ALMA Memo 386
ALMA+ACA Simulation Tool

J. Pety, F. Gueth, S. Guilloteau
IRAM, Institut de Radio Astronomie Millimétrique
300 rue de la Piscine, F-38406 Saint Martin d'Hères

August 13, 2001

Abstract

This memo describes the February 2001 version of the ALMA+ACA simulation tool that has been developed and implemented in the GILDAS package. It was used to compute the simulations presented in the ALMA memo 387.

This simulation tool includes the following capabilities: single-dish observations, ACA observations, ALMA observations, pointing errors, thermal and phase noise, multi-configuration, mosaic simulation, CLEAN-based deconvolution. A deconvolution method allowing us to reduce an heterogeneous dataset (such as observations from ALMA+ACA) has been developed. The simulator is controled via a simple window interface; a pipeline mode is also available.

1 Introduction

With the prospect of Japan joining the ALMA project in a 3-way partnership, it is necessary to explore what are the best scientific options for the possible extensions of the baseline ALMA proposal. These include a compact array of smaller antennas (the ALMA Compact Array – ACA), which would hopefully provide enhanced wide-field imaging capabilities by measuring the short spacings poorly or not sampled by ALMA.

To properly assess the impact of ACA on the imaging capabilities of the combined array, it is necessary to perform realistic simulations, including noise and pointing errors. This document describes the simulation tools that have been developed at IRAM during the last few months for that purpose.

2 Simulation tool

The simulation package was developed and implemented in the GILDAS environment, used for the data reduction softwares of the IRAM instruments. It runs using the MAPPING software, under HPUX, Linux (RedHat), or Windows. Starting from the existing tools, a number of new algorithms were developed and implemented,

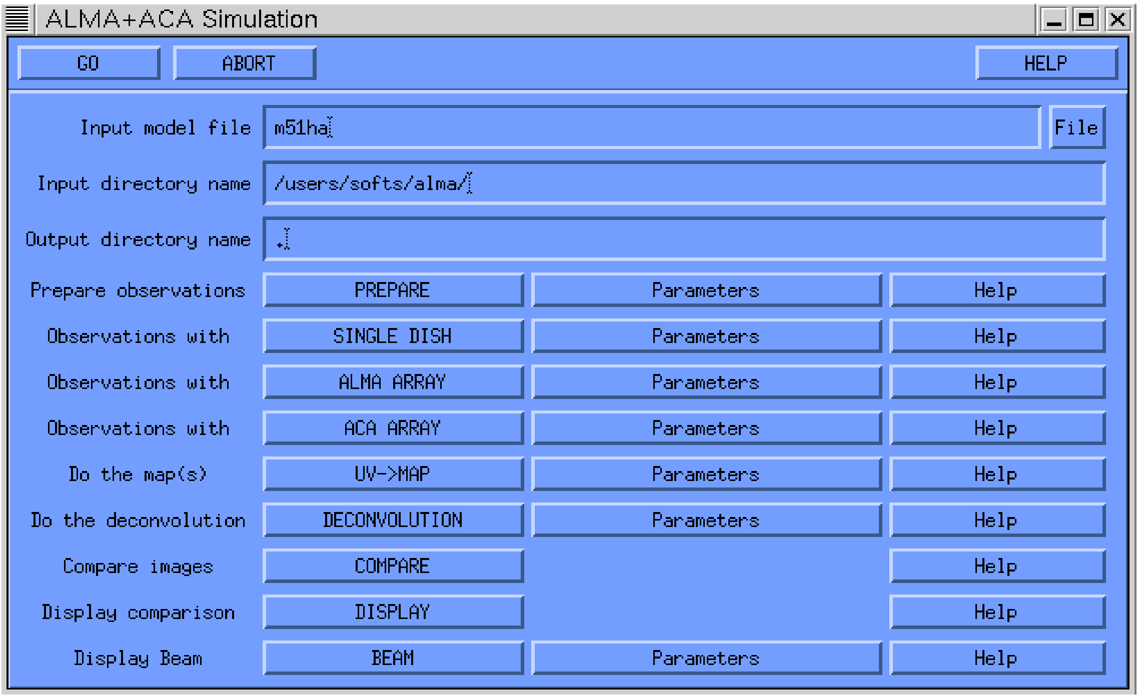

Figure 1: Control window of the simulation package.

as e.g. the simulation of single-dish measurements, pointing errors models, the estimation of visibilities in presence of pointing errors, or the deconvolution of data taken with an heterogeneous array. In addition, several existing tasks were improved, either to be more flexible, and/or to save some memory space, and/or to speed up the computation time. For that purpose, part of the MAPPING software code was translated from FORTRAN77 to FORTRAN90, which allows an easier implementation of mathematical algorithms.

In order to use the various tasks and commands involved in the simulation, flexible procedures as well as a window interface were also written. Fig. 1 shows the control window of the simulator. Several buttons are available, each of them corresponding to a procedure to be executed. Clicking on the "parameters" button opens up a new window (an example is shown in Fig. 2) which allows the user to modify the default input parameters of the procedure. The following list describes the action of each button/procedure (a more detailed description of the critical steps is given in the next section):

`PREPARE` reads the model image, which features the sky brightness distribution, and defines several critical parameters. Depending on the source and primary beam size, the simulator will switch between single field and mosaic modes, and automatically compute the field(s) position(s). Input parameters: declination and size of the source, observing frequency, bandwidth, size of the region to be observed.

`SINGLE DISH` simulates single-dish observations of the source, to be used for short spacings computation. Calibration and thermal noise can be added. Input parameters: antenna diameter, sampling criteria, pointing errors model, weight-

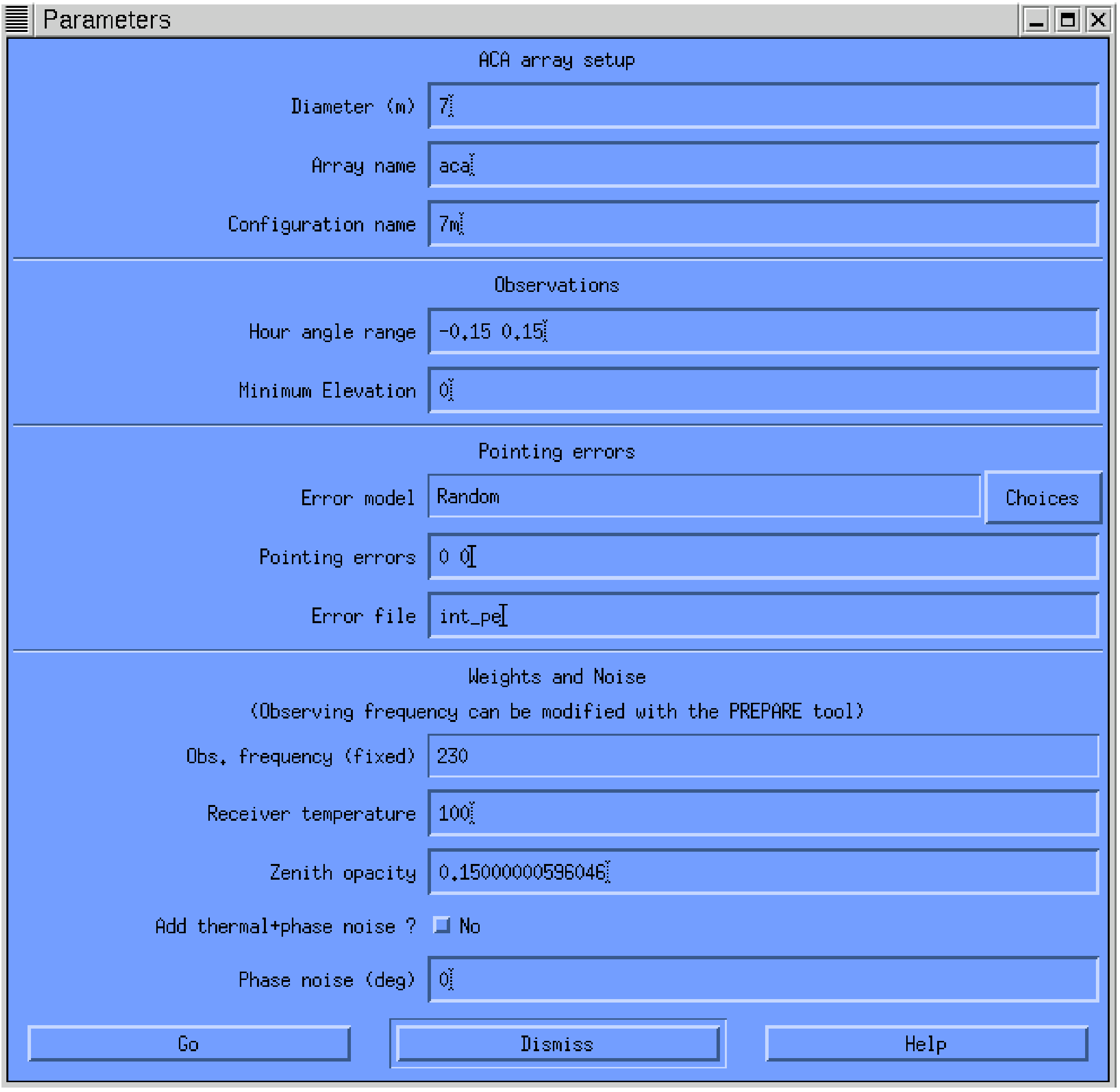

Figure 2: Window obtained by clicking on the "parameter" button in the main window, allowing the user to modify the default parameters (here, of the ACA procedure).

ing parameters (e.g. T_{sys}). Note that the system temperature is automatically computed from the frequency, but can be modified by the user.

`ALMA` simulates observations with the ALMA array, including pointing errors, phase noise, and thermal noise. Input parameters: configuration to be used, hour angle range, pointing error model, weighting parameters. It is possible to run this procedure several times in a row and write the result in the same output *uv* table. This provides a simple way to simulate multi-configuration use of the ALMA array.

`ACA` simulates observations with the ACA array. As compared to the previous procedure, the only additional parameter is the antenna diameter (6, 7, or 8 meters...). Note that all input parameters (hour angle range, pointing errors, etc.) can differ from the ALMA case. Note also that the ACA and ALMA arrays are simulated independently, which implies that the hybrid correlations are not computed and thus not considered by the imaging procedure.

`UV→MAP` computes the maps from the observations simulated in the previous steps. It can process the ALMA array and/or the ACA array. The user can also choose to add the short spacing information to the data: in that case, pseudo-visibilities are computed from the single-dish observations, and added to the interferometric *uv* table(s) before Fourier Transform.

`DECONVOLUTION` makes the deconvolution of the image computed at the previous step, using the CLEAN algorithm (or one of its variant). It can process an heterogeneous array (ALMA+ACA), using a CLEAN-based multi-scale algorithm.

`COMPARE` performs the comparison between the model image and the result of the simulation. The comparison is done by resampling the original model on the grid of the simulated image, and by convolving it with the clean beam. The difference and the fidelity image can thus be easily computed. Image indicators such as the recovered flux or the fidelity range are also derived.

`DISPLAY` plots the original model together with the simulated, difference, and fidelity images. It also presents the cumulated histogram of the fidelity.

`BEAM` plots the dirty beam of the observations, as well as the position of the antenna, and the radial distribution of the weights in the *uv* plane.

The input parameters of the simulation and the intermediate results are stored in files, thus allowing the user to re-run part of the simulation using previously computed data, i.e. without starting from the very beginning. The amount of time needed to perform a simulation depends on the size of the problem. On a 800 MHz Pentium-III PC with 756 Mbytes of memory, the simulation of a 7–fields mosaic observed with ALMA takes about 15 minutes. The simulation of ALMA+ACA is somewhat longer, ∼25 minutes, because the deconvolution of an heterogeneous array is quite slow. As a rule, the imaging part (Fourier Transform and deconvolution) is always the time-consuming step of the simulation.

Finally, a pipeline was written to run automatically all the procedures and save the results. This allows us to run a large number of simulations to assess the impact of the different input parameters (e.g. short spacings addition, noise level, pointing errors, etc.) on the images quality.

3 Simulation methods

This section describes in more details each step of the simulation.

3.1 The `PREPARE` procedure

The original model is first plunged into an image whose size in pixel is a power of two. The header is then edited to modify the pixel size so that the angular size of the image corresponds to that required by the user; the declination of the source can also be modified.

Depending on the image and ALMA primary beam size, the simulator switches between the single-field and mosaic modes. In the latter case, the number and positions of the fields are automatically computed following an hexagonal pattern, sampled at half the primary beam width.

3.2 The `SINGLE DISH` procedure

This task simulates on-the-fly observations performed with single-dish antennas, to be used later on as short spacing information. The model image is convolved by the antenna lobe (via Fourier Transform) and the intensity is then estimated at *(i)* the position of the mosaic fields, and *(ii)* on an extended grid – with typically 3 pixels per beam. If the required position does not coincide with a pixel center, an interpolation from the neighbour pixels is performed.

Pointing errors can be simulated by estimating the intensity at a slightly wrong position. The same kind of pointing error models as for the interferometric observations (see below) are used.

Thermal noise can then be added to the data, and the corresponding weight is stored. Finally, a calibration noise can be simulated: the observed intensities are multiplied by a random factor whose mean value ($\neq 1$ if a systematic error is present) and rms can be specified.

3.3 The `ALMA` and `ACA` procedures

The simulations of the ALMA and ACA observations are actually done using the same procedure. The *uv* coverage is derived from the antenna locations (read from an input ascii file), the source declination, and the hour angle range of the observations. The integration time corresponding to a Nyquist sampling is used if it does not exceed 300 seconds.

For a mosaic, the procedure assumes that the fields are observed in a loop (i.e. in a snapshot mode), each visibility dump corresponding to another position in the sky. The total integration time is truncated so that an integer number of cycles is observed.

Visibilities computation

In the absence of errors, the result of the simulation depends on the accuracy of the simulation process, and on the quality of the imaging methods. Calibration errors (phase and amplitude, but mostly pointing errors) greatly influence the image quality and are often the major limiting factor. These are inherently statistical errors, thus requiring several simulations to provide a representative sample of problems. Hence, computational speed becomes an issue. We have thus chosen a fast, but approximate, method to simulate the visibilities from an input model image and a required *uv* coverage.

For a single field, the observed visibility is the Fourier Transform of the sky brightness distribution multiplied by the primary beam of the antenna, i.e. the convolution of the Fourier Transform of the sky brightness by the Fourier Transform of the antenna primary beam. For rapidly varying pointing errors, this quantity has to be computed for each visibility dump. To fasten this process, we use a "gridded convolution": the visibilities are estimated by the convolution of the FFTs, i.e. of two already-gridded distributions (note that one could in theory correct for this gridding effect later on). Furthermore, the convolution is approximated by a weighted sum of the neighbour points: this is a local operation in the *uv* plane, to be performed at each required *uv* position.

Using images of 4096×4096 pixels, we obtain a precision of the order of 0.3%. This limits the fidelity of the final image to 300:1, but this is acceptable since pointing errors contribute more significantly. Note also that a 1% calibration error would limit the image fidelity to 100:1...

For a mosaic, the same algorithm is used, the field offset being added to the pointing error. The resulting *uv* table is then sorted to get one table per field.

Pointing errors models

The task simulating the visibilities can generate random uncorrelated pointing errors but it can also read an input pointing error table. This allows us to run more sophisticated pointing error models to produce such a table. The model we developed takes into account different timescales and amplitudes for the pointing errors due to the following effects:

- errors of measurement at each pointing calibration (e.g. every 30 minutes)
- tracking uncertainty (timescale: 1 sec)
- bearing error (timescale: a few minutes)
- antenna structure change due to temperature variation (timescale: 1 hour)
- wind (timescale: 1 sec)

The wind and thermal errors are assumed to be correlated among antennas (the correlation factor can be adjusted).

Noise

Adding thermal and phase noise to the simulated *uv* table(s) is an easy task. The thermal noise is derived from $T_{\rm sys}$ (taking into account the variation with elevation), the integration time, and the bandwidth. The corresponding weight is also stored, to be used in the imaging process. As of today, our simulator only includes a constant phase noise, independent of the baseline length. Another limitation is the lack of any calibration error.

3.4 The UV→MAP procedure

The ALMA and/or ACA simulated *uv* tables are Fourier transformed using classical gridding and FFT algorithms. The size and number of pixels are automatically computed. Natural weighting is used by default. For a mosaic, this process is done for each field.

Short spacings

If the short spacing information is to be used, it is added to the *uv* table before imaging. The following method is used (assuming here that the single-dish antennas have a 12-m diameter):

- If only the ALMA array is processed, only the *zero spacing* (the total flux) can be added to the interferometric data. In that case, the flux measured by the 12-m antennas at each pointing position is written into the ALMA *uv* table.
- If the ACA array is processed (together with ALMA or not), then *short spacings* can be estimated:
  - The on-the-fly single-dish measurements are re-gridded and Fourier transformed in the *uv* domain.
  - The data are corrected for the single-dish lobe by division by its Fourier Transform (truncated to the antenna diameter).
  - The data are Fourier transformed back to the image plane and the ACA primary lobe is applied (multiplication).
  - The visibilities are then estimated on a regular grid after yet another Fourier Transform.
  - In the case of a mosaic, the two last operations are performed for each pointing center.
  - Note that applying the ACA primary beam is a convolution in the *uv* plane, i.e. a local average. It can therefore be estimated only in the central region, up to an *uv* radius equal to the difference of the two antenna diameters.
- In both cases, a scaling factor can be applied to the natural weights of the zero- or short-spacings visibilities.

3.5 The DECONVOLUTION procedure

The deconvolution algorithms are based on CLEAN. The default is to use the Clark method, but other variants (Hogbom, Steer-Dewdney-Ito, multi-resolution) are available within the MAPPING software.

Mosaic deconvolution

The mosaic deconvolution is done using a CLEAN-based method developed for the IRAM Plateau de Bure interferometer (see e.g. the chapter on mosaicing in the proceedings of IRAM millimeter interferometry summer school). In short: the dirty mosaic is reconstructed as a linear combination of all dirty maps,

$$J = \frac{\sum_i \frac{B_i}{\sigma_i^2} F_i}{\sum_i \frac{B_i^2}{\sigma_i^2}}$$

where F_i and B_i are the dirty map and the primary beam of each field, respectively. The mosaic is thus homogeneous to the sky brightness distribution, but the noise level is not uniform and strongly increases at the edges of the field of view. As a consequence, using the classical CLEAN method can be dangerous, as noise peaks might be selected as CLEAN components. The method used for Plateau de Bure data consists in selecting the CLEAN components on a modified mosaic: the initial mosaic is normalized by the noise level distribution. Hence, this distribution is homogeneous to the signal-to-noise ratio.

The algorithm turns out to give excellent results, both in terms of image fidelity and convergence speed. It has been used during the last years to deconvolve the mosaics observed with the Plateau de Bure interferometer.

Deconvolving an heterogeneous array

We developed a CLEAN-based method to deconvolve interferometric data observed with an heterogeneous array such as ALMA+ACA. The algorithm takes as input the two dirty maps, and uses a multi-scale approach: at each iteration, the CLEAN components are found on one *or* the other map, depending on which has the highest signal-to-noise ratio. Note that the hybrid correlations are not taken into account in such a method.

Starting from two *uv* tables corresponding to the observations of the same source with two arrays, the following operations are performed:

1. Fourier Transform of the data, and reconstruction of the two mosaics. The fields positions do not need to coincide in the observations performed by the two arrays (although this is the case with the current version of our simulation package).

2. Major cycles are performed, in which CLEAN components are identified in the residual image with the highest signal-to-noise ratio. A very satisfactory method (in terms of image quality and convergence speed) is to use the Clark method if the components are to be found on the ALMA image, and the Steer-Dewdney-Ito approach if they are to be found on the ACA image.

3. The components found in each major cycle are removed from both maps. If they were identified on the ALMA image, a compression is used to re-sample them on the ACA grid. However, the components found on the ACA image need a more tricky method, in order to make sure that the same quantity is removed from both images (otherwise the method would diverge): we first derive from each ACA component a set of positions on the ALMA grid, such that they cover exactly the area of pixel of the ACA grid; these components are then removed from the ALMA map; they are also compressed – via Fourier Transform – back to the ACA grid, to be removed from the ACA image. This back-and-forth method allows us to get rid of the uncertainties that would occur if the ACA components would have been directly interpolated on the ALMA grid.

4. Finally, the CLEAN components identified by the methods are convolved by the ALMA clean beam (highest angular resolution), and the weighted residuals are added.

This CLEAN-based algorithm adapted to heterogeneous arrays could still be improved in several ways, e.g. by constraining the number of components to be found in each image, by using a smoother kernel to derive the ALMA components from the ACA components, etc.

3.6 The `COMPARE` procedure

The comparison between the simulated and original image is done by resampling the original model on the grid of the simulated map, and by convolving it with the clean beam. Hence, the two images are directly comparable, and registration or scaling problems are avoided.

Estimating the quality of the simulated image as compared to the original model is not an easy task: the actual criteria to estimate the observation quality depends on the science that is to be done!

Fidelity

The `COMPARE` procedure computes the so-called fidelity of the simulation. It is defined as the ratio of the input model by the difference (model−simulated image). The higher the fidelity, the better the simulated image. In practice, the very high fidelity points due to value coincidence between the model and simulation have to be flagged out, because they do not reflect the accuracy of the simulation. For that purpose, the lowest values of the difference image are truncated, and the fidelity image is thus computed as:

$$\text{fidelity} = \frac{\text{input model}}{\max(\text{difference}, 0.7 \times \text{rms}(\text{difference}))}$$

The `DISPLAY` task plots the fidelity image as well as a cumulated histogram of the fidelity values, i.e. the number of pixels whose values are larger than a given fidelity. It also computes the median fidelities, taking into account only the pixels whose intensity in the initial model image is higher than 0.3, 1, 3 or 10% of the peak.

Fidelity range

Having one single number to quantify the accuracy of the simulated image is very convenient when doing a large number of simulations: this allows a global estimate of the effect of one of the parameters (e.g. pointing errors) on the simulation results. We used the "fidelity range" defined by L.Kogan:

$$\text{fidelity range} = \frac{\text{max(abs(model))}}{\text{rms(difference)}}$$