. Fluxes are expressed in Jansky and sizes in arcseconds.

UVFIT%PARAM01 Guesses of input parameters in the order defined in the above table. Six parameters must always be given. You can put whatever value for the parameters that will not be used in the fit. *Important warning: The guess of spatial length (*e.g. *major, minor) must be non-zero valued, otherwise it will stay zero during the fit...*

UVFIT%START01 Number of time the fit will be started with slightly different input parameters in agreement with the value of UVFIT%RANGE01. The default (*i.e.* 0) means that the fit will be started only once with the values of UVFIT%PARAM01 as initial guess. The value of -1 is an important particular case. Indeed -1 implies that the associated parameters will *not* be fitted. It will instead keep the initial guess value.

UVFIT%RANGE01 Not fully clear *** ? JP ***

UVFIT%SUBSF01 Logical variable indicating whether the associted fitted function will be subtracted from the input *uv* table in the residual *uv* table.

**Plot parameters** are

UVFIT%NP Number of fitted functions for which the values of the fitted parameters will be plotted. Each fitted function will be associated one color.

UVFIT%ORDER Order in which the fitted functions will have their parameters plotted. For instance, if you want to plot only the parameters of the second fitted function, you would set UVFIT%NP to 1 and UVFIT%ORDER to "2 1".

UVFIT%NX **and** UVFIT%XTYPE Number and type of the fitted parameters used to define the X axes of the plots. Up to 6 parameters can be plotted in the same plot. The UVFIT%XTYPE must be a string formatted like "type min max". Possible type are CHANNEL, VELO, FREQ, RMS, RA, DEC, FLUX, WIDTH, MAJOR, MINOR, and ANGLE. A [*] instead of min and/or max implies the use of the default minimum and maximum values of the plotted parameter.

UVFIT%NY **and** UVFIT%YTYPE Idem for the Y axes.

A known bugs is that when you change the number of simultaneously fitted functions, you need to manually remove the uvfit file to avoid trouble.

The procedures are in fact wrap-up around the UV_FIT task. It uses the public domain SLATEC library which is shipped by default with GILDAS.

The results are stored in a GDF table which can read with the `COLUMN` command. The table is organized as a $M \times N$ matrix. $M$ is the number of channels in the input table. The organization along the other axis is as follows `P1 P2 P3 Vel A1 A2 A3 Par1 Err1 Par2 Err2 ... A1 A2 A3 Par1 Err1 ...` where

`P1` RMS of the fitting process.

`P2` Number of simultaneously fitted functions.

`P3` Total number of fitted parameters.

`Vel` Velocity of the ith channel (i lies between 1 and M).

`A1` Number of the fitted function (1 for the first function, 2 for the second function, ...).

`A2` Code of the function kind (POINT = 1, E_GAUSS = 2, ...).

`A3` Number of fitted parameters in the current function.

`Par1` Value of the first fitted parameters.

`Err1` Uncertainty on the value of the first fitted parameters.

Hence, $N = 19$ when fitting a single function (4 generic columns + 3 columns associated to the fitted function + 6 times 2 columns per parameters) and $N = 34$ when fitting simultaneously two functions. This data format as the use of the task itself is not very convenient. We thus recommend to use the procedures and/or the associated widgets.

### 3.5.3 Practical advices

- Always make an image first. This is mandatory if you have no idea of the structure of your source but this is also useful since the image will directly show you the calibration problems, which are in general much more difficult to detect directly in the visibilities.
- Use any knowledge you can have on your source. For instance, fitting a disk function is probably the most appropriate for barely resolved planets and satellites.
- If your source is barely resolved and you have no *a priori* knowledge of the source structure, you should start with the simplest possible model (*e.g.* `POINT`) and complexify the model only gradually (*e.g.* `C_GAUSS`, `E_GAUSS`, etc...). Verify how the uncertainties on the fitted parameters evolves to decide which model is most appropriate.
- Always be critical with the results. In particular, image and deconvolve the residual *uv* table. You should obtain just noise if the fit succeeded.

## 3.6 The widget `Operations on uv tables`

Fig. ??? shows the widget interface which has been designed around the most usual operations that a user will do on a *uv* table. These includes: flagging out of outlier visibilities, scatter plots of *uv* table data, translation and rotation of coordinates, fits of simple geometry in the *uv* plane with the visualization of the fitted parameters.

# Chapter 4

# Single-field imaging and deconvolution

## 4.1 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)

$$V(u,v) = \mathrm{FT}\left\{B_{\mathrm{primary}}.I_{\mathrm{source}}\right\}(u,v) + N \tag{4.1}$$

where $\mathrm{FT}\left\{F\right\}(u,v)$ is the bi-dimensional Fourier transform of the function $F$ taken at the spatial frequency $(u,v)$, $I_{\mathrm{source}}$ the sky intensity distribution, $B_{\mathrm{primary}}$ the primary beam of the interferometer (*i.e.* 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 multiplication of the sky intensity by the primary beam implies a distortion of the information about the intensity distribution of the source.

3. The presence of the Fourier transform implies that visibilities belongs to the Fourier space while most (radio)astronomers are used to work in the image space. A step of *imaging* is thus required to go from the $uv$ plane to the image plane.

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.

To show that deconvolution techniques are needed to overcome the incomplete sampling of the $uv$ plane, we need additional definitions

- Let's 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. $\sigma$ 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

$$I_{\rm dirty} = {\rm FT}^{-1}\{S.V\}\,. \tag{4.2}$$

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

$$I_{\rm dirty} = B_{\rm dirty} * \{B_{\rm primary}.I_{\rm source}\}\,, \tag{4.3}$$

where $*$ is the convolution symbol. Thus, the incompleteness of the $uv$ sampling translate in 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 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.

## 4.2 Imaging

### 4.2.1 Image size, pixel size, robust weighting and tapering

The process known as *imaging* consists in computing the dirty image and the dirty beam from the measured visibilities and the sampling function. This is done through Fast Fourier Transform, which implies first a stage of gridding of the visibilities on a regular grid in the $uv$ plane.

#### Link between image size and $uv$ cell size

The gridding requires at least Nyquist sampling of the $uv$ plane to avoid the artifact known as aliasing. For the signal, the Nyquist sampling in the $uv$ plane is obtained with a size of the grid cells equal 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 bounded: Noise aliasing can not be avoided, *i.e.* the noise increases at the edges of the produced image.

#### 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 process while deconvolution techniques are non-linear process. We thus recommend to pick a pixel size between 1/3 and 1/4 of the synthesized beam to ease the deconvolution. Unless you have good reasons, you should not choose too large an image size nor too small a pixel size at least because it would considerably slow down any deconvolution algorithm.

**Weighting**

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 of the imaging process. In particular, the user may change this scaling to give more or less weight to the large/short spatial frequencies.

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

$$B_{\rm dirty} = {\rm FT}^{-1}\left\{W.S\right\} \tag{4.4}$$

and

$$I_{\rm dirty} = {\rm FT}^{-1}\left\{W.S.V\right\}. \tag{4.5}$$

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 that the cells of the inner $uv$ plane (just because there are less cells in the inner $uv$ plane). Robust weighting thus increase the spatial resolution by emphasizing the large spatial frequencies at the cost of a worst sensitivity to point sources...

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

$$W = \exp\left\{-\frac{\left(u^2+v^2\right)}{t^2}\right\}, \tag{4.6}$$

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 almost never be used as this somehow implies that you throw away large spatial frequencies measured by the interferometer... In other words, use compact configuration of the arrays and not tapering to increase sensitivity to extended structures in your source.

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

### 4.2.2 Implementation (`READ UV`, `UV_MAP` and `UV_STAT`)

In MAPPING, gridding in the $uv$ plane and computation of the dirty beam and image are coded in the `UV_MAP` command. This commands works on an internal buffer containing the $uv$ table read from a file through the `READ UV` command. The `UV_MAP` behavior is controlled through the following sic variables

**Gridding kernel**

`CONVOLUTION` Gridding in the *uv* plane is done through convolution and this variable is the kind of convolution kernel. Default is 5, meaning spheroidal functions. Unless you are a real expert, you should never change the value of this variable: Spheroidal functions are the right answer.

**Data cube setup**

`WCOL` Number of the used weight column. The default is to compute only one dirty beam for the whole data cube. This default may be incorrect when adding data obtain at different rest frequencies.

`MCOL` First and last velocity/frequency channels to map (default value is 0 meaning all the channels).

`MAP_CELL` Pixel size in arcsec. Default is 0, meaning between 1/3 and 1/4 of the synthesized beam.

`MAP_SIZE` Map size in pixel. Default is 0, meaning a power of two which gives about twice the size of the primary beam size. For efficiency of FFT, it is better to pick a power of two.

`MAP_RA` Right Ascension of the phase center. Default is an empty string, meaning do not change the right ascension.

`MAP_DEC` Declination of the phase center. Default is an empty string, meaning do not change the declination.

`MAP_ANGLE` Map position angle in degree. Default is 0, meaning do not rotate the map.

`UV_SHIFT` Logical which toggle the change of phase center and position angle. This may be used to align the map axes with a feature of interest of the source, by rotation of the *uv* coordinates.

**Weighting**

`WEIGHT_MODE` Kind of weighting. Possibilities are `NATURAL` or `UN` (which stands for `ROBUST`...). Default is `NATURAL` weighting to maximize the point source sensitivity.

`UV_CELL` Size of the *uv* plane cell and threshold of the robust weighting. Default are 7.5 m (half the diameter of the PdBI antenna) and 1.

**Tapering**

`UV_TAPER` Properties of the Gaussian used to apodize the weights of the *uv* visibilities (minor and major axes in meters and position angle in degree). Default is 0, meaning no tapering.

`TAPER_EXPO` *** ? JP *** Default is 0, meaning the use of a Gaussian as tapering function.

The current values of all those parameters may be found through the `INPUT UV_MAP` command. The main difficulty in the use of the robust weighting and tapering is the choice of the right parameters for respectively `UV_CELL[2]` and `UV_TAPER`. MAPPING offers the `UV_STAT` command to guide your choice.

```
 1 MAPPING> read uv demo.uvt
 2 MAPPING> uv_stat weight
 3 Robust      Major    Minor     PA            Noise      Sidelobe
 4               (")      (")     (deg)    (mJy)     (K)       %
 5    0.10      0.780    0.479     19.8     0.988    0.066  -21.1  18.3
 6    0.18      0.784    0.485     19.5     0.889    0.058  -20.9  19.2
 7    0.32      0.785    0.493     19.3     0.798    0.051  -20.4  20.2
 8    0.56      0.786    0.503     19.3     0.723    0.045  -19.7  21.5
 9    1.00      0.784    0.509     19.4     0.672    0.042  -19.3  23.8
10    1.78      0.789    0.507     18.3     0.632    0.039  -19.2  26.3
11    3.16      0.820    0.505     14.8     0.601    0.036  -16.0  30.2
12    5.62      0.828    0.525     13.1     0.586    0.033  -13.0  32.3
13   10.00      0.851    0.539     13.2     0.577    0.031  -11.9  34.2
14    None      0.882    0.559     13.2     0.568    0.029  -10.4  38.2
15 MAPPING> uv_stat taper
16 Taper       Major    Minor     PA            Noise      Sidelobe
17  (m)          (")      (")     (deg)    (mJy)     (K)       %
18   30.00      6.586    3.533     10.6     1.727    0.002  -58.8  26.5
19   42.43      5.724    2.896     11.7     1.385    0.002  -36.8  25.8
20   60.00      5.100    2.485     12.8     1.178    0.002  -31.4  24.5
21   84.85      4.623    2.279     13.2     1.035    0.002  -28.8  22.6
22  120.00      3.956    2.058     13.2     0.895    0.003  -26.5  23.4
23  169.71      2.173    1.565     21.8     0.752    0.005  -23.1  27.8
24  240.00      1.161    1.072    161.1     0.646    0.013  -19.8  32.4
25  339.41      0.962    0.756     11.9     0.594    0.020  -17.5  35.1
26  480.00      0.910    0.645     13.2     0.576    0.024  -16.1  36.8
```

Comments:

**Line 1** Read a *uv* table in an internal buffer as the `UV_STAT` works only on this buffer.

**Lines 2-14** Displays the synthesized beam (major, minor, position angle), the noise levels (point source in mJy and brightness in K) and the level of the sidelobes as a function of the threshold (1st column) to be used in the robust weighting method. The usual way to choose the right threshold is to select the one that gives a substantially better spatial resolution without increasing the noise by more than 10%.

**Lines 15-26** Displays the same parameters as above as a function of the tapering distance (1st column) in meter.

### 4.2.3 Typical imaging session

```
1 read uv demo
2 input uv_map
3 uv_stat weight
4 let weight_mode "UN"
5 let uv_cell[2] 0.1
6 input uv_map
7 uv_map
8 show beam
```

```
 9 show dirty
10 let map_size 256
11 uv_map
12 show beam
13 show dirty
14 go wedge
15 hardcopy demo-dirty
16 write dirty demo
17 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 3-6** Select the robust weight mode (step 4) and threshold (step 5) from the result of the `UV_STAT` command (step 3) and recheck the current state of the variables that control the imaging (step 6).

**Steps 7-9** Image and plot the dirty beam and image.

**Steps 10-13** Tries a larger size of the map as the default imaged field-of-view looked too small from previous plots.

**Steps 14-15** Make a Post-Script file from the dirty image.

**Steps 16-17** Write dirty image and beam in `demo.lmv` and `demo.beam` files for deconvolution in a future MAPPING session. These steps are optional as you may directly proceed to the deconvolution stage without writing the files.

## 4.3 Deconvolution

### 4.3.1 Philosophy

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. But, more fundamentally, 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 millimeter interferometer and it will also be for the extended configuration of 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 exclude part of the sky region really emitting. Support must be thus used with caution.

### 4.3.2 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).

#### `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":

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

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. A good compromise is to clean up to or slightly below (typically $0.5\sigma$) the noise level.

### 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 reintroduce the finite resolution of the observation, 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 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 matches). 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.

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.

### 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.

$\gamma$ is the loop gain. It controls the convergence of the method. In theory, $0 < \gamma < 2$. High values of $\gamma$ 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. 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.

MX The MX 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 enable to clean the largest region of the dirty map. However, this is a relatively slow algorithm because the imaging step must be redone at each major cycle.

**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 has been devised to answer 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 is the fraction used for the selection of 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`

### 4.3.3 Implementation and typical use

#### Implementation

In MAPPING, 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

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

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

#### Stopping criteria

`NITER` Maximum number of clean components. Default is 500.

`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 is 0.

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

#### Support

`BLC` **and** `TRC` Bottom Left Corner and Top Right Corner of a square support in pixel units. Default is 0, meaning do not use such a square support.

`SUPPORT` A command to interactively define the support where to search for clean components. 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 support definition is stored in the `POLY` sic structure. A side effect is that any definition of a polygon (*e.g.* through the `GO FLUX` command) will imply the definition of a support...

#### 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 parameter is dangerous.

Other variables control specific aspects of a subclass of the `CLEAN` algorithms:

`NMAJOR` 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 all algorithms using the minor cycle notion. Default is 0, meaning *** ? JP ***

`SMOOTH` *** ? JP ***

**Typical deconvolution session**

```
 1 read beam demo
 2 read dirty demo
 3 input clean
 4 hogbom /flux 0 1
 5 show residual
 6 show clean
 7 write clean demo
 8 let name demo
 9 go noise
10 let ares 0.5*noise
11 input clean
12 hogbom /flux 0 1
13 let niter 2000
14 input 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 show clean
26 write residual demo
27 write clean demo
28 write cct demo
29 let name demo
30 let type lmv-clean
31 go cct
32 go view
```

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 MAPPING 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 `GO 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 that the residual image reached the `ares` value. Restart deconvolution and look at results.

**Steps 18-25** Tries to improve deconvolution by definition of a support per plane and deconvolve this plane accordingly. The support is store in a file for memory. The deconvolution results are then displayed.

**Steps 26-28** Write residual image, clean image and clean component list in `demo.lmv-res`, `demo.lmv-clean` and `demo.cct` files for memory.

**Steps 29-32** Visualize the cumulative flux as a function of the clean component number and visualize the clean spectra cube in an interactive way.

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 input uv_map
 3 input clean
 4 mx /flux 0 1
 5 show residual
 6 show clean
 7 write beam demo
 8 write dirty demo
 9 write clean demo
10 write residual demo
11 write cct demo
```

Comments:

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

**Steps 2 and 3** Check current state of the variables that control the imaging and deconvolution.

**Steps 4-6** Deconvolve and look at the results.

**Steps 7-11** Write all the internal buffers for memory.

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

### 4.3.4 Comparison and practical advices

#### 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 the possible `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.

**A few (obvious) practical recommendations**

**Map size** Make an image twice the size of the primary beam (*e.g.* $2 \times 55''$ at 90 GHz and $2 \times 22''$ at 230 GHz for PdBI 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 bounded, 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 biased your deconvolution.

**Stopping criterion** Choose the right stopping criterion.

- Estimate an empirical noise on your first deconvolved cleaned image with GO NOISE.
- If this empirical noise value is larger the value computed from the visibility weights (This noise value is one of the output of the UV_MAP command), your observation is probably dynamic limited, *i.e.* you have a bright source whose dirty sidelobes are much larger that the thermal noise. In this case use, 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 converge 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).
- The residual map looks like noise where the source appears in the clean map.

Else change the values of the stopping criterion, in particular the number of clean components (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 there is no spurious visibilities that should be flagged.
- Then check that your deconvolution converged.

- 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 PdBI.

**Outside help** Always consult an expert until you become one.

# Chapter 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. In addition, the first goal of the current generation of (sub)millimeter interferometer was to demonstrate the feasibility of routine interferometric observations in the (sub)millimeter range. The need of doing wide-field imaging at high spatial resolution has thus been recognized only after the success of high spatial resolution single-field imaging. This makes the implementation of wide-field imaging more difficult. The situation will improve with ALMA because wide-field imaging has been included from scratch in its design. It remains that wide-field imaging pushes (sub)millimeter interferometers to their limits.

From a software point-of-view, we always try to hide the technical complexity in more user-friendly tools as soon as our experience increases. However, because of its paradoxical nature, wide-field imaging with an interferometer implies a knowledgeable use of those tools.

## 5.1 General considerations about wide-field imaging

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

$$V(u,v) = \mathrm{FT}\left\{B_{\mathrm{primary}}.I_{\mathrm{source}}\right\}(u,v) + N \tag{5.1}$$

where $\mathrm{FT}\left\{F\right\}(f)$ is the bi-dimensional Fourier transform of the function $F$ taken at the spatial frequency $f$, $I_{\mathrm{source}}$ the sky intensity, $B_{\mathrm{primary}}$ 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 PdBI (see section 5.2), 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

$$V(u=0,v=0) \overset{\text{FT}}{\rightleftharpoons} \sum_{ij \,\in\, \text{image}} \{B_{\text{primary}}.I_{\text{source}}\}_{ij} \,. \tag{5.2}$$

*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 5.3.

## 5.2 Mosaicing

### 5.2.1 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 PdBI, 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.

After calibration and production of an image per field, two different processing flow are possible.

1. The field are deconvolved independently and the clean image are combined to obtain the final image.

2. The dirty images are combined together and a global deconvolution happen on the combined dirty image.

While solution 1 is simpler to implement, solution 2 gives a better deconvolution because 1) information about sources at the edges of and individual field is furnished by the other fields and 2) the signal coming from the different field add up to get a better signal-to-noise ratio. The next two sections describes the particularities of mosaic imaging and deconvolution.

### 5.2.2 Imaging (`UV_MAP` and `RUN MAKE_MOSAIC`)

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

$$M(\alpha,\delta) = \frac{\sum_i \frac{B_i(\alpha,\delta)}{\sigma_i^2} F_i(\alpha,\delta)}{\sum_i \frac{B_i(\alpha,\delta)^2}{\sigma_i^2}}, \tag{5.3}$$

where $M(\alpha,\delta)$ is the brightness of the dirty/cleaned mosaic image in the direction $(\alpha,\delta)$, $B_i$ 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 $\sigma_i$ 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 $(\alpha, \beta)$, we have

$$N(\alpha,\delta) = \frac{\sum_i \frac{B_i(\alpha,\delta)}{\sigma_i^2} N_i(\alpha,\delta)}{\sum_i \frac{B_i(\alpha,\delta)^2}{\sigma_i^2}}, \tag{5.4}$$

and

$$\sigma(\alpha,\delta) = \frac{\sqrt{\sum_i \frac{B_i(\alpha,\delta)}{\sigma_i^2}}}{\sum_i \frac{B_i(\alpha,\delta)^2}{\sigma_i^2}} = \frac{1}{\sqrt{\sum_i \frac{B_i(\alpha,\delta)^2}{\sigma_i^2}}} \tag{5.5}$$

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.

### 5.2.3 Deconvolution (`MOSAIC`, `HOGBOM`, `CLARK` and `SDI`)

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.

### 5.2.4 Implementation and typical use

#### Implementation

The typical processing of PdBI mosaics has four main steps

**Creation of *uv* tables** The convention inside MAPPING is that the visibilities associated to one field of the mosaic are grouped in one *uv* file whose name is made of a generic part plus the field number. For instance, a mosaic of 7 fields would imply the presence in the same working directory of 7 *uv* table, *e.g.* `demo-1.uvt`, `demo-2.uvt`, `demo-3.uvt`, ... `demo-7.uvt`. This is the user responsibility to enforce this at the time of creation of the *uv* table in CLIC.

**Imaging of individual fields** Each field (*uv* table) must then be individually imaged using the standard `UV_MAP` command. This is in this step that the fields are put in the larger framework of the mosaic, *i.e.* they are placed in an image large enough to contain the whole mosaic. This implies that

1. The user must feed the size of the final mosaic through the `map_size` sic variable because the `UV_MAP` command has no information about the mosaic as it works on one field after the other.

2. The projection (*i.e.* phase) center will have to be shifted in this step with the use of the `UV_SHIFT`, `MAP_RA` and `MAP_DEC` variables. Indeed, the observing setup of a mosaic at PdBI implies that each field has its own projection center (the sky tracking direction of the field). The new value of the phase center will be stored in the `A0` and `D0` values of the header while the `RA` and `DEC` values will keep the tracking direction of the observation. Indeed, the latter information is needed to correct for the primary beam attenuation and to form the mosaic. The new phase center can be arbitrarily chosen but it is recommended to choose it around the middle of the mosaic.

At the end of this step, the dirty beams and images must be stored in files (*e.g.* `demo-1.beam`, `demo-2.beam`, ... `demo-7.beam` and `demo-1.lmv`, `demo-2.lmv`, ... `demo-7.lmv`) in the same directory.

**Creation of the mosaic** The `MAKE_MOSAIC` task linearly combines the dirty images using the least square formula 5.3 into a single dirty image named `demo.lmv` and it creates a weight image ($1/\sigma^2$) stored in the `demo.weight` file. In fact, to simplify the deconvolution bookkeeping, the `demo.lmv` contains only the numerator of formula 5.3. As the denominator is equal to the weight image, formula 5.3 can easily be retrieved by dividing the dirty image by the weight image. *** Is this fully true? JP *** The `MAKE_MOSAIC` task also gathers the different dirty beam as different planes of the `demo.beam` file. Finally, it creates a data cube of primary beams (`demo.lobe`). The four files created at this step are needed during the deconvolution. Two input parameters of the `MAKE_MOSAIC` task are important for mosaicing: the primary beam width (FWHM), and the truncation threshold of the primary beam. It is recommended to use a relatively high value for this threshold, typically 0.2, to avoid contaminating adjacent field in case of pointing errors.

**Deconvolution of the mosaic** The `HOGBOM`, `CLARK` and `SDI` variants of the `CLEAN` algorithm has been adapted to treat mosaics. The same commands are used to deconvolve a single field or a mosaic. The change of behavior of the `CLEAN` algorithms is visualize through the change of prompt from `MAPPING>` to `MOSAIC`. The toggle from single field to mosaic is done by the `MOSAIC` command. Note that the use of the `READ PRIMARY` command also automatically switch on the mosaic mode. All standard control variables for a single-field deconvolution control the same aspect of a mosaic deconvolution. Two additional variables are used for mosaic deconvolution

`SEARCH_W` *** ? JP ***

`RESTORE_W` *** ? JP ***

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

An additional subtlety of the current MAPPING implementation of mosaicing is that MAPPING assumes that the individual fields of the mosaic have similar noise level. This is why combining two mosaics observed at different moments is not straightforward.

### Typical mosaicing session

```
1 input uv_map
2 let map_size 256 512
3 let map_cell 0.5
```

```
 4 let uv_shift yes
 5 let map_ra  20:39:10
 6 let map_dec 68:01:20
 7 input uv_map
 8 for ifield 1 to 20
 9    let name "demo-"'iplane'
10    read uv 'name'
11    uv_map
12    write beam  'name'
13    write dirty 'name'
14 next ifield
15 run make_mosaic
16 read beam demo
17 read primary demo
18 read dirty demo
19 input clean
20 hogbom /flux 0 10
21 show residual
22 show clean
23 write residual demo
24 write clean demo
25 write cct demo
26 let name demo
27 go view
```

Comments:

**Steps 1-7** Define the mosaic size, pixel size and projection center.

**Steps 8-14** Image each field individually following the MAPPING name convention.

**Step 15** Combine the different fields into the dirty mosaic.

**Steps 16-22** Read the primary beams, dirty beams, dirty image and weight image into internal buffers. Deconvolve and visualize the results.

**Steps 23-27** Write the results on disk for memory and interactively visualize the deconvolve data cube.

## 5.3 Short spacings (RUN UV_SHORT and RUN UV_ZERO)

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: $D \geq d$.

### 5.3.1 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

$$I^{\rm sd}_{\rm meas} = B_{\rm sd} \star I_{\rm source} + N, \tag{5.6}$$

*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

$$I^{\rm int}_{\rm meas} = B_{\rm dirty} \star \{B_{\rm primary}.I_{\rm source}\} + N, \tag{5.7}$$

*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.

#### 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

$$I^{\rm int}_{\rm clean} = B_{\rm clean} \star I_{\rm source} + N', \tag{5.8}$$

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

$$I^{\rm int}_{\rm clean} = \text{Highpass-filter}\,\{B_{\rm clean} \star I_{\rm source}\} + N'. \tag{5.9}$$

The simplicity of equation 5.8 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 PdBI).

### 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

$$V(u,v) = \left\{ {\rm FT}(B_{\rm primary}) \star {\rm FT}(I_{\rm source}) \right\}(u,v) + N. \tag{5.10}$$

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 ($= 30\,{\rm m} - 15\,{\rm m}$). 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.

**An important subset: The zero spacing** 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 spectra 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's solution as only part of the short spacing information is recovered by this technique.

**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 PdBI 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 + PdBI 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).

### 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 because it is truly difficult to obtain a reliable deconvolution of interferometric data alone when short-spacing information is important. Indeed, a multiplicative interferometer filters 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). When we add the short-spacing information (and in particular the zero spacing) through the pseudo-visibility method, we enforce 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's assume an intensity distribution made of a large scale structure (*e.g.* a uniform intensity) superimposed with a small scale distribution both in emission and absorption. A multiplicative interferometer will filter out the uniform intensity distribution. If there is no additional zero spacing information, the uniform intensity 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. Algorithms of the MEM family enforce positivity in the deconvolved image. In addition, the presence of large negative structures makes instable 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 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's why only the pseudo-visibility method is packaged in a user-friendly way (*e.g.* through the `Short Space Processing` widget). 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.

### 5.3.2 Practical considerations

**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 PdBI antennae, 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 PdBI antennae, 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.

**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 different from the interferometer. In this respect, the IRAM-30m and PdBI are very complementary. Nevertheless, when observing with the single-dish 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...

**How to use your single-dish observations?**

Our algorithm to produce the short-spacing information is coded in the `UV_SHORT` task. This task have about 25 input parameters which are not always intuitive to fill in (in particular the relative weight between single dish and interferometric data). We thus wrote a SIC procedure which wraps up most of the task input parameters according to the context. This procedure is driven by the widget called `short-spacings processing` in the main MAPPING menu. Although you have the possibility to access all the control parameters of the `UV_SHORT` task through this widget, you should have only the file names and the single-dish data unit to fill in, all other parameters being guessed by the procedure.

This task starts from the data format produced by the CLASS `TABLE` command. Basically, this is a GDF table containing one line per spectrum, the columns representing the lambda offset, beta offset, weight, and the spectrum intensities. This format is subject to change: Please, refer to the `TABLE` documentation for up-to-date information. At least two characteristics of this table must be tuned on the characteristics of the interferometric observations:

**The velocity axis** because all the computations are done plane by plane without velocity resampling.

**The center of projection** because all the computations are done using the offset to this center.

To do this, the user needs to work both in MAPPING and CLASS.

**Inside** MAPPING**,** both informations can be found with the following commands

```
1 read uv field-1
2 uv_map
3 write dirty field-1
4 header field-1.lmv
```

Comments:

**Steps 1-3** Images the first field of a mosaic because the information about the velocity axis will be much easier to dig up in the dirty cube `field-1.lmv` than in the *uv* table `field-1.uvt`.

**Step 4** Displays the header as follow:

```
1  File : field-1.lmv                                          REAL*4
2  Size      Reference Pixel        Value                   Increment
3     256   129.0000000000000   0.000000000000000   -1.5514037841057871E-06
4     256   129.0000000000000   0.000000000000000    1.5514037841057871E-06
5      49   25.50000000000000   10.50000000000000    0.2000000029802322
6       1   0.000000000000000   0.000000000000000    0.000000000000000
7 ...
8  Axis 1     A0       05:40:54.270        Axis 2     D0      -02:28:00.00
9 ...
```

**Line 8** Indicates that the axes 1 and 2 are space coordinates around the phase center $\alpha_{2000} = 05:40:54.270$, $\beta_{2000} = -02:28:00.00$.

**Lines 3-6** Describes the axes and in particular the velocity axis as line 5, *i.e.* 49 channels of 0.2 km/s width, the velocity of the 25.5 channel being 10.5 km/s.

**Inside,** CLASS you need to type the following commands to align the characteristics of your single-dish data to the above values

```
 1 file in  single-dish-reduced.30m
 2 file out single-dish-reprojected.30m single
 3 set level 5
 4 set system equatorial 2000.00
 5 find
 6 for ientry 1 to found
 7    get next
 8    modify position 05:40:54.270 -02:28:00.00
 9    write
10 next ientry
11 file in single-dish-reprojected.30m
12 find
13 table short-spacings new /resample 49 25.50 10.50 0.2000000029802322 V
14 xy_map short-spacings
15 header short-spacings.lmv
```

**Steps 1-10** Will ensure that the used data is in Equatorial 2000 coordinate system (The default for PdBI data) with the right projection center. Steps 1 and 2 opens the input and output files. Step 5 decreases the verbosity of CLASS: This is useful when dealing with OTF maps with a large number of spectra. Step 4 enforces the use of the 2000 equatorial coordinate system. Steps 5 to 10 modifies the coordinates of all spectra inside the input file and write them in the output files.

**Steps 11-13** Create the table with an on-the-fly resampling of the velocity axis. In the future, the CLASS `TABLE` will also be able to modify the projection center on-the-fly.

**Steps 14-15** Enable to verify by eye that the consistency of the single-dish data with the interferometric data, *i.e.* the displayed header information for velocity axis and projection center should be identical to the above ones. Those steps are optional as the input really used by the short-spacing processing is produced at step 13.

It is clear that your single-dish data must be reduced before applying the above steps. CLASS has many facilities to visualize and reduce your data.

# Chapter 6

# Internal Helps

## 6.1 CLEAN Language Internal Help

## 6.2 Description of Associated Tasks

### 6.2.1 mapping

```
UV_AVERAGE       Channel averaging of UV data
UV_CASA          Prepare a UV Data Set imported from CASA by FITS for GILDAS
UV_CENTER        Find the centroid of (centro-symmetric) emission from a UV tabl
UV_CHECKBEAM     Check how many beams may be required
UV_CIRCLE        Compute the azimuthal average of an input UV table
UV_CLIP          Clip UV data to suppress outliners
UV_CTIME         Time Average a UV data set
UV_CUTS          Output radial visibility profiles from an input UV table
UV_DFT           Image a UV table through Direct Fourier Transform (no FFT)
UV_FIT           Fit simple functions through a UV table (using SLATEC)
UV_FITC          Fit spherically symmetric, centered models through a UV table
UV_FIT-PROPER    Fit functions and proper motions through a UV table
UV_GETINTERVAL   List the time intervals in a UV table
UV_HANNING       Channel Hanning smoothing of a UV table
UV_INVERT        Make image from visibilities in a transposed UV table, using gr
UV_LIST          List part of a UV table
UV_MERGE         Merge two UV table with the same spectral characteristics
UV_MFLAG         Flag visibilities too far from a source model
UV_MOSAIC        Gather or Split Mosaic UV Tables
UV_MULT          Multiplication by a constant of the U and V columns of an input
UV_NOISE         Recompute visibility weights based on the visibility RMS outsid
UV_PROPER_MOTION Apply proper motion to a UV tqble
UV_REF_GAUSS     Phase and/or amplitude reference a UV table on a nearby Gaussia
UV_REF_MODEL     Phase and/or amplitude reference a UV table on a nearby modeled
UV_REF_POINT     Phase and/or amplitude reference a UV table on a nearby point s
UV_ROBUST        Reweight a UV table
UV_SHORT         Compute short/zero spacing visibilities from single-dish data.
UV_SPLITPOLAR    Split a multi-POLAR UV table into a series of single POLARs
```

```
UV_SUBTRACT      Subtract continuum averaged visibilities from a line UV table
UV_STOKES        Extract polarization states from a polarized UV table
UV_TEMPLATE      Template task to handel UV data
UV_TRIM          Trim a UV table of useless data
UV_UPDATE_FIELDS Support task for @ fitsovt script.
UV_SPLITPOLAR    Split a multi-POLAR UV table into a series of single POLARs
UV_VLA_REORDER   Reorder a VLA UV table by Frequency
UV_ZERO          Add zero spacing to a UV table from a single dish spectrum
MAP_REFINE        Refine precision of dirty images
UV_CAL           Apply an input table of gain (from UV_GAIN) to a table of uncal
UV_GAIN          Compute (from self-calibration of observation on a given model)
UV_CAL           Apply an input table of gain (from UV_GAIN) to a table of uncal
UV_GAIN          Compute (from self-calibration of observation on a given model)
```

### 6.2.2 UV_AVERAGE

Averages velocity channels of a UV data set to produce a pseudo continuum (single-channel) output UV table. To smooth the UV data but still get several output channels, use UV_COMPRESS or UV_HANNING.

### 6.2.3 UV_CASA

UV_CASA

This task performs all necessary steps to convert a UV Table created from a UVFITS file created by CASA into a shape suitable for normal processing by GILDAS. The steps involved are

- Extraction a single polarization state
- Adjustment of weights from the noise statistics over channels
- Identification of flagged data (which are sometimes written by CASA)
- Trimming of flagged data

UV_CASA combines in a single step UV_SPLITPOLAR, UV_NOISE and UV_TRIM.

### 6.2.4 UV_CENTER

UV_CENTER

Find the centroid of (centro-symmetric) emission from a UV table.

### 6.2.5 UV_CHECKBEAM

UV_CHECKBEAM

This task checks how many set of channels have different weight distributions, due to concatenation of non overlapping UV tables for example.

### 6.2.6 UV_CIRCLE

UV_CIRCLE

Compute the azimuthal average of the visibilities around the UV plane center. The output UV table gives for each bin one visibility associated to one UV radius. This radius is the actual average of sqrt(U^2+V^2) of the visibilities in the bin (i.e. it is not the theoretical bin center). Flagged visibilities are ignored. If no input visibility is found inside a bin, then it is skipped in the output table.

### 6.2.7 UV_CLIP

UV_CLIP

Clip UV data to suppress grossly discrepant values. All channels are flagged if any channel between the first and last considered (specified by CHANNELS$) has an amplitude greater than VCLIP$. Clipped channels are set to zero amplitude and zero weight.

### 6.2.8 UV_CTIME

Averages over TIME$ time interval a UV data set to produce a smaller one.

### 6.2.9 UV_CUTS

UV_CUTS

Compute radial visibility profiles averaged on regularly sampled angle intervals around the UV plane center. The results is output in a 4-D GDF file ordered as:

- Axis 1: Radial points;
- Axis 2: Azimuth angles;
- Axis 3: Frequency channels;
- Axis 4: Amplitude, phase and weight.

This task may be used to check circularity on a source which is centered on the phase reference.

### 6.2.10 UV_DFT

UV_DFT makes a map from UV data using (slow) Fourier Transform. This method introduces no aliasing, and no grid correction, but it is slow.

This task processes all table channels at a time.

### 6.2.11 UV_FIT

This task allows you to fit models directly through the uv visibilities. The models are either simple functions or linear combinations of simple functions. The results of the fitting process are the position offsets in R.A. and Dec (in arc second) of the model source from the phase reference center, and its flux (Jansky). Depending on the fitting functions additional fit results are possible, in such cases, sizes and FWHMs are in arc seconds and position angles are in degrees, counting from the North towards the East. Currently supported distributions and additional fit parameters are:

```
POINT       Point source                      : None
E_GAUSS   Elliptic Gaussian source   : FWHM Axes (Major and Minor), Pos Ang
C_GAUSS   Circular Gaussian source   : FWHM Axis
C_DISK    Circular Disk              : Diameter
E_DISK    Elliptical (inclined) Disk : Axis (Major and Minor), Pos Ang
RING      Annulus                    : Diameter (Inner and Outer)
EXPO      Exponential brightness     : FWHM Axis
E_EXPO    Elliptic exponential       : FWHM Axes (Major and Minor), Pos Ang
POWER-2   B = 1/r^2                  : FWHM Axis
POWER-3   B = 1/r^3                  : FWHM Axis
U_RING    Unresolved Annulus         : Radius
E_RING    Inclined Annulus           : Inner, Outer, Pos Ang, Ratio
SPERGEL   Spergel brightness profile : Half light radius, nu
E_SPERGEL Elliptic Spergel profile   : Half light semi-Axes (Maj. and Min.),
```

### 6.2.12 UV_FITC

This task allows fitting of simple models directly from the visibilities. The models are linear combinations of 2 radial distributions, assumed to be centered on the current phase tracking center. Use tasks UV_CENTER and UV_SHIFT if necessary before fitting.

A more flexible and more general fitting procedure is available in the UVT task. This version is more suitable for plotting a fit over rebinned visibilities.

```
The type of the first radial distribution. Available types are
    NONE      No function to fit (for 2nd function usually)
    POINT     Point source
    GAUSS     Gaussian sources
    DISK      Disk
    RING      Annulus
    EXPONENT  Exponential brightness
    POWER-1   B = 1/r
    POWER-2   B = 1/r^2
    POWER-3   B = 1/r^3
    PARABOLA  Parabolic brightness distribution
```

### 6.2.13 UV_FIT-PROPER

This task allows you to fit models directly through the UV visibilities. The models are either simple functions or linear combinations of simple functions. The results of the fitting process are the position offsets in R.A. and Dec (in arc second) of the model source from the phase reference center, and its flux (Jansky). Depending on the fitting functions additional fit results are possible. Currently supported distributions and additional fit parameters are:

```
POINT     Point source              : None
E_GAUSS   Elliptic Gaussian source  : FWHM Axes (Major and Minor), Pos Ang
C_GAUSS   Circular Gaussian source  : FWHM Axis
C_DISK    Circular Disk             : Diameter
E_DISK    Elliptical (inclined) Disk: Axis (Major and Minor), Pos Ang
RING      Annulus                   : Diameter (Inner and Outer)
EXPO      Exponential brightness    : FWHM Axis
POWER-2   B = 1/r^2                 : FWHM Axis
POWER-3   B = 1/r^3                 : FWHM Axis
E_RING    Inclined Annulus          : Inner, Outer, Pos Ang, Ratio
```

### 6.2.14 UV_GETINTERVAL

List the time intervals in a UV table.

### 6.2.15 UV_HANNING

Makes a Hanning averaging of channels of a UV data. The output data set is spectraly smoothed, but keeps the same number of channels. Check also UV_AVERAGE and UV_COMPRESS for related functions.

### 6.2.16 UV_INVERT

```
   Warnings: JP May 30th, 2004.
 - It is not clear that the image noise computed from the visibility
   weights is correct when tapering and/or weighting.
 - It is not clear what is the 4th element of UV_TAPER$.
```

Optimized version of UV_MAP using a .tuv table rather than a .uvt table (i.e. in time vs channels order, rather than in the natural channels vs time order). This version is disk-based, and computes the optimal number of channels to be processed together based on the specified memory usage by SPACEPPING. It is not memory limited, provided one channel fits into virtual memory. This means that this task will be able to process ALMA data...

UV_INVERT makes a map from UV data by gridding the UV data using a convolving function, and then Fast Fourier Transforming the individual channels. UV_INVERT can produce one beam per channel or a common beam for all channels. In the latter case, the gridding is done only once, thus neglecting frequency change between channels; this is of course much faster.

UV_INVERT can shift the map center and rotate the image, by shifting the phase tracking center and rotating the UV coordinates of the input UV table.

UV_INVERT produces an output LMV cube (X, Y, Velocity) and a BEAM image or cube from the input UV table.

### 6.2.17 UV_LIST

This task list one channel of a UV data set, by specified time steps.

### 6.2.18 UV_MERGE

```
UV_MERGE
```

This task merges together two tables of UV data, to form a single output table. The two input tables must have the same spectral characteristics (not checked). This task is to be used to merge two uv Tables, e.g. to combine Lower Side Band and Upper Side Band continuum UV tables.

Multiplicative factors can be applied to both the amplitudes and weights of the first input UV Table.

### 6.2.19 UV_MFLAG

Flag the data if the REAL part of the visibility is greater than and lower than 2 thresholds = (1+/-TOLER)*model. Data are flagged setting weights to 0.

### 6.2.20 UV_MOSAIC

This task combines individual UV tables corresponding to individual primary beams into a single UV table with additional pointing center information, or vice-versa (for compatibility & tests) depending on the value of FIELDS$.

### 6.2.21 UV_MULT

UV_MULT modifies the U,V length of a table in order to change the size of the final image.

MULTA and MULTB are the multiplication factor. The total flux is also modified to be compatible with the new image (i.e. a doubled size object, has 4 times more flux). An Intensity calibration factor can also be used to modify the total flux by MULTC factor.

### 6.2.22 UV_NOISE

UV_NOISE

UV_NOISE computes for each UV distance the RMS of the visibilities outside the line. This rms is compared to the current weight of the visibilities. After this, a median scaling factor is computed on a series of visibilities, and the weight of the visibilities is rescaled using this median factor.

UV_NOISE optionally flag data for which the estimated scaling factor deviates substantially from the median.

### 6.2.23 UV_PROPER_MOTION

UV_PROPER_MOTION

Apply proper motions to a UV data set to bring them to the same common date. This common date is that of the Equinox specified in the Source Position information, usually J2000.0

### 6.2.24 UV_REF_GAUSS

UV_REF_GAUSS does a Phase and/or amplitude referencing of a UV table, using another one representing a Gaussian source observed quasi-simultaneously. In details:

- UV_REF_GAUSS self-calibrates first the observed visibilities of the Gaussian source used as reference, based on the knowledge of the geometry and flux of this source. The self-calibration is baseline based (i.e. it does not take into account closure relationships).
- UV_REF_GAUSS then applies the gains derived in previous self-calibration to the visibilities of the source to be calibrated.

It is STRONGLY ADVISED to self-calibrate the phase first, then the amplitude using a longer smoothing time. Moreover UV_REF_GAUSS enables to subtract (part of) the input reference source, typically to provide continuum-free spectral line maps.

See also UV_REFPOINT or UV_REFDEL, which does the same thing, but using a point or modeled source as input.

### 6.2.25 UV_REF_MODEL

UV_REF_MODEL does a Phase and/or Amplitude self-calibration of a UV table, another observed UV table for which a correct model is known. In details:

- UV_REF_MODEL self-calibrates first the observed visibilities of the modeled source used as reference. The self-calibration is baseline based (i.e. it does not take into account closure relationships).
- UV_REF_MODEL then applies the gains derived in previous self-calibration to the visibilities of the source to be calibrated.

It is STRONGLY ADVISED to self-calibrate the phase first, then the amplitude using a longer smoothing time. Moreover UV_REF_MODEL enables to subtract (part of) the input reference source, typically to provide continuum-free spectral line maps.

See also UV_REF_POINT, UV_REF_GAUSS, which does the same thing, but using a point and a Gaussian source as input.

### 6.2.26 UV_REF_POINT

UV_REF_POINT does a Phase and/or Amplitude Referencing of a UV table, using another one representing a point source observed quasi-simultaneously. In details:

- UV_REF_POINT self-calibrates first the observed visibilities of the Gaussian source used as reference, based on the knowledge of the geometry and flux of this source. The self-calibration is baseline based (i.e. it does not take into account closure relationships).
- UV_REF_POINT then applies the gains derived in previous self-calibration to the visibilities of the source to be calibrated.

It is STRONGLY ADVISED to self-calibrate the phase first, then the amplitude using a longer smoothing time. Moreover UV_REF_POINT enables to subtract (part of) the input reference source, typically to provide continuum-free spectral line maps.

See also UV_REF_GAUSS or UV_REF_MODEL, which does the same thing, but using a Gaussian or modeled source as input.

### 6.2.27 UV_ROBUST

UV_ROBUST

Re-weight a UV table in the same way as UV_MAP.

### 6.2.28 UV_SHORT

UV_SHORT

This 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 added to the output UV table (-merged.uvt) alongside the original visibilities.

This task computes short spacings, i.e. when the Interferometer dish diameter is smaller than the Single-dish diameter, otherwise the task UV_ZERO should be used instead (see HELP UV_ZERO).

The task performs 4 actions:
(1) Setup: Initialization of all relevant variables
(2) Grid: Creation of a map from the spectra over a grid

(3) Pseudo: Computation of UV visibilities from the gridded spectra
(4) Merge: Creation of a new UV table (-merged) with the pseudo visibilities plus the original visibilities

### 6.2.29 UV_SPLITPOLAR

UV_SPLITPOLAR

Extract a single polarization from a UV table containing data for multiple polarization states.

### 6.2.30 UV_SUBTRACT

UV_SUBTRACT subtracts a time average continuum UV table from a spectral line one, typically to provide continuum-free spectral line maps. Caution: it will not work if the continuum source is too complex, because of the time averaging.

### 6.2.31 UV_STOKES

UV_STOKES

Extract a single polarization from a UV table containing data for multiple polarization states.

### 6.2.32 UV_TEMPLATE

Averages NC$ channels of a UV data set to produce a smaller, spectrally smoothed one. Check also UV1

### 6.2.33 UV_TRIM

UV_TRIM

Trim a UV table by suppressing all flagged data or trailing columnds

### 6.2.34 UV_UPDATE_FIELDS

UV_UPDATE_FIELDS

Replace Fields ID by Fields offsets from Ra, Dec center. This task is to be used only through the "@ fitsovt" script, to import UV Tables from UVFITS files.

### 6.2.35 UV_SPLITPOLAR

UV_SPLITPOLAR

Extract a single polarization from a UV table containing data for multiple polarization states.

### 6.2.36 UV_VLA_REORDER

UV_REORDER

Reorder a VLA UV table by Frequency

### 6.2.37 UV_ZERO

UV_ZERO

This task appends zero-spacing (single-dish) data (spectrum and/or continuum flux ) to an existing UV table. If zero-spacing data already exists in the UV table, they are replaced by the new data. The zero spacing spectrum is a GILDAS table, which should be created in CLASS by the command GREG. Additional inputs are: the weight to be used, an amplitude calibration factor affecting the spectrum, and a continuum flux.

The single-dish spectrum is resampled to the UV table spectral characteristics.

### 6.2.38 MAP_REFINE

Task to "refine" the imaging process.

The zeroth order in imaging is to compute a single gridding and beam for all channels, at the center frequency: this is the standard way of making imaging. Strictly speaking, it produces an angular scale inversely proportional to frequency. This is obtained by setting ONEBEAM$ to NO (and BCOL$ = 0) in the imaging tasks.
The infinite order is to compute one beam per channel, obtained by stting ONEBEAM$ to YES, but it is a costly process as gridding is to be redone for each channel.

This task is here for the first order correction of the angular scale varying with frequency, just making a linear interpolation between images computed with two different gridding because of the angular scale. The images (or beams) are assumed to be computed for the frequencies of the 2 extreme channels.

These files (.lmv and .beam) can be prepared by running twice the tasks UV_MAP or UV_INVERT with parameter BCOL$ set to 1 and the number of channels respectively on each execution.

### 6.2.39 UV_CAL

UV_CAL

UV_CAL applies phase and/or amplitude calibration gains to a table of uncalibrated visibilities. The gains are read from a table in general produced by the UV_GAIN task. This task enables phase and/or amplitude referencing from continuum to line or from one simple source (i.e. easily self-calibrated) to a complex one.

### 6.2.40 UV_GAIN

UV_GAIN

UV_GAIN computes a table of phase and/or amplitude gains from the self-calibration of an observed UV table on a model UV table. The table of self-calibrated visibilities is also output. The table of gain can then be applied to another UV table through the UV_CAL task. This is useful for instance to calibrate a line UV table from reference to the gain derived from the self-calibration of the high signal-to-noise ratio continuum UV table of the same source. Only one channel of the observed UV table can be calibrated at a time.

It is STRONGLY ADVISED to self-calibrate the phase first, then the amplitude using a longer smoothing time. You can choose between baseline-based and antenna-based method to determine the gain.

The antenna-based method takes the closure relationships into account. You may thus iterate the self-calibration using this method to try to converge. However, for this method to work, decorrelation on the duration of each observed scan must be small to avoid the loss of the amplitude closure relationships in the averaging. In particular, the atmospheric phase contribution must be stable on the scan duration.

Similarly, the source structure should be small enough so that visibilities take during the averaging time represent the same measurement (within the Nyquist sampling). This can be ensured to first order by centering the image on the strongest source.

### 6.2.41 UV_CAL

UV_CAL

UV_CAL applies phase and/or amplitude calibration gains to a table of uncalibrated visibilities. The gains are read from a table in general produced by the UV_GAIN task. This task enables phase and/or amplitude referencing from continuum to line or from one simple source (i.e. easily self-calibrated) to a complex one.

### 6.2.42 UV_GAIN

UV_GAIN

UV_GAIN computes a table of phase and/or amplitude gains from the self-calibration of an observed UV table on a model UV table. The table of self-calibrated visibilities is also output. The table of gain can then be applied to another UV table through the UV_CAL task. This is useful for instance to calibrate a line UV table from reference to the gain derived from the self-calibration of the high signal-to-noise ratio continuum UV table of the same source. Only one channel of the observed UV table can be calibrated at a time.

It is STRONGLY ADVISED to self-calibrate the phase first, then the amplitude using a longer smoothing time. You can choose between baseline-based and antenna-based method to determine the gain.

The antenna-based method takes the closure relationships into account. You may thus iterate the self-calibration using this method to try to converge. However, for this method to work, decorrelation on the duration of each observed scan must be small to avoid the loss of the amplitude closure relationships in the averaging. In particular, the atmospheric phase contribution must be stable on the scan duration.

Similarly, the source structure should be small enough so that visibilities take during the averaging time represent the same measurement (within the Nyquist sampling). This can be ensured to first order by centering the image on the strongest source.

# Appendix 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.

# Bibliography

Clark, B. G. 1980, A&A, 89, 377

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


CLARK, 23–26, 28, 29, 34, 35
COLUMN, 16

GO, 4, 5
  CLEAN, 4
  FLUX, 26
  NOISE, 8, 27, 29
  PLOTFIT, 14
  UV_MAP, 4
  UVALL, 4, 6, 7, 12, 13
  UVCOV, 12, 13
  UVFIT, 14

HEADER, 10
HOGBOM, 23–29, 34, 35

INPUT, 4, 5
  UV_MAP, 20

MAKE_MOSAIC, 35
MAP_REFINE, 53
mapping, 43
MOSAIC, 34, 35
MRC, 23, 25, 26, 29
MULTI, 23, 25, 26, 29
MX, 23–26, 28, 29

READ
  BEAM, 26
  DIRTY, 26
  PRIMARY, 35
  UV, 19
RUN, 4, 5
  MAKE_MOSAIC, 33
  UV_CLIP, 13
  UV_MAP, 4
  UV_MFLAG, 13
  UV_SHORT, 36
  UV_ZERO, 36

SDI, 23, 25, 26, 29, 34, 35
SUPPORT, 26

TABLE, 41, 42

UV_AVERAGE, 44
UV_CAL, 54, 55
UV_CASA, 44
UV_CENTER, 44
UV_CHECKBEAM, 44
UV_CIRCLE, 45
UV_CLIP, 13, 45
UV_CTIME, 45
UV_CUTS, 45
UV_DFT, 46
UV_FIT, 15, 46
UV_FIT-PROPER, 47
UV_FITC, 46
UV_FLAG, 13
UV_GAIN, 54, 55
UV_GETINTERVAL, 47
UV_HANNING, 47
UV_INVERT, 48
UV_LIST, 48
UV_MAP, 7, 13, 19, 26, 29, 33, 34
UV_MERGE, 48
UV_MFLAG, 13, 49
UV_MOSAIC, 49
UV_MULT, 49
UV_NOISE, 49
UV_PROPER_MOTION, 49
UV_REF_GAUSS, 50
UV_REF_MODEL, 50
UV_REF_POINT, 51
UV_ROBUST, 51
UV_SHORT, 41, 51
UV_SPLITPOLAR, 52, 53
UV_STAT, 19–22
UV_STOKES, 52
UV_SUBTRACT, 52
UV_TEMPLATE, 52
UV_TRIM, 52
UV_UPDATE_FIELDS, 52

UV_VLA_REORDER, 53
UV_ZERO, 40, 53

WRITE
    SUPPORT, 26
    UV, 13