SKA MEMO #X

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
HAYSTACK OBSERVATORY
Westford, Massachusetts 01886

December 7, 2009

Telephone: 1-781-981-5407
Fax: 1-781-981-0590

To: SKA Calibration and Processing Group (CPG)

From: Lynn D. Matthews

Subject: Importing Model Skies into MAPS

This Memorandum describes procedures for importing model sky images into the MIT Array Performance Simulation (MAPS) package and for incorporating sky models into simulated observations with an array of radio telescopes.

1 Background

The Square Kilometer Array (SKA) is a proposed next-generation radio telescope that will operate at frequencies between ∼80 MHz to 30 GHz. Once fully operational, the SKA will be 50-100 times more sensitive than existing radio arrays. At 1.4 GHz, this translates to a limiting flux density of ∼10 nJy (1σ). This leap in sensitivity implies that observations, particularly at lower frequencies, will be confusion limited. Moreover, currently favored SKA designs comprising large numbers of small-diameter of dishes (so-called large-N, small-D arrays) will naturally have large fields-of-view. Consequently, not only will the confusion problem be enhanced, but instrumental and atmospheric effects will vary significantly across the field. If not handled properly, these effects will dramatically limit the achievable dynamic range. New hardware developments (e.g., correlator FOV shaping; Lonsdale et al. 2004) and new software (e.g., Cornwell 2007) will be required to overcome these limitations. Testing and optimizing these tools, as well as optimizing the overall design of the SKA, demands the ability to realistically model the sky background seen during routine SKA observations.

In a companion document (Matthews 2009), I described how model sky backgrounds suitable for SKA simulations can be generated using the SKADS Simulated Skies (S^3) package developed at the University of Oxford (Wilman et al. 2008). Here I will describe how these sky models can be incorporated into simulated radio frequency observations with a user-defined "virtual observatory" using MAPS.

2 A Brief History and Overview of MAPS

Development of the MIT Array Performance Simulator (MAPS) began at MIT Haystack Observatory in 2001 with the goal of providing a flexible tool for the generation of simulated low frequency radio array observations and for testing new radio calibration and processing algorithms (Lonsdale 2001). The code is written in C.

Since its inception, contributions to MAPS have been made by various groups. In early 2004, various modules of MAPS were shipped to Swinburne University in Australia where a now-defunct "members only" interface to the program still resides. More recently, R. Wayth (now at Curtin University) and colleagues have incorporated several new features into MAPS to make it suitable for simulated observations with the Murchison Wide-Field Array (MWA)[1]. These developments, as well as a more extensive technical overview of the workings of MAPS and its full suite of capabilities will be documented in Wayth et al., in preparation. In the mean time, a bugzilla database for MAPS is currently being maintained at http://mwa-lfd.haystack.mit.edu/bugzilla, and a brief summary of its capabilities of MAPS can be found at http://www.haystack.mit.edu/ast/arrays/maps/.

MAPS comprises a collection of several different program modules. Each is operated via a command line-driven interface, with user inputs specified via command line switches and ascii templates (hereafter "meta-files"). A schematic illustrating how MAPS may be used for an arbitrary simulation is shown in Figure 1.

MAPS was designed to accommodate detailed descriptions of heterogeneous interferometric arrays and a variety of other highly flexible user inputs. Some key features include:

- ability to input an arbitrary sky brightness distribution
- option to include out-of-beam sources
- user-specified array geometries (including placement and orientation of individual receptors)
- station-based beam forming
- variable station beams
- observing specifications through an input template (RA and DEC; field-of-view; time and frequency resolution; bandwidth; channel width; correlator integration times; observation start and stop times)
- option to include thermal noise
- time- and location-dependent ionospheric effects; modeling of large- and small-scale ionospheric structure
- fully polarized instrument response
- ability to do all-sky simulations

[1] http://www.mwatelescope.org/

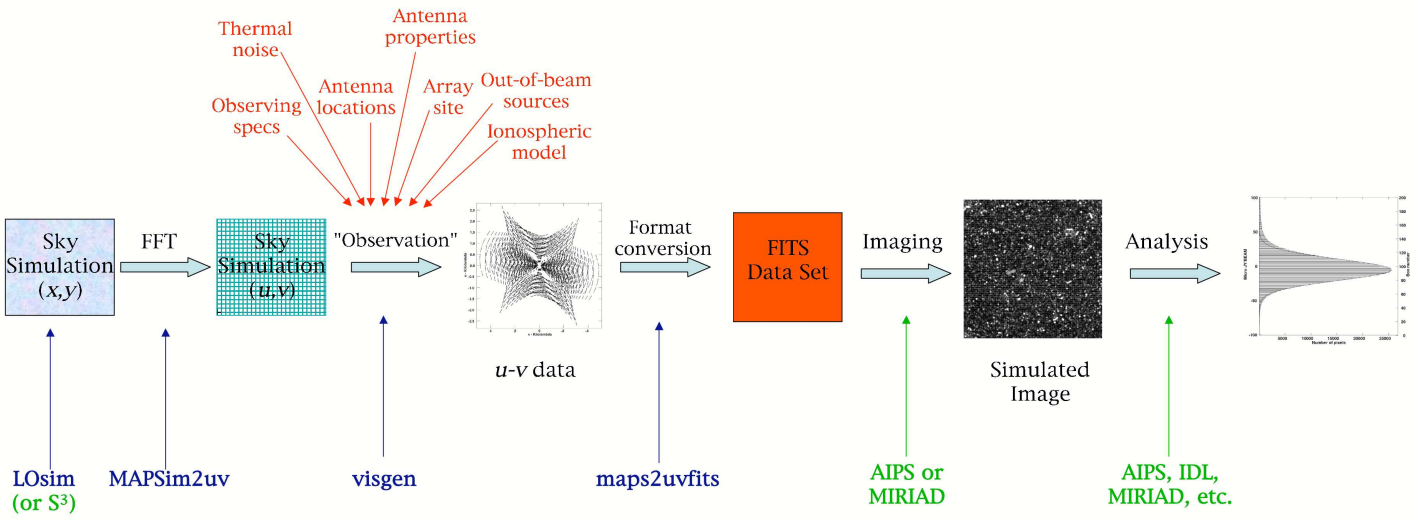

Figure 1: A schematic depicting the process of performing a realistic, simulated radio observation using the MAPS package. Text in blue refers to MAPS modules used to perform various steps. Green labels/arrows refer to external packages that may be used for generating the initial input sky model and for performing imaging or other analysis of the simulated data. Red text and arrows depict user-specified inputs that may be included in the simulation.

- ability to export simulated data into FITS format

MAPS simulations may be constrained to cover a small patch of sky, but all-sky simulations are also possible. Results can be exported in FITS format to allow further analysis through any number of external analysis packages.

This Memo is not intended as a comprehensive Users Guide to MAPS, nor does it describe the inner workings of the code (for some information on the latter, see Doeleman 2001a,b; Cappallo 2002; Wayth et al., in prep.); rather this Memo is intended as a primer for novice users who wish to get started using MAPS and to become familiar with some its basic capabilities. This is a working document, so I also draw attention to current bugs, quirks, and limitations, and point to areas where future development work might be particularly valuable.

3 Installing MAPS

MAPS remains under active development, and for this reason it is not yet a publicly available code. The latest version of MAPS is maintained by R. Wayth (r.wayth@curtin.edu.au), who may be contacted for additional information on recent developments. The repository for the code resides on the server mwa-lfd at Haystack. Persons wishing to download MAPS will need to have an account on this server and should contact R. Crowley (rcrowley@haystack.mit.edu) for further information.

MAPS uses a "subversion" (svn) server for version control. For your initial installation, you will need to "checkout" the code; thereafter, updates can be made using the command

"svn update" from your MAPS base directory.

To perform the initial checkout, enter the following at the command line from the directory where you would like to install MAPS:

% svn checkout svn+ssh://mwa-lfd.haystack.mit.edu/svn/MAPS

This will create a subdirectory called MAPS in the current directory. The entire code requires roughly a few hundred megabytes of space. Several additional third-party packages are also required before MAPS will fully function. A partial list of these can be found in the README file that will be automatically deposited in the MAPS directory. If other necessary packages are missing, error messages that indicate these deficiencies will result when you attempt to run various MAPS modules. The complete instructions for installing MAPS are also contained in the README file.

Once MAPS is installed, the MAPS directory will contain a number of subdirectories:

array contains a series of ascii files with suffix ".txt" that are used to describe antenna locations and properties

doc contains a few pieces of (incomplete) documentation (§ 4.2)

stn_layout contains a set of files (with suffix ".layout") that give the properties of the antennas or antenna tiles to be used for a given simulation

test contains several subdirectories (e.g., test01) that guide the user through sample MAPS computations. In each case, these subdirectories include a "notes" file (e.g., notes_test01.txt) that gives a description of the computation and step-by-step instructions for its implementation, together with any meta-files needed to execute the various steps. At the time of this writing, not all of these test subdirectories were complete, and some of the notes files contain minor typos. Nonetheless, working through a few of these examples is highly instructive for the novice MAPS user.

LOsim, maps2uvfits, maps_im2uv, maps_ionosphere_generation, maps_makesky contain source code and "Make" files for the respective MAPS programs

4 Running MAPS

4.1 Starting MAPS

The first step to any MAPS session is to "source" one of the required set-up files (either sim_setup.csh or sim_setup.sh, depending on your shell) in the MAPS directory. Before running MAPS for the first time, you will need to edit whichever of these files you will use in order to define some necessary paths. Paths that will (or may) require editing in these set-up files are clearly labeled with comment lines. Once the paths are set, type:

% source sim_setup.csh (or source sim_setup.sh)

and you should now be able to run MAPS programs from within this window. Of course you can also set up your login file to do this automatically in the future.

A few other things that are useful to know before proceeding:

- All MAPS programs/commands are executed from the command line
- Typing the name of most MAPS modules without any arguments (e.g., % maps2uvfits), will print to the terminal the calling sequence required to execute that program
- Input to certain MAPS modules is supplied via ascii templates or "meta-files" that must reside in your current directory (editing and use of these meta-files is discussed further below)
- MAPS does not have any native image display capabilities. However, FITS images created using the MAPS program LOsim can be viewed with any standard FITS viewer (e.g., ds9). Visibility data created with MAPS will need to be converted to 'UVFITS' format (using maps2uvfits; see below) and then imported into other packages (e.g., AIPS, MIRIAD, CASA, IDL, etc.) for visualization or further processing.

4.2 Where to go for Help

Unfortunately, a comprehensive Users Guide to MAPS does not yet exist. A few documents with some useful information are contained in the doc subdirectory:

- **LOsim.ps** is an outdated description of the MAPS LOsim program. Ignore all information in the "Installation" section of this manual, as it is obsolete; LOsim is now automatically installed with the rest of MAPS. The remainder of this document, however, contains brief descriptions of the meta-files needed to run LOsim.
- **manual.html** The most "complete" MAPS manual available, this web page contains a brief description of several MAPS modules, the commands for executing them, a list of input and configuration files needed, and a list of available command line switches
- **obs_spec.html** describes each of the fields in the meta-files used to run the MAPS program visgen (see § 5.2.2)

Users who wish to better understand the inner workings of MAPS may also wish to consult the series of memos on the Haystack MAPS home page. These discuss the simulation of station beams within MAPS (Doeleman 2001a); the intrinsic accuracy of LOsim (Doeleman 2001b); and the integration in the u-v plane of the Fourier-transformed sky brightness distribution (Cappallo 2002).

5 Some Step-by-Step Examples

The best way to familiarize yourself with MAPS is by using it. As noted above, the test subdirectory contains instructions for various sample computations. Here I supplement these with some additional examples specifically aimed at importing model sky images into MAPS and incorporating these into mock observations.

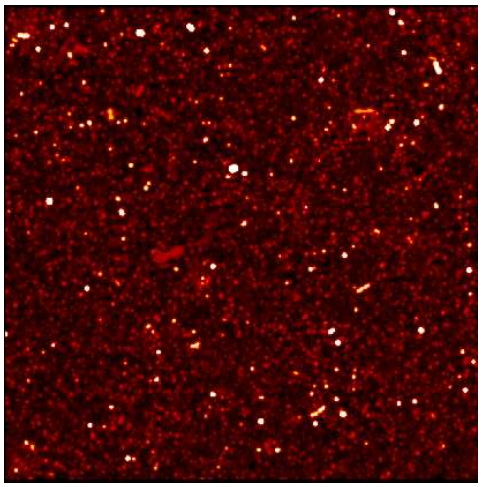

Figure 2: A simulated $1^\circ \times 1^\circ$ image of a patch of radio sky at 1.4 GHz, produced using the S^3 software package (see Matthews 2009). The intensity scale is logarithmic. The simulation is noise-free, so all features correspond to "real" sources. For display purposes, the image has been convolved with a $18''$ circular beam.

5.1 Example 1: Importing a Model Sky Image into MAPS

Suppose you have in hand a FITS image of a $1^\circ \times 1^\circ$ patch of sky at a center frequency of 1.4 GHz called "1.4GHzmodelsky.fits" (Figure 2), generated via the use of the S^3 software package (see Matthews 2009), and this model image has $6''$ pixels.

The MAPS module MAPS_im2uv is able to read a FITS format image, fast fourier transform (FFT) it, and write the results in a binary format that the MAPS program visgen will be able to read (see visgen_binary_format.html in the MAPS/doc directory for information about this format). The program visgen is the "heart" of MAPS, and in the next step we shall use it to generate model visibilities. Ideally, we should be able to perform this FFT and reformating with the following command:

% MAPS_im2uv -i 1.4GHzmodelsky.fits -o SKAtst_Visibility.dat -n 3.59e5

Here, the "- i" switch indicates the name of our input maps; "-o" precedes the desired output name (which must contain the suffix _Visibility.dat). Unfortunately, MAPS_im2uv does not use any of the coordinate information in the FITS header. This means that hereafter, MAPS no longer "knows" the intrinsic spatial scale of your image, the coordinate epoch, or the brightness scale. We shall see that the loss of this information can cause problems later on if we are not careful. We can, however, adjust the units of the image using the "-n" switch; this switch feeds MAPS_im2uv a multiplicative constant to apply to the image such that its output units will be Jy steradian^{-1}. These are the only three switches currently accepted by MAPS_im2uv; all other options have now been depreciated.

Any image outputted from S^3 (regardless of whether it has been convolved with a "beam") will have units of Jy pixel^{-1}. In our example, the pixels are $6''$ per size, implying a multiplicative constant of 3.59e5 will be required to maintain the correct units.[2]

[2]If instead your sky image was created with the MAPS program LOsim (see § 5.5), the units are such

If we now attempt the above command, we will immediately receive an error message:

% input image is NOT 2 dimensional. wrong wrong wrong.

Our image has only two spatial dimensions, but the program disputes this because it interprets the Frequency and Stokes axes in the header as additional "dimensions". One way to solve this problem is to import the original image into AIPS (using task FITLD), transpose the image so that Frequency (rather than RA) is the first axis (task TRANS), and then use task XSUM (with OPCODE='AVE') to "collapse" this axis, making it disappear from the header. Repeat the same procedure for the Stokes axis, and you will be left with an image that has only RA and DEC axes in its header.

While we have the data inside AIPS, we can perform a second crucial operation: padding the periphery of the image with zeros. This will mitigate aliasing that can later introduce bogus sources (and/or their sidelobes) into our model images. Padding can be accomplished using the AIPS task PADIM, and it is recommended to add a border equal to half the image width (resulting in an image twice as large as the original). Following this step, we can write the result back into FITS format (using FITTP), to create the file 1.4GHzmodelsky2Dpad.fits, and we are ready to continue our MAPS simulation.

% MAPS_im2uv -i 1.4GHzmodelsky2Dpad.fits -o SKAtst_Visibility.dat -n 3.59e5

This time we should have successfully created an output visibility file in "visgen" binary format. We are now ready to use our model sky as part of a simulated observation (§ 5.2).

5.2 Example 2: Using an Imported Model Sky Image as Part of a Simulated Observation

Once our model sky image has been stripped of its Stokes and Frequency axes, padded with zeros, and FFTed as described in the previous section, we may now use it as part of a simulated MAPS observation.

5.2.1 Preparatory Steps

Several preparatory steps are required before performing a simulated MAPS observation:

1. Enter the coordinates of your observatory (in East Longitude and Latitude) into the sites.txt file in the $SIM/text subdirectory. Several observatories are already listed in this file, but you may make additions as needed. If you wish to use any of the default entries, it is recommended that you double-check to be sure that they correspond to the latest and best-determined values. Each entry in this file consists of three columns; the first field is a character string of up to 12 letters that uniquely specifies a designation for the array. The next two columns are East Longitude and Latitude, respectively of the array center. For locations less than 180° West of Greenwich, a minus sign should precede the Longitude entry. Comment lines in the sites.txt file should contain an asterisk in the first column.

that the required scaling factor will be equal to the size of the image in radians squared.

2. Define the locations of the stations making up your radio telescope array by creating an "array" file with suffix ".txt" in the $SIM/array subdirectory. This directory will already contain several sample files of this type. The "old" format for these files (which is still accepted; see, e.g., mwa_32_crossdipole_gp.txt) is to specify in the first two columns offsets (in meters East and North, respectively) relative to the array center. The third column may optionally contain a z coordinate. The fourth column (or third column if the z-term is null) should contain a string indicating the type of antenna that is present at each position. For a uniform array, this latter entry will be the same for all antennas. The file that defines the corresponding antenna properties resides in the stn_layout directory (see Step 3).

   The "new" format for the array files contains 8 columns (e.g., Appendix A). The first column contains a character string specifying a unique name for each station (e.g., "ant1", "ant2", etc.). Columns 2, 3, and 4 contain the absolute X, Y, and Z coordinates of each station (in meters) relative to the center of the Earth[3]. Column 5 should contain a string indicating the type of antenna (whose antenna properties will be defined by a file in the stn_layout directory; see Step 3). Columns 6 and 7 contain the lower and upper elevation limits of the antenna in degrees, while Column 8 specifies the system equivalent flux density (SEFD) for the antenna. See merlin4.txt for an additional example of an array file using the "new" format.

3. The $SIM/stn_layout subdirectory contains a series of "station" files, all with suffix ".layout". These files are used to define the properties of the antennas (or multi-element stations) that make up your array. The base name for the station file you wish to implement must match the string used in Column 8 of your "array" file (or Column 4 if you are using the old format; see Step 2). Suppose we wish to build an of 2-m parabolic dishes, and in our "array" file we have designated these with the moniker dish_2m_unpol. We now must create a file called dish_2m_unpol.layout that consists of two lines. The first line defines the name of the antenna elements. In our case, this line would read, verbatim,:

   NAME dish_2m_unpol

   The second line consists of several columns:

   0.0 0.0 0.0 3 1.0 0.0 2.0

   The first three columns define the X, Y, and Z offset of the antenna from its nominal position (as defined in $SIM/text/sites.txt). Generally these values will be zero unless there are multiple antennas comprising a single station. Column 4 defines the antenna type (3=ideal parabola with unpolarized receptor). Column 5 is the phase (assumed here to be 0), and finally Column 6 is the antenna diameter in meters. Note that the number of columns and their meaning changes depending on antenna type.

[3]For an overview of this and other commonly used coordinate systems, see: http://www.colorado.edu/geography/gcraft/notes/coordsys/coordsys.html

5.2.2 Preparing to Run visgen

After Steps 1-3 are completed, we are now ready to use visgen to perform a simulated observation. Running visgen without any arguments will list a complete summary of command line switches. These are also described in $SIM/doc/manual.html. In addition to the command line switches, visgen must be fed a series of inputs via an ascii-format meta-file known as an "obs spec" file, whose format will look similar to the following:

```
FOV_center_RA = 00:00:00
FOV_center_Dec = -26:00:00
// FOV in arcsec
FOV_size_RA = 7200.0
FOV_size_Dec = 7200.0
Corr_int_time = 1.0
Corr_chan_bw = 0.001
Time_cells = 0
Freq_cells = 0

// Scan_start = 2006:274:4:00:37
Scan_start = GHA -7.8237892
// scan duration in seconds
Scan_duration = 3600.0
// freq, bandwidth in MHz
Channel = 1400:0.001
Endscan
```

The obs spec file must reside in the directory where you plan to run visgen. Comment lines are indicated with double backslashes.

As noted above, MAPS_im2uv does not pass along with it any information from the header of the original sky image when it creates the Fourier transform of this image. Consequently, visgen does not have any *a priori* information about the coordinates of the field center or the size of the field-of-view (FOV), and these values must be explicitly passed to the program through the obs spec file. While the RA and DEC of your field center may seem arbitrary for a simulated patch of sky, you must choose coordinates such that your region will be observable from your observatory site at the start time that you specify. Otherwise visgen will (without warning) simply produce an output of all zeros. Also note that the FOV that you specify in your obs spec file *must* match that of your original input sky image, inclusive of any border of zero padding that has been added to its periphery. If you specify a smaller region, visgen will not select a sub-region of your input sky image, it will assume that whatever you specify is the intrinsic FOV of your sky model and scale it accordingly!

The field "Corr_int_time" of the obs spec file allows you to specify your integration time in seconds; sometimes referred to as "dump time" or "record length"; this is not the total duration of your observation, but rather the sampling and recording interval of the data. "Corr_chan_bw" specifies the channel width in MHz. "Time_cells" and "Freq_cells" allow you to subdivide the time and frequency cells into smaller increments to improve computational accuracy; setting these values to zero will turn off this option.

Two options are available for specifying the start time ("Scan_start"). The first option is to explicitly specify the time of your observation in Universal Time (UT) (This option is commented out in the sample file above). The required format is:

Scan_start = YYYY:ddd:hh:mm:ss

where YYYY is the year, ddd is the day number (e.g., December 31 is day 365); hh is the UT hour, mm is the UT minute, and ss is the UT second. Alternatively, you may specify a Greenwich Hour Angle (GHA; sometimes called GST) as follows:

Scan_start = GHA sH.DDDDDDD

The GHA refers to the hour angle your source would have if measured at a given moment from Greenwich (irrespective of whether or not the source is actually visible from your observing location at this time). Here s refers to the sign (+ or −), H indicates the hour, and .DDDDDDD indicates decimal hours. Appendix B contains a recipe for converting between LST and GHA. The "Scan_duration" field specifies the total duration of your observation. Finally, the "Channel" field is specified as follows:

Channel = FFFF:B.BBB

where FFFF is the center frequency of your observation in MHz and B.BBB is the total bandwidth in MHz. If the latter equals the value of "Corr_chan_bw", your resulting u-v data set will comprise a single channel.

5.2.3 Running visgen

Once your obs spec file is set up, you are ready to run visgen. In this example, we will use the obs spec file shown above to perform a noise-free simulation. We will perform the observation from a site in the Australian outback, whose latitude and longitude are -26.62°, E117.51°. Our array configuration will comprise 49 2-m parabolic antennas, whose locations are specified according to the sample "array" file given in Appendix A. The maximum baseline is ∼2.5 km.

The calling sequence for visgen would be as follows:

%visgen -n SKAtst -s SKA_SITE -A $SIM/array/SKA_core.txt -G SKAtst_Visibility.dat -V obs_spec_SKAtst_gha -N -m 0 > visgen_SKAtst.out

Here, the various command line switches have the following meanings:

-n: name of the resulting model visibility file (a ".vis" extension will be added to the specified name) (required)
-s: site name as defined in $SIM/text/sites.txt (required)
-A: the array configuration file (in $SIM/array) defining the coordinates and types of the antennas or stations (required)
-V: name of the obs spec file (required)

-G: name of the gridded *uv* data produced by maps_im2uv (or LOsim; see § 5.5) (optional)
-N: instructs visgen to do a noiseless simulation; otherwise Gaussian noise is added
-m value: controls how verbose are the output messages; possible values are integers ranging from -2 (lots) to +2 little (optional)

In additional to the switches used in our example, other possible visgen command line switches include the following:

-O name: include a list of user-specified point sources in the simulation (see § 5.4)
-I name: turn on ionospheric modeling by naming an ionospheric model file
-i name of ionospheric settings configuration file (used only with "-I")
-S sefd: specify a global system equivalent flux density, sefd, for all stations when adding noise
-Z do not compute visibilities from an input *uv* image (i.e., do not include a model sky background image from LOsim or maps_im2uv)

5.2.4 Converting visgen Output into FITS Format using maps2uvfits

The output of visgen will be a set of vector-averaged visibilities written to a binary data file with extension ".vis". The formatting of this file is further described in the following document: $SIM/doc/visgen_binary_format.html. This output file cannot be read by standard interferometry reduction packages such as AIPS or MIRIAD; therefore, to permit importation of your model visibilities into these packages for further analysis, you will therefore need to convert your visgen output into FITS format using the MAPS module maps2uvfits. The calling sequence for our example is:

% maps2uvfits SKAtst.vis SKAtst.uvfits -26.62 117.51 20.0 $SIM/array/SKA_core.txt

The file SKAtst.vis is our output from visgen while SKAtst.uvfits is the name of the FITS file to be outputted. The three numerical values that follow specify the latitude (in degrees), longitude (in degrees), and elevation (in meters) of our array. Finally, maps2uvfits needs to be directed to to the array configuration file. The latitude, longitude, and array configuration file are used in conjunction to produce an antenna (AN) file to attach to the output FITS file. This is most critical if you have used the "old" array configuration style format that specifies latitude and longitude rather than X,Y,Z coordinates. However, even in the latter case, this information is still used to define the array center. This will be important if you wish to plot your visibilities (e.g., elevation versus time), although for imaging, just the *u-v* coordinates and visibility values are used.

5.3 Imaging Your Model Visibilities

Model visibilities produced by visgen and converted to FITS format using maps2uvfits can be imported into any standard interferometry reduction package for further analysis. Because these steps are specific to the package being used (AIPS, MIRIAD, etc.), details about this process will not be described in this document. I note however two important cautions that the user should be aware of at this stage. First, after importing the data into an external

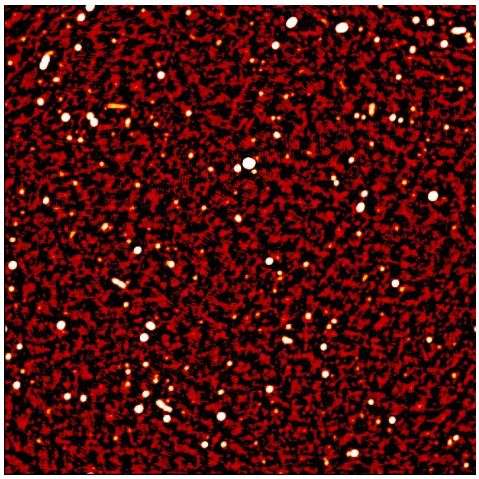

Figure 3: The sky image from Figure 1, as "observed" for one hour with a compact array of 49 2-m antennas and deconvolved using a standard CLEAN deconvolution algorithm (11,000 iterations with no "clean boxes"). The "noise" in this image arises primarily from limitations of an unrestricted CLEAN on such a crowded field. Note that owing to a MAPS bug, the image is flipped east-west relative to the original image in Figure 1.

package, the user will likely need to sort the model visibilities in order to arrange them in time-baseline (TB) order. Secondly, upon imaging the model visibilities (Figure 3), you will find that the resulting image is flipped east-west relative to your input image. Furthermore, because the sign of CDELT1 is actually incorrect in the image header, this cannot be rectified by a simple $x - y$ transpose. If the absolute coordinates of the sources in the image are important, you will to alter the image header to fix the erroneous value of CDELT1.

5.4 Example 3: Adding Additional Sources to an Existing Simulation using visgen

For many types of simulations, the user may wish to add additional sources with specific positions and intensities to existing sky models. For example, one may wish to test how a single bright point source located just outside the primary beam will affect the ultimate dynamic range of the image, or one may wish to test how well one can detect faint background galaxies in an image where several bright radio sources lie near the field center. In such cases, additional point sources can be included in the simulation through the use of a so-called "out-of-beam" file when running visgen. Note the sources contained in this file (which must be named ooblist.txt) need not lie outside the primary beam, but can lie anywhere within the field-of-view of interest.

A sample ooblist.txt file is reproduced below:

```
# format: RA (decimal hours), DEC (decimal degs), Stokes I,Q,U,V
0.177 -0.177 20.0 0.0 0.0 0.0
```

In this example, we will introduce into our simulation an unpolarized 20 Jy point source at a projected distance of $\sim 15'$ from the field center. Additional sources can be added

by including additional lines. At present, there is no way to include *extended* sources through the use of an out-of-beam file, although these can be incorporated into model images computed using the LOsim module (§ 5.5).

5.5 Creating a Sky Model using LOsim

TBD

6 References

- Cappallo, R. J. 2002, LOFAR Memo #006, http://www.haystack.mit.edu/ast/arrays/maps/006.pdf
- Cornwell, T. J. 2007, Astronomical Data Analysis Software and Systems XVI, ASP Conf. Series 376, ed. R. A. Shaw et al., 223
- Doeleman, S. S. 2001a LOFAR Memo #004 http://www.haystack.mit.edu/ast/arrays/maps/004.pdf
- Doeleman, S. S. 2001b LOFAR Memo #005, http://www.haystack.mit.edu/ast/arrays/maps/005.pdf
- Lonsdale, C. 2001, LOFAR Memo #002, http://www.haystack.mit.edu/ast/arrays/maps/002.pdf
- Lonsdale, C. J., Doeleman, S. S., & Oberoi, D. 2004, Exp. Astron., 17, 345
- Matthews, L. D. 2009, CPG memorandum
- Wilman, R. J. et al. 2008, MNRAS, 388, 1335

A Sample "array configuration" File Format

The first five lines of the antenna array configuration file ($SIM/array/SKA_core.txt) used for Example 2 (§ 5.2) are reproduced below. In this example, the "new" antenna format is used, whereby the antenna positions are specified in X, Y, Z coordinates, in meters, relative to the center of the Earth:

```
#ANTENNA X Y Z DIAM
SKA0001 -2781116.248 5068884.493 -2680845.279 dish_2m_unpol 5 90 350
SKA0002 -2780931.295 5068861.322 -2681080.944 dish_2m_unpol 5 90 350
SKA0003 -2780362.166 5069180.303 -2681068.123 dish_2m_unpol 5 90 350
SKA0004 -2781009.073 5069176.864 -2680403.600 dish_2m_unpol 5 90 350
SKA0005 -2780720.855 5068957.308 -2681117.739 dish_2m_unpol 5 90 350
.
.
.
```

B Conversion from LST to GHA

To convert from Local Sidereal Time (LST) to GHA, use the following recipe:

1. Convert the LST to decimal hours.
2. Convert the longitude difference between your observatory and Greenwich (which is at 0 deg longitude). Convert the result from decimal degrees to hours of time time by dividing by 15.
3. If the observatory is at West longitude, add the result to the LST; if it is at East longitude, subtract. If the result is greater than 24, subtract 24; if the result is negative, add 24. The result is the GHA in hours.