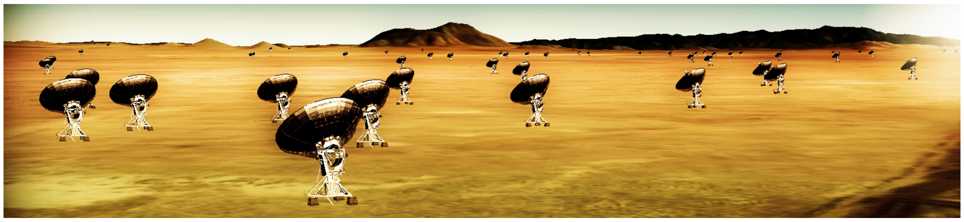

Instructions for using CASA simulator for the ngVLA

C.L. Carilli, E. Greisen, K. Nyland, & R. Indebetouw (NRAO)
original: November 2, 2016; revised: Sepetmber 2017

1 Introduction

Following are simple instructions as to the use of idealized configuration files in generating mock observations using CASA.

The idealized .cfg table is a simple ascii file with x,y,z positions, plus antenna diameter and name. The x,y,z positions are in meters relative to Earth-center, using standard coordinates defined in the FITS WCS[1].

In the latest version of CASA (5.1), SIMOBSERVE should use the antenna diameter in the .cfg file in both calculating the visibilities in the .ms, and in the CLEAN PB correction step. The NGVLA is now in the Observatory table in CASA.

2 SIMOBSERVE

The task SIMOBSERVE in CASA takes a FITS image model as input, plus the .cfg file, and generates a measurement set.

FITS model: The FITS image can be a spectral cube or a single image, and simobserve will generate a measurement set with the correct number of spectral channels given in the header of the input FITS image. The frequency of the mock observations is read from the FITS header. The flux density unit in the FITS image needs to be Jy/pixel.

The pixel size needs to be small enough that the outer spacings of the array are reasonably sampled, otherwise simobserve crashes. This can be a problem for mock observations of large scale structures at low resolution. One solution is to resample the FITS image to small pixels, making a really big image. Make sure to maintain the Jy/pixel scaling in this case (smaller

[1]see https://fits.gsfc.nasa.gov/fits_wcs.html and references therein

pixels mean less Jy). A second solution is to remove the outer antennas from the .cfg file. We include the SWcore configuration in the package to facilitate low resolution simulations. This configuration includes just the 114 core stations in the SW array, within 2km diameter.

The FITS file also needs to have the the proper coordinate strings, such as: "RA—SIN' 'DEC–SIN', STOKES, etc...

Following are inputs to SIMOBSERVE. This mock observation used 100s records, and a total integration time of 1000s centered at transit. We have removed many of the parameters that could be default (use: 'default simobserve' to begin).

```
simobserve(project="TEST",skymodel="PBTEST.FITS",complist="",
compwidth="1GHz",setpointings=False,ptgfile="pnt.txt",
integration="100s",obsmode="int",refdate="2014/05/21",
hourangle="transit",totaltime="1000s",antennalist="SWcore.cfg",
outframe="LSRK",thermalnoise="")
```

The thermalnoise parameter is set to blank, which implies no noise is added. We do not add noise in this step, since there are many parameters applicable predominantly to ALMA, which we are not sure how they would work for the ngVLA (gain gain corrections, atmospheric corrections, etc...).

The channel width and number of channels will be taken from the FITS header, so 'compwidth' is ignored.

The setpointings=False and the pnt.txt is required to ensure that CASA does not try to generates its own pointings to cover the field (except, of course, if you want it to). The pnt.txt file has the format:

#Epoch RA DEC TIME(optional)

J2000 16h30m00.0 024d00m00.0

Simobserve will create a visibility measurement set with no noise, sampled by the array with the given model as the image. The visibilities will be in Janskys.

Adding noise Thermal noise can be added to each visibility based on the integration time and band or channel width. The commands are:

- sm.openfromms("test.ms")
- sm.setnoise(mode="simplenoise",simplenoise="1Jy")
- sm.corrupt()
- sm.done()

To derive the simplenoise scaling, we have followed an empirical process:

- 1. Calculate the expected thermal noise for your final naturally weighted image. The noise performance values as a function of frequency can be found in Table 1 of ngVLA memo 17.
- 2. Start simobserve with a FITS model that has zero flux but otherwise same inputs (integration time) as final simulation.
- 3. Add unit noise per visibility to the resulting zero.ms (simplenoise="1Jy").
- 4. Make a naturally weighted image and derive the rms noise in this test image using the viewer or whatever.
- 5. Calculate the scale factor needed for 'simplenoise' to get to your expected thermal noise (ie. desired noise/test noise)
- 6. Return to the real mock measurement set for the simulation you are performing, and add noise as per step 5 scaling.

Note that the sm.corrupt process corrupts the visibilities, so you should save the original .ms from simobserve before proceeding.

This should yeild a model measurement set that results in a CLEAN image with a naturally weighted noise expected for the proposed observation. However, if there is a bright source in your model that might cause dynamic range issues, you might want to go back to the test/zero image case and rerun, starting from simobserve with the zero model, through setnoise with the scaled simplenoise from step 5, to CLEAN/NA, to check you get the right noise in the end.

At this point, you have a visibility data set with correct flux densities for the source, and correct thermal noise. You can then explore eg. what Briggs weighting does to the final image noise and quality.