SimCa 101

Ilse van Bemmel, Maaijke Mevius

Version 0.6, February 9, 2009

Introduction

In this document we describe how to perform simulations of ionospheric effects for LOFAR. We assume some basic knowledge of MeqTree, as well as familiarity with LOFAR. The user should have access to a version of Timba and Frameworks. This document was written with svn version 6727. Some basic Python knowledge is recommended to read the scripts. The user should have access to the most recent MeqTree distribution, and the most recent Waterhole and Cattery distributions, which are found in the same repository. At the moment a working version of AIPS++ glish is required as well. In a later stage we hope to replace this with the CASAcore libraries.

0.1 The Measurement Set (MS)

The basic input for every simulation is a measurement set (MS), the default data format of AIPS++. A MS is in essence similar to a FITS file, but with different formatting. Inside the MS the *uv* data is stored in 3 columns: DATA, MODEL_DATA and CORRECTED_DATA. For the simulations a MS is produced using glish, the AIPS++ scripting language. For LOFAR, the script reads the station positions and effective areas from a file, taking into account the different sensitivity for each frequency. In addition, it allows the user to set the epoch, location of the phase center, frequency, bandwidth and other instrumental parameters. It is possible to generate a number of MS for different frequencies in one go. Note that for practical reasons the core of LOFAR in the MS is located at WSRT and not at the proper location near Exloo. This should not affect the simulations, since the positions in the file are all relative.

0.1.1 Making your first MS

Make a directory to house your collection of MSs, and go in there. From the directory Waterhole/contrib/IVB/MSets/ copy or link the scripts sim_LOFAR.g and mkcomps.g, and the station position file LOFAR_posn_diam.txt. Then type `glish -l mkcomps.g` at the command line. The script writes to the directory from which it is called. If this is ready, you should have a file named newmodel.cl. Then type `glish -l sim_LOFAR.g` and wait a while. Depending on the settings it can take the blink of an eye or up to several minutes. Now there should be a directory named LO20_FirstSIM_lotsofnumbers.MS. Congratulations, you've generated your first MS!

0.1.2 Advanced MS making

For each new simulation it is recommended to make a new MS, although at times it may be preferable to just use another data column in an existing MS. Be aware that MeqTree will not warn you if you overwrite a previously generated column. In the sim_LOFAR script the user has full control over all the MS parameters. For new users there is a clear section of adjustable parameters. Take note of the formatting, there is no logic here. The LOFAR_posn_diam.txt should not be altered unless you know exactly what you're doing. Updates of this input file will become available in the repository if parameters change.

- Phase center in right ascension and declination, and the date of observation.

```
RA0          := '7h0m00.0';
DEC0         := '50d00m00s';
epoch0       := '2000/01/01';
```

- Frequency, bandwidth and number of channels. The script includes sensitivities for the standard LOFAR frequency bands of 15, 30, 45, 60, 75, 120, 150, 180, 210 and 240 MHz. If the frequency is set to another value, the scrip may fail or produce rubbish. To set the bandwidth, either select the bandwidth fraction (default setting), or comment this line and uncomment the next to give an array of bandwidth. The bandwidth array allows the user to specify the bandwidth for each frequency independently, and should therefore have the same length as the frequency array. The bandwidth is divided equally over the number of channels.

```
# The frequency is an array, e.g. [15, 30, 45, 60, 75]
Frequency    := [60] #Central frequency in MHz
Freq_unit    := 'MHz'
BW_fraction  := 15  # BW = Freq/BW_fraction
# BW := [1, 2, 3, 4, 5] # Array for bandwidth
n_chan       := 32
BW_unit      := Freq_unit
```

- The integration time and total observting time. Note that t_int requires a text string as input, while t_obs is a number. By default the observation is through the meridian, with equal time spend on either side of it.

```
t_int      := '20s' # integration time per scan
t_obs        := 1800  # total observing time in seconds
```

- Each MS is automatically placed in the directory from which the script is called. The goalname can be changed for each new MS. The MS name also includes number of stations, frequency, bandwidth, observing and integration time to facilitate multiple simulations with the same goalname.

```
goalname    := 'FirstSim'
```

- Number of stations. A selection can be made of any number up to 49 stations. Realistic LOFAR arrays will include 20 stations (phase 1 LOFAR), 36 stations (full Dutch LOFAR) or 49 stations (Dutch LOFAR in case more funding becomes available).

```
NN      := 20
```

0.2 SimCa

The main module for the simulation is part of the Lions framework, and called SimCa, which stands for Simulation and Calibration. It is found in Frameworks/Cattery/Lions/, along with all its helper modules. In this manual we focus only on the simulation aspect. SimCa depends on a range of other modules, which all perform a certain task. E.g. there are modules to generate a source distribution on the sky, and of course there are modules to generate an ionospheric screen. Only the main SimCa module is loaded in the MeqTree browser, then compiled with the proper settings, and finally the required jobs are executed to perform the simulation or produce an image. You can copy the script to your working directory to use and adjust it as prefered. For advanced usage, some helper modules may need to be changed. These can then also be copied to the user directory and adjusted there.

0.2.1 Setting up and compiling the simulation

We will now describe the setup of the simulation, and guide you through a basic setup. First thing to do is to make a directory in which your simulation scripts will live. In this directory make a copy of the SimCa.py script found in Frameworks/Cattery/Lions/. Start the MeqBrowser by typing `meqbrowser.py` at the command line and start a meqserver from the pop-up screen. From the TDL menu select Load TDL and then select the file SimCa.py. A TDL Compile-time Options window will appear, as shown in Fig. 1. This window is the heart of setting up a simulation, as it enables you to set the values of each and every parameter by clicking on it. For many parameters a selection can be made from sensible default values. In addition, user specific values can be filled in for some parameters. As MeqTree advances, the look of this window may change. Go with the default settings if things do not make sense. Note that after hitting the Compile button, all values are stored in a local file .tdl.conf. When you open the script again, it will automatically return to your previous settings. If things go crazy, remove the .tdl.conf file to return to the default settings.

- MS: Name of the MS. Clicking on this field will look for directories ending with MS in your working directory. You can browse to where your MS live, or link your MS to the simulation directory for easy access. Click on this and select the MS you have just produced.
- Antenna subset: selects a subset of antennae to be used in the simulation. Leave this set to None, to select all antenna in the MS.
- Correlations: selects the correlations to be simulated. By default all four correlations are generated: XX, XY, YX and YY. Other options are available, but have not been tested for ionospheric simulations.
- What do we want to do: The script can both simulate and calibrate. The latter is the default option, so ensure that this is set to simulate. This will also change the next option.
- Simulation options: If you don't see this, make sure you have set What we want to do to simulate. The simulation options allow to add noise to a simulation, or to add more sources to an existing simulated MS. The option 'Add sky model to existing data' will soon be superseded by the UVBrick module, which is discussed below under 'Sky model'. The noise option adds baseline noise to the simulation. The flux in Jy is the FWHM of the Gaussian noise. For now leave this set to None, but for more realistic simulations, you can add a given fraction of the total source flux in your sky model.
- Measurement Equation options: Allows you to set a number of things related to the ME. In some cases you may want to include time and bandwidth smearing, but we choose to ignore these settings for the purpose of simplicity. The UVW coordinates can be computed, or read directly from the MS. The latter option is the simplest, so we will use the default.
- Sky model: defines the distribution of sources on the sky. There are three modules available to produce a sky model, each uses a different method of compiling the model. For bright and complex sources the MeowLSM module is most accurate, as it deals properly with image plane effects and w-projection. For large numbers of faint sources the OMS.fitsimage_sky should be used to speed up the simulation. These two modules can be combined to produce the best sky model. For testing purposes there is an option to make very unrealistic source grids with Lions.gridded_sky.
  - MeowLSM reads a source catalog from a file, and is available as part of the Meow framework. The LSM file needs to point to your actual catalog file. The module supports a

range of preset catalog format types, which can be selected in the File format field. Two standard primary beam expressions are available, but have not been tested for ionospheric simulations. For large catalogs you need to select a subset, which will automatically select the x brightest sources. For a reasonable simulation speed the total number of sources should not be much larger than 20. By checking the Show LSM GUI box, a window will appear during compilation which shows the location of the sources on the sky, with a colour and size coding related to the source brightness. All the other boxes should be left unchecked.

- Siamese.OMS.fitsimage_sky uses the UVBrick module to read in a fits file which includes a large number of sources. It performs an inverse Fourier transform and interpolation to produce the *uv*-data. Although this method works in combination with the ionosphere module, it does not deal with the effect properly. This will be implemented shortly.
- Lions.gridded_sky can produce a range of virtual layouts of point sources. There are several grid options. A 3x3 single grid will produce 3x3 sources at fixed distances in RA and DEC, with the central source at the phase center. A 3x3 double grid will add 4 sources in the middle of each square formed by 4 sources of the 3x3 grid, totalling 13 sources. In this module the source flux and grid-step are identical for all sources.

For now we will produce a grid. Select the Lions.gridded_sky module, choose 'single_grid_model, number of sources is 3, grid step is 60 arcmin, source flux is 10 Jy. Leave the remaining options set to their defaults.

- Export sky model as kvis annotations: use this option to automatically generate an annotations file for your source model, which can be loaded in kvis.
- Use G Jones (receiver gains/phases): this is only used when calibrating, leave this unchecked for simulations.
- Use B Jones (bandpass): this is only used when calibrating, leave this unchecked for simulations.
- Use G_sim Jones (simulate ...): Allows for including a variation in instrumental gain during an observation. The module itself is part of Siamese. The variations can be specified independently for amplitude and phase, by selecting either a random error or a periodical fluctuation. The user can define the limits between which the variations occur, and the timescale of the periodical fluctuations. For the simulation we will leave this unchecked.
- Use Z_Jones (iono): This is the heart of the SimCa module, allowing the simulation to include an ionospheric screen. Inside this option there is MIM model, which includes a range of ionospheric phase screens available for simulation and calibration. At the moment all screens are assumed to be thin, and fixed at an altitude of 300 kilometers. The Create Log Nodes option generates text files with the simulated phases for later inspection, leave this unchecked. Rotate frame can force the module to use a given reference antenna, this should be set to None, which is equivalent to using the first antenna as the reference. There are currently four models for including an ionosphere:
  - TID_MIM: models one or two sine waves traveling over the array. For each wave the wavelength, speed, angle of propagation and relative amplitude can be set. A single wave is produced by setting the amplitude of the second wave to 0. The background TEC value should be larger than 0. For each parameter the user can select from a range of typical values. This is the most tested option and recommended for starting. For proper inclusion of Earth curvature, make sure to check the box Use Longitude/Lattitude. When using this

module in combination with the UVBrick module, it only calculates ionospheric phases for the phase center and applies them to the entire field. This is only realistic for small fields of view.

- Kolmogorov_MIM. This module generates a Kolmogorov turbulent screen using a separate Python script. The screen is identical for each time interval, but can be moved over the array with a given speed and direction. The turbulence parameter β can be set, and should be 5/3 for 2-dimensional Kolmogorov turbulence. The number N defines the number of pixels in x and y direction of the screen. After the screen is generated, it is moved with a velocity specified by Speed in either direction. Scale and amplitude are used to ensure the resulting TEC values are sensible. Setting a value for 'Seed' will generate a screen with the same random seed number each time. When set to None, each simulation will have a different screen. This module still needs to be tested properly, and may not produce sensible results instantly.
- Poly_MIM: produces a polynomial screen. The polynomial rank can be given in both directions, and it is recommended to use long/lat projection here. This module is mainly used for calibration purposes.
- KL_MIM: will produce a Karhuenen-Loève screen, for calibration purposes only.

For the first simulation select the TID_MIM and set the values as shown in Fig. 2. Also make sure you have checked the main Use ZJones (iono) box.

- Use GD Jones: Leave this unchecked for simulations

When you have checked the settings for your simulations, press Compile. The meqbrowser window will now show you the source on the right and the tree on the left. You can browse through the tree by opening up the root nodes, especially the VisDataMux can be useful to check. The total number of nodes should be 15800 when using Timba version 6727, this may change in future versions. However, it is always useful to check if the number of nodes corresponds to your expectations, e.g. a heavy simulation with lots of sources, lots of antennae and an ionosphere should not have on the order of 1000 nodes. If you are suspicious, check your compile settings and the tree in depth.

0.2.2 Running and imaging

Depending on your machine, you will see a new window appear in a minute or so. This is the TDL Jobs & Runtime Options menu. You can also get this by pressing 'Exec' or 'TDL Exec' in the meqbrowser window. In this window. In this window you will first need to select MS/data selection options. For simulation only the 'Output' column is important, which is where MeqTree will write the simulated visibilities. Note that MeqTree *will* overwrite the column you select. The tile size allows the processing to be done faster, by combining more time steps in one iteration. In general: the higher this value, the faster your simulations. For this simulation set the 'Input' and 'Output' column both to DATA. Select the tile size to be 10, and leave the rest as is. Now you are ready to run the simulation, by pressing Simulate. In the bottom bar of the meqbrowser window you will see the progress of the simulation, which should take no more than a minute on a good machine.
If you managed to get this far, congratulations! You now have your first ionospheric simulation, but there is nothing to show for it yet. Next you want to make an image and have a look at what you just produced. Press the TDL Exec button again, and open the Imaging options. The menu inside should look like Fig. **??**. These options are all used to call an external imager, either the CASAcore lwimager, or the AIPS++ imager.

- Imager to use: if you have the lwimager, use it. It is a lot faster.

- Image type or column: this should be automatically set to the same value as 'Output column', but check this each time you image.
- Frequency channels in image: if you have a multi-band MS (which we do), you can choose to average the channels or not, or even do multi-frequency synthesis. You can also manually select channels, but this option has not yet been tested. For now, select the option '1 (average all)'.
- Imaging weights: selects the way the *uv*-data are weighted. This option has not been tested for ionospheric simulations, but it is a default AIPS++ option and should not provide any problems. The default is natural weighting.
- Apply Gaussian taper: pretty self-evident, but again this has not been tested as part of the Lions framework.
- Stokes parameter to image: you can image just Stokes I, or all four Stokes parameters. Leave this set to I.
- Image size: the next two options define the scale and pixelsize of the image. Note that the image size in pixels needs to be identical to the size of the fits image used if you used 'OMS.fitsimage_sky' option for the simulation. The size in arcmin should automatically be adjusted to include the entire grid of sources if you used 'Lions.gridded_sky'. For the 'MeowLSM' sky model make sure the size in arcmin includes all your sources.
- Image padding factor: should always be set to 1
- Enable w-projection: for large fields this box should be checked and the number of convolution functions inside should be set to 128 or even 256 for very large fields.
- Phase center: allows you to shift the image phase center. Check the comments in this option on how to do that, but leave set to 0 for now.
- Use custom MS selection: you can (in principle) select a part of the MS for imaging. This requires knowledge of the query language for glish and is not straight-forward. Leave this unchecked.

All values should now be properly set, so click Make a dirty image. The imager produces screen output in your terminal window, allowing you to check the basic imaging settings when you scroll up. It will take a few seconds to produce the image. A kvis window will appear when it is done. In kvis, set the flux range from -0.5 to 2.5. The image should look as displayed in Figure 4.
NEED TO UPDATE THE FIGURES!!

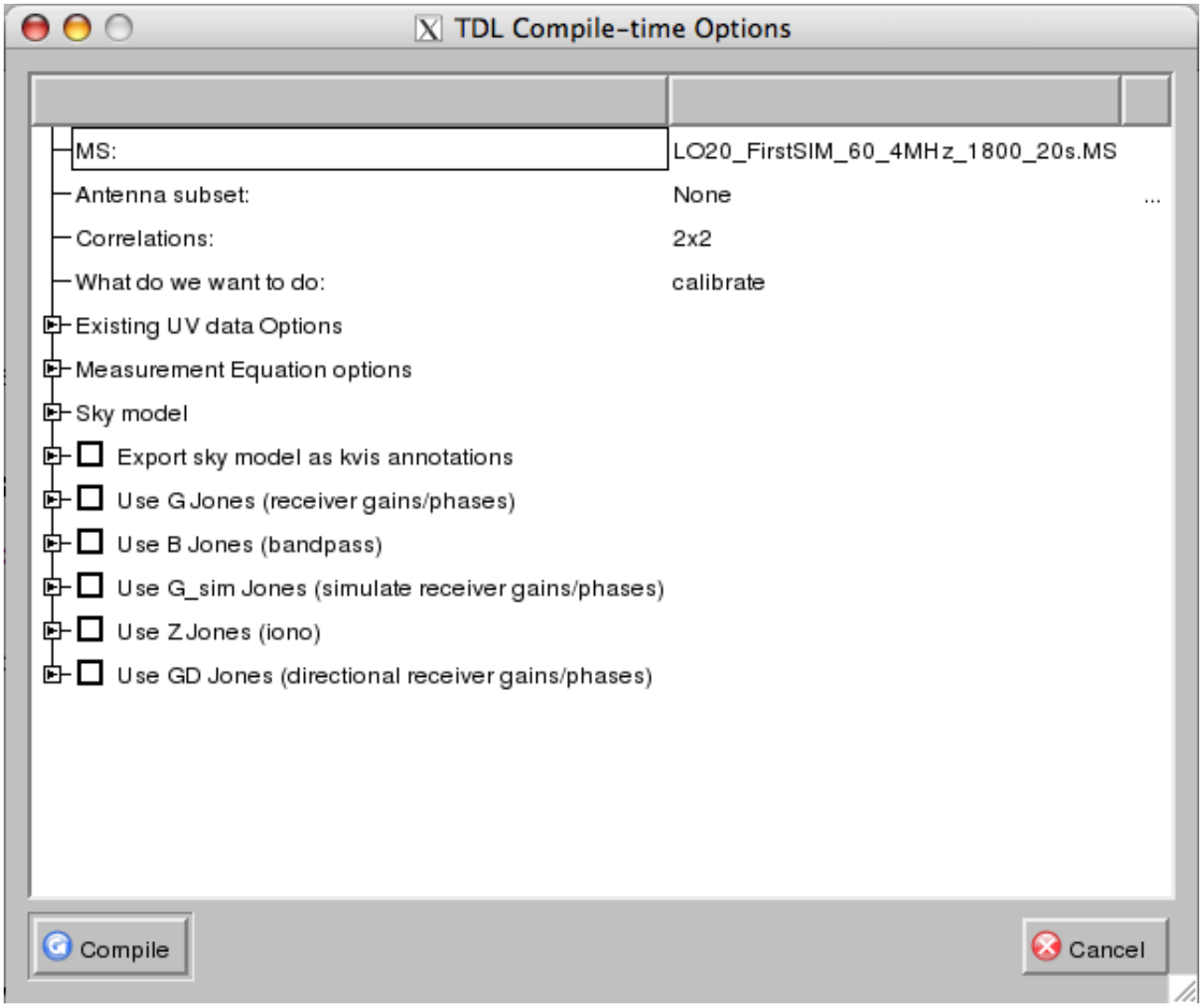

Figure 1: Screen shot of the TDL compile options after loading SimCa.py in the MeqBrowser.

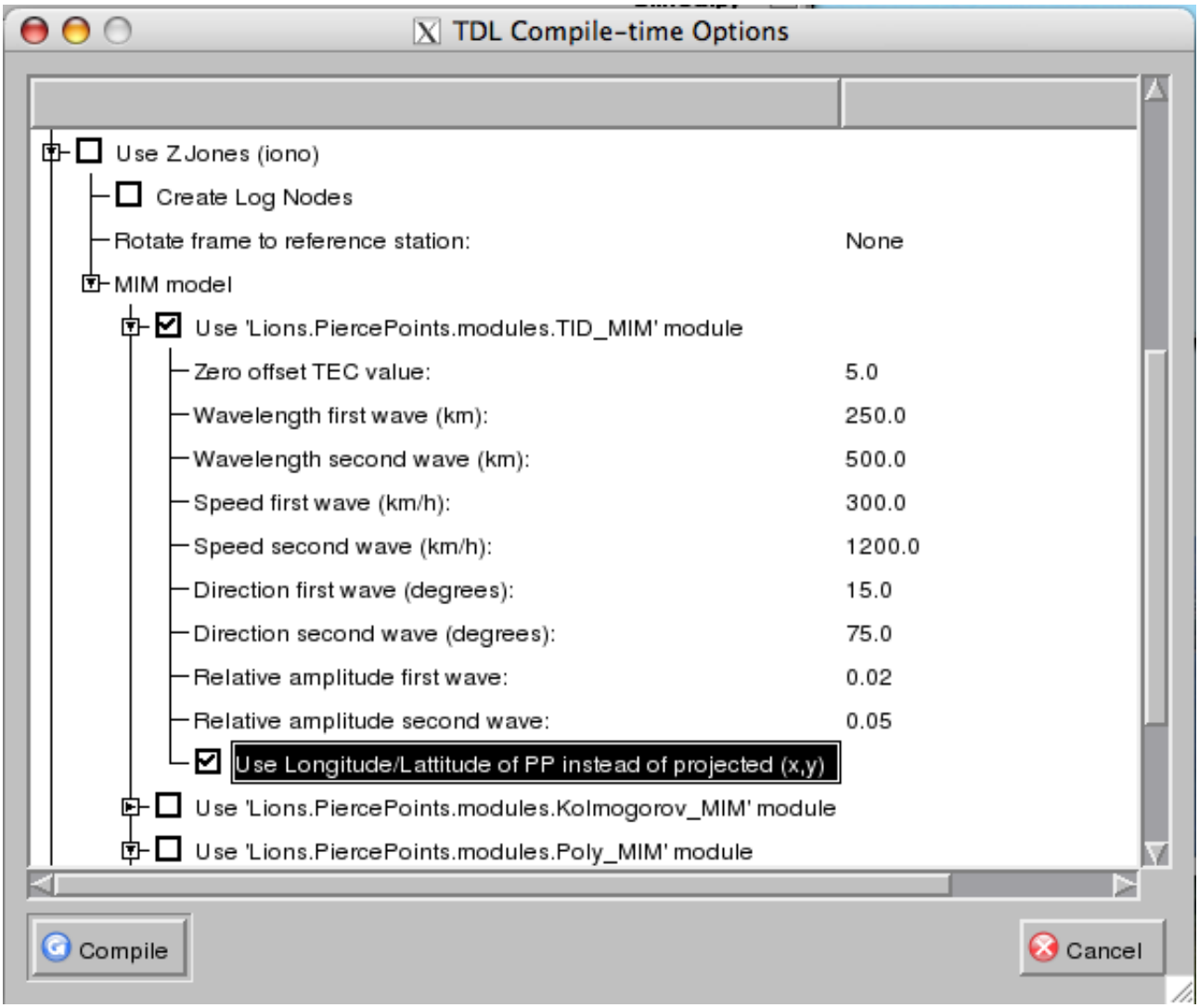

Figure 2: Screen shot of the settings for Z Jones.

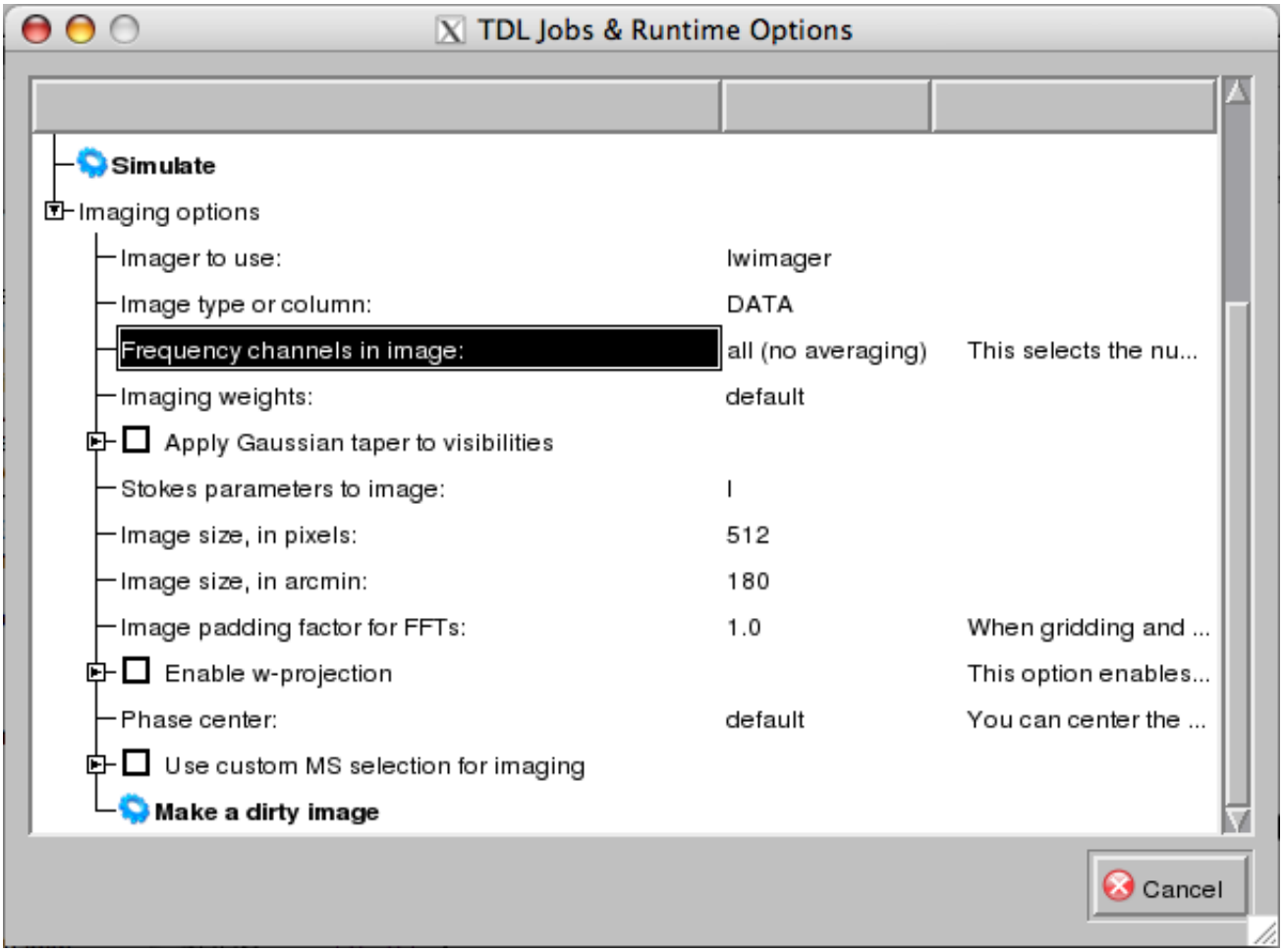

Figure 3: Screen shot of the imaging options in the TDL Jobs & Runtime Options

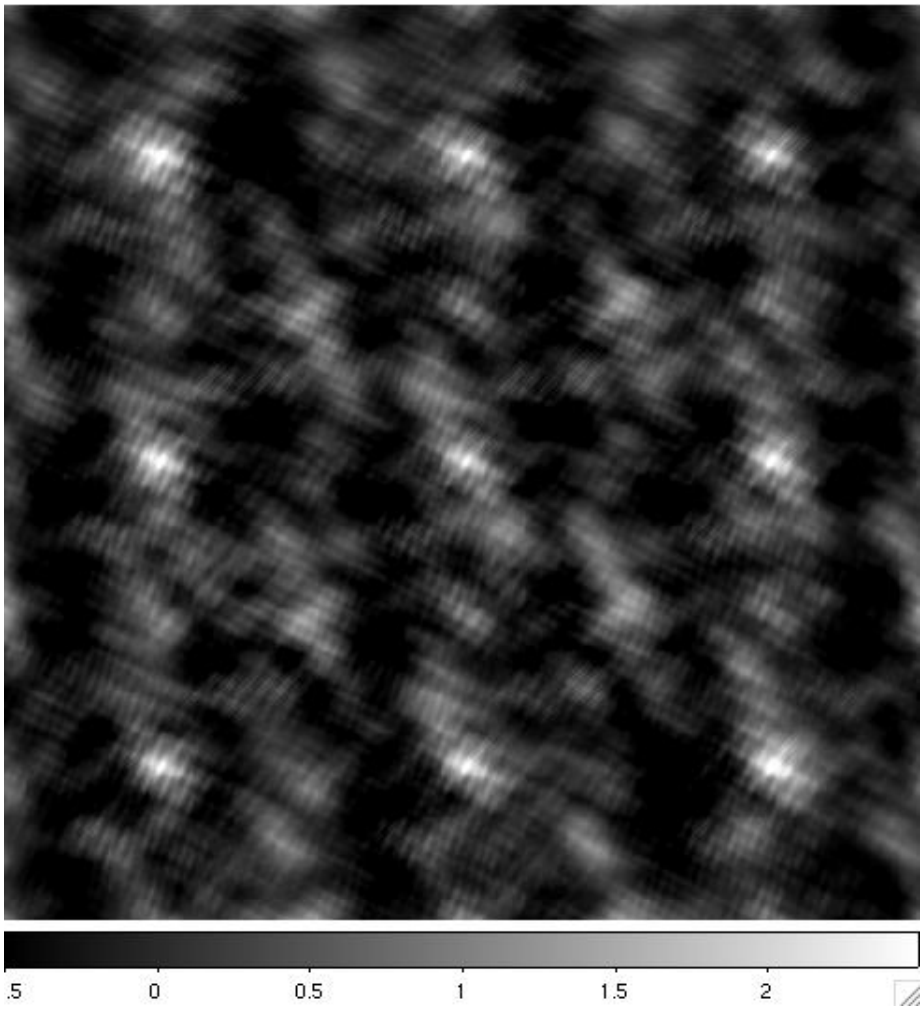

Figure 4: Resulting image of the simulation as described in Section **??**.