Original version at `http://iram-institute.org/medias/uploads/mapping-noema.pdf`

IRAM Memo 2016-1
MAPPING for NOEMA: Concepts and Usage

S. Guilloteau[1]

1. LAB (Bordeaux)

14-Jul-2016 – version 1.0
09-Sep-2016 – version 1.1 – Add "On Going Work" section

Abstract

With the advent of ALMA and NOEMA, the interferometers deliver much larger data sets than initially anticipated for GILDAS software. This document describes new facilities in MAPPING to handle 1) Mosaics and 2) Wide bandwidth data sets, and a proposed new (simpler) way of using MAPPING.

Related documents: *Mapping documentation*

Contents

1 Goals

The two goals of the modifications are

1. to implement a simpler (and incidentally faster) scheme to process Mosaics
2. to offer a proper implementation of imaging in case of wide relative bandwidth, where the natural angular resolution changes with frequency.

In addition, the code was structured so as to take advantage of parallel programming in all possible cases, and maximize code re-use.

2 Principle

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

¿From the user point of view, Mosaics are now treated as the Single field case. The same commands UV_MAP and CLEAN automatically recognize whether there is one or more fields to be treated. For the user, the imaging sequence is thus always the same

```
READ UV MyData.uvt /RANGE Min Max Type
! here, optionally use UV_TIME, UV_COMPRESS, UV_BASELINE
! or UV_FILTER, UV_CONT to filter lines or remove continuum
UV_MAP  ! Image
CLEAN   ! Deconvolve
WRITE * MyData  ! Save result if OK
```

The only difference between the two cases is that MAPPING yields a Sky brightness image for Mosaics, but does not automatically correct for primary beam attenuation for Single fields.

Imaging and deconvolution parameters are controlled by the MAP_* variables. Suitable defaults are provided, and variations can be suggested by command UV_STAT

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

2.1 Multi-Fields UV table

One major step to allow this simplification has been the possibility offered by the GDF UV data format version 2 to allow "extra" columns, i.e. additional data for each visbility. We use it here to store several possible quantities

- column of type `code_uvt_id` contains a real (actually, only integer values are to be found) representing a numerical identification number. This typically handles the field number, for example, but the column type is generic.
- columns of types `code_uvt_loff` and `code_uvt_moff` contain the phase offset center.

- columns of types `code_uvt_xoff` and `code_uvt_yoff` contain the pointing offset center.

If the phase offset columns are present, but the pointing offset columns are not present, the pointing center is assumed to be the phase center. This naturally happens for data taken by all interferometers (NOEMA, ALMA, JVLA, ...).

2.1.1 Importing from UVFITS

To import UVFITS files, some trick is needed because the UVFITS format has some specific structure and implicit assumptions that do not map directly to the GILDAS UV data format. Specifically, UVFITS provides for each visibility an ID number corresponding to the source coordinates given in the FITS binary table named "SU".

Thus, importing cannot be done through a simple command, and is done only through the `fits_to_uvt.map` script The UV table is temporarily created with two additional columns, one of which is holding the source ID number, and after the SU Table is read, these two columns are filled with the phase offset centers, and labelled of type `code_uvt_loff` and `code_uvt_moff` respectively. As UVFITS provides absolute coordinates, and not offset, the first field center is arbitrarily taken as the reference for offsets. This may change for a more convenient value (e.g. the centroid) later. If only one field is found, the columns are simply labelled as type `code_uvt_id` so that they remain ignored in any further processing.

NOTE: *The UVFITS format allows for cases where the SU tables also provide pointing centers. So far, this has not been implemented in any known FITS writer, and CASA does not support this possibility. However, once GILDAS UV tables have been converted to a common phase center (see below), this possibility may be required to export simply the data into UVFITS format for archival. The current code does not allow this yet.*

2.1.2 Importing from CLIC

CLIC can produce multi-field UV tables. This is done using command

```
TABLE [MyTable [Old|New]] /MOSAIC
```

which adds the phase offset columns and automatically avoids checking pointing and phase offsets.

Note: the /MOSAIC option replaces the experimental /POSITION option. The same result can also be obtained for spectral line data only (SET SELECTION LINE) through the command

```
SG_TABLE [MyTable [Old|New]] /ADD L_PHASE_OFF M_PHASE_OFF /NOCHECK POINT PHASE
```

2.1.3 Importing from separate fields.

It is also possible to merge UV tables corresponding to separate fields into a multi-field UV table by using task UV_MOSAIC. The same task can also do the splitting per field (for test purpose).

2.2 Multi-Field UV_MAP processing

When UV_MAP encounters more than 1 field in a UV Table, it first verifies whether phase (code_uvt_loff & code_uvt_moff) or pointing (code_uvt_xoff & code_uvt_yoff) offsets are present. In case phase offsets are present, it first automatically process the UV table to recenter it onto a common phase center, and converts the offsets to pointing offsets. Here, the centroid of all fields

is used by default as phase center. This action can be obtained independently by the UV_SHIFT command. When pointing offsets are present, this step is no longer required and imaging can proceed immediately. Further changes of field center and map orientation are dictated by variables MAP_SHIFT, MAP_RA, MAP_DEC, and MAP_ANGLE as for single fields.

Dirty and primary beams are frequency sliced in "chunks" as for single fields.

SHOW FIELDS (or VIEW FIELDS) can display the observed pointing centers.

The display of frequency dependent beams is still to be implemented for Mosaics, as SHOW or VIEW only handles 3-D data cubes.

With such data, UV_MAP automatically activates the MOSAIC mode for further image processing.

2.3 Multi-Field CLEANing

Apart from the proper selection of possibly frequency dependent beams, there has been no change here. The new UV_MAP command produces the same results as the old system using imaging of separate fields and mosaicing through the MAKE_MOSAIC task.

3 Control parameters

The UV_MAP command is controlled by a set of SIC variables of names starting by MAP_

MAP_ANGLE	Position angle of map axis when MAP_SHIFT is set
MAP_BEAM_STEP	Number of channels per common dirty beam, if > 0. If 0, only one beam is produced in total. If -1, an automatic guess is performed from the map size and requested precision (MAP_PRECIS).
MAP_CONVOLUTION	Convolution mode (default 5) (old name CONVOLUTION)
MAP_CELL	Pixel size in arcsecond
MAP_DEC	Declination of new map center
MAP_FIELD	Image size in arcsecond
MAP_PRECIS	Position precision at edge of image, in fraction of pixel size. Default is 0.1.
MAP_RA	Right ascension of new map center
MAP_ROBUST	Robust weighting factor, in range 0 - infty. Default 1.0. (Old name UV_CELL[1])
MAP_SHIFT	Logical indicating whether the phase center must be shifted and/or the image rotated (old name UV_SHIFT).
MAP_SIZE	Image size in pixels
MAP_TAPER_EXPO	the taper exponent. Default 2 (Old name TAPER_EXPO).
MAP_UVCELL	UV Cell size for Robust weighting. Default is 0, meaning that the cell size is derived from the antenna diameter. (Old name UV_CELL[2])
MAP_UVTAPER	Array of 3 values giving the UV taper in m (first two values), its position angle (third value). Default (0,0,0). (Old name UV_TAPER[3]).
MAP_VERSION	Version of code to be used. This is a temporary variable to allow comparison between the new and old codes without quitting

	MAPPING.
MAP_WEIGHT	Weighting mode (old name WEIGHT_MODE)

In addition, WCOL indicates the weight channel and MCOL the channel range to be imaged. However, WCOL should in general be set to zero to allow the beam steps to be set.

3.1 Implementation issues

The implementation has been made in such a way that the name changes do not break backwards compatibility.

The scheme is the following. A Fortran derived type handles all UV_MAP associated parameters, and SIC variables are pointing directly towards one instance of this derived type, handling all default values. Command UV_MAP converts the default values to actual values. Command UV_STAT SETUP does the same. Both should use the same routine (**To Be Checked...**).

A second set of instances of the same Fortran derived type is used as target for the old SIC variables, so that the code can check whether the user has been modifying these ones instead of the new ones, and a warning is issued in such cases. Both instances are made identical after each UV_MAP command.

The `define.map` script as been changed to provide an implementation which is independent of the MAPPING version.

3.2 UV_CONT

UV_CONT requires some knowledge of the image size to evaluate how many channels should be averaged together. This is done using the same routine as in UV_STAT SETUP. Optimization of the evaluation of the Min-Max baseline length has also been made.

4 On going work

This section, dated Sep 2016, describes the required developments for a fully integrated use by users.

4.1 Finalizing the default mode

MAP_VERSION is currently not implemented. For single fields, commands UV_MAP and UV_RESTORE uses the old code, while commands SG_MAP and SG_RESTORE use the new code. For mosaics, UV_MAP and SG_MAP are identical.

The RESTORE facility is not yet available for mosaics.

4.2 UV_SHORT

The task UV_SHORT should support the mosaic-like UV tables. This work is underway, with the following design constraints

- ability to process data as previously
- Maximal backward compatibility: only one new parameter in the .init file for the task to distinguish between the various modes.

- Minimal interface: ability to recover mosaic characteristics from a mosaic-like UV table and telescope characteristics from the telescope section.
- ability to produce a mosaic-like UV table of the short spacings, with either phase or pointing offsets.
- ability to produce a combined mosaic-like UV table handling both the original mosaic-like UV table and the short spacings derived from the Class table.
- ability to treat the zero-spacing case

In the end, UV_SHORT would combine the current capabilities of UV_SHORT, UV_ZERO and UV_MERGE.

4.3 Widgets

Once all the above tools are ready, it will be useful to refurbish the widgets for an efficient use of the new capabilities. Nothing has been undertaken at this stage.