UVSim: Obit UV Data Simulator

W. D. Cotton, March 23, 2009

***Abstract*—This memo describes the Obit UV data simulator, UVSim. This task is capable of generating simulated data with Gaussian noise and sky models added. These datasets can be of an arbitrary size, making this task suitable for testing techniques for dealing with large datasets.**

***Index Terms*—interferometry**

I. Introduction

TESTING of techniques for handling large datasets before actual data becomes available needs realistic simulated data. The following describes a task, UVSim, implemented in Obit ([1], http://www.cv.nrao.edu/∼bcotton/Obit.html) which allows simulation of datasets of realistic size and complexity.

II. UV Coverage

The portions of the Fourier (UV) plane covered by an interferometer are determined by the locations of the antennas, the sky frequencies used, the celestial position of the source and and the day and times of the observations. Given a specification of the data to be simulated, UVSim creates a new dataset or appends to an existing one.

Computation of the spatial frequency coordinates, u,v,w, implemented in ObitUVUtil:ObitUVUtilUVW is given by the following:

$$u \;=\; B_x\,\cos(ha) + B_y\,\cos(ha) \qquad 1)$$

$$v \;=\; -(B_x\,\cos(ha) + B_y\,\sin(ha)) \;+\; B_z\,\cos(\delta)$$

$$w \;=\; (B_x\,\cos(ha) + B_y\,\sin(ha)) \;+\; B_z\,\sin(\delta)$$

where δ is the declination of date of the pointing and ha is the hour angle for short baseline interferometers is given by:

$$ha \;=\; GST@0 + longitude + time * rotation\_rate \;-\; \alpha \;\; 2)$$

where $GST@0$ is the Greenwich Sidereal time at midnight of the time system used, $longitude$ is the longitude of the array, $time$ is the time of the observations, $rotation\_rate$ is the Earth's rate of rotation and α is the Right Ascension of date.

The vector (B_x, B_y, B_z) is the vector difference between the two antennas of a baseline where y is east, z is towards the north celestial pole and x the orthogonal direction in a right–handed system (celestial equator on the meridian).

One convention not implemented in UVSim is that of rotating the (u,v) coordinate pair such that north of the standard equinox will align with the declination axis of images derived from the data. This corrects for the effects of differential precession. If this is ever an issue, UVSim should be run with the reference date set to that of the standard equinox (0 Jan 2000).

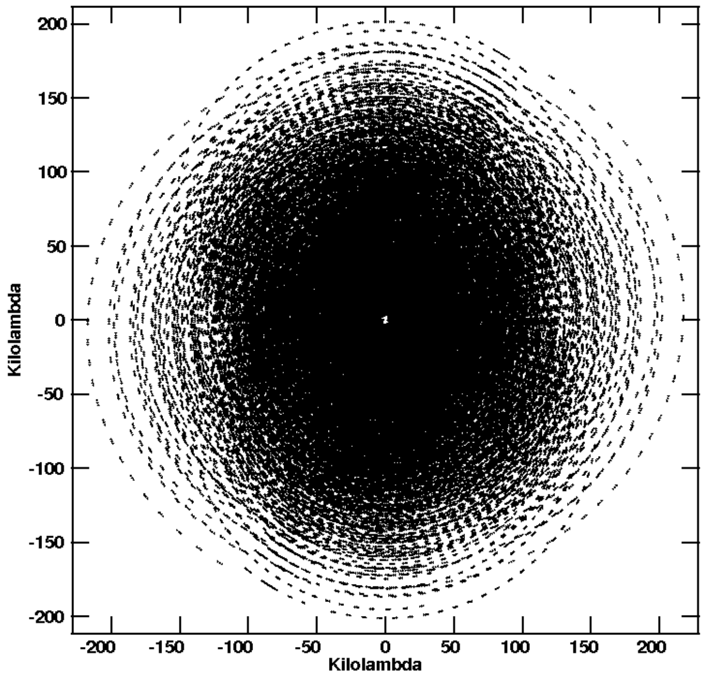

Fig. 1. UV coverage for a single channel from data simulated by UVSim.

The uv coverage can be extended by means of "bandwidth synthesis" in which observations over a range of frequencies is made. An example uv coverage from UVSim for a source at declination 60° is shown in Figure 1.

III. Noise

Noise among baselines and frequency channels is weakly correlated at best so independent, Gaussian distributed noise is an adequate approximation. The Obit implementation uses the GNU Scientific Library (GSL) random number generator gsl_ran_gaussian to add "noise" to the real and imaginary parts of all visibilities.

IV. Sky Model

Realistic data simulations need realistic sky models. The implementation in UVSim allows either single component models or the set of CLEAN components from an actual or simulated observation to be used as the sky model. The implementation uses the ObitSkyModel class to compute the instrumental response to the given sky model. Models which vary with frequency are supported; spectral index as well as

National Radio Astronomy Observatory, 520 Edgemont Rd., Charlottesville, VA, 22903 USA email: bcotton@nrao.edu

Manuscript received ; revised

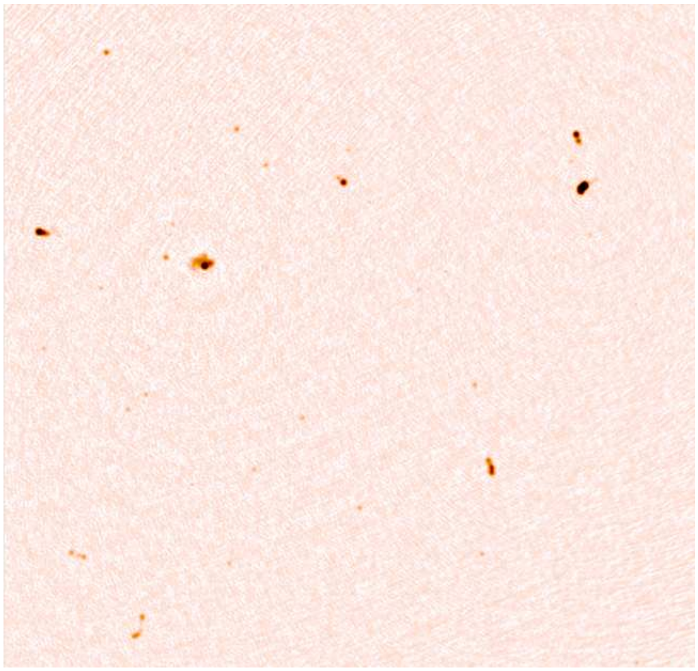

Fig. 2. Portion of an image derived from data simulated by UVSim to which the sky model derived from a moderate depth VLA survey was added. The resolution is $\sim$ 5" and the dynamic range is $\sim$2000. The image is clipped and shown with a $\sqrt{}$ stretch to emphasize low level emission.

curvature terms can be explicitly given in either the single component passed through the task parameters or in the CLEAN component tables attached to the sky model images. Multithread computation of the sky model is allowed.

An example of an image derived from simulated data using an actual measured sky model (CLEAN components) is shown in Figure 2. By construction, such simulations include realistic distributions of flux densities and sizes.

V. UVSim

UV data simulation is implemented in Obit using the task UVSim. In general, UVSim will be run multiple times to generate a simulated data set with one execution per "scan". The on-line documentation for UVSim is given in the Appendix.

UVSim works a scan at a time generating and writing the data with zeroed visibilities and unit weights to a scratch file. Then, if Gaussian noise and/or a sky model is to be added, these operations are performed on the scratch file. Finally, the contents of the temporary scan dataset are appended to the end of the output dataset. This technique allows generating multisource datasets with varying source properties including name, celestial position, noise and sky model.

The frequency coverage is specified as a number of "IFs" of given frequency separation from the reference frequency, each containing a specified numbers of "channels" of given bandwidth. "Integration" time inside of each scan is also specified.

The data set generated by UVSim will not contain either an index (AIPS NX table) or a default calibration (AIPS CL) table. These can be generated using the ObitTalk functions UV.PUtilIndex and UV.PTableCLGetDummy to produce a dataset similar to what could be obtained from an actual observation.

As a test of the feasibility of creating large datasets, UVSim was run simulating a VLA A configuration dataset containing 3.4 million visibilities with full polarzation, 4 IFs of 32 channels each and adding the sky model, part of which is shown in Figure 2. This model consists of 116 facets which were CLEANed for 15,000 iteration. Further, frequency dependent primary antenna gain corrections were applied. The resultant data set was 20 Gbytes in size. The execution time on the Obit development computer, mortibus, using all 8 cores was 2.7 hours. During the majority of this time top showed 799% or more of a cpu usage; this process makes very efficient use of the available parallel compute power.

VI. Discussion

Task UVSim allows simulation of realistic datasets. Such data will be "perfect" in the sense that it does not include real world effects such as atmospheric phase or antenna gain fluctuations. If needed, such effects can be simulated and the inverse written to an "AIPS SN" table. Application of this table to an AIPS CL table which is then applied to the data will add these corrupting effects to the data. Actual, measured variations derived from observations can be used by generating a simulated dataset with scans at the same times as the actual observations and using the derived calibration tables.

Time and bandwidth smearing can be simulated by first generating a simulated dataset using high temporal and frequency resolution. Then, average this data in time and/or frequency sufficiently to cause the desired level of smearing. This can be done using Obit task UVBlAvg.

The timing test for a moderate size dataset shows the speed of UVSim to be sufficient to generate large databases; even with moderately complex sky models.

Appendix

The "help" and "explain" segments for the on-line documentation for UVSim are given. The python code to define the various VLA configurations are given.

References

[1] W. D. Cotton, "Obit: A Development Environment for Astronomical Algorithms," *PASP*, vol. 120, pp. 439–448, 2008.

```
UVSim
Task:
   Create a UV data set with specified parameters.  If the output
file exists, new entries will be appended to the end of the file.
Data are given zero visibility and unit weight.
A run of UVSim essentially creates a single scan, multiple scans
with different multiple sources are allowed.
    NOTE: This does not rotate in [u,v] such that north of the
standard equinox (J2000.0) is aligned with the v axis.
If this matters, use refDate="2000-01-01"
   If multiple scans are included, they should be added strictly in
time order, else the data should be sorted.
For true multisource files, the data should be indexed.
   Optionally noise and a source model can be added.
Adverbs:
  refDate....Reference day as "YYYY-MM-DD", best if "2000-01-01"
  timeRange..Time range of the data to be simulated. In order:
             Start and end times in days relative to ref. date.
             Use  dhms2day to convert from human readable form
             Times are wrt 0 UT on refDate
  delTime....Time increment between data samples (sec)
  Source.....Name of source
  RA.........J2000 Right ascension of source (deg)
             Use UVDesc.PHMS2RA to convert from human form
  Dec........J2000 Declination of source (deg)
             Use UVDesc.PDMS2Dec to convert from human form
  minEl......Minimum elevation (deg) to include
  refFreq....Number of frequency channels
  nFreq .....Reference Frequency (Hz) at Ch 1
  delFreq....Channel bandwidth
  nIF .......Number of IFs
  delIF......Frequency increment between IFs
  arrayXYZ...Earth centered XYZ (m) of array center, default=VLA:
             [-1.601185365e+06,-5.041977547e+06,3.554875870e+06]
  nAnts......Number of antennas in AntXYZ
  antXYZ.....Array of XYZ (m) offsets from ArrayXYZ
             See Explain for some samples

  Noise......Gaussian sigma of noise to add to each visibility

                   Source model
  in2Name....Model AIPS name .      Standard defaults.
  in2Class...Model AIPS class .     Standard defaults.
  in2Seq.....Model AIPS seq. # .    0 => highest.
  in2Disk....Disk drive # of model (FITS or AIPS). NO default
  in2File....FITS input root if Type=='FITS'
             Any digits should be left off the end of the name as
             the 0-rel field number is added (no leading zeroes).
  nmaps......Number of image files to use for model.  If more than one
             file is to be used, the Name, Class, Disk and Seq of the
             subsequent image files will be the same as the first file
             except that the LAST two characters of the Class will be
             '01' thru 'E7' for files number 2 thru 4192.  Maximum 4192.
  CCVer......CC file ver. number.          0 => highest.
  BComp......The first clean component to process. One value is
             specified for each field used.
  EComp......highest CLEAN comps. to use for model. ALL 0 => all.
             This array has one value per field up to 64 fields.  All
             components are used for fields > 64.
```

```
            If any EComp[i] < 0, then components are only used up to
            the first negative in each field.
  Flux.......Only components > Flux are used in the model.
  Cmethod....This determines the method used to compute the
            model visibility values.
            'DFT' uses the direct Fourier transform, this
            method is the most accurate.
            'GRID' does a gridded-FFT interpolation model
            computation.
            '    ' allows the program to use the fastest method.
  Cmodel.....This indicates the type of input model; 'COMP' means that
            the input model consists of Clean components, 'IMAG'
            indicates that the input model consists of images.  If
            Cmodel is '    ' Clean components will be used if present
            and the image if not.  Note that Clean images do not make
            good models.  The Clean components have been convolved with
            the Gaussian Clean beam making their Fourier transform be
            rather tapered compared to the original uv data.
  Factor.....This value will be multiplied times the CLEAN component
            flux densities before subtraction.
            If the image is not in Jy/pixel, Factor should be
            used to convert to those units.  Note that a Gaussian beam
            has area 1.1331 * Major_axis * Minor_axis / (axis_incr)**2
            pixels for square pixels.
  modelParm..Other model parameters
            modelParm[3] = 0; Point - no other parameters
            modelParm[3] = 1; Gaussian on sky:
              [0:2] = major_axis (asec),  minor_axis (asec),
              position_angle (deg),
            modelParm[3] = 3; Uniform sphere:
              [0] =  radius (asec)
            modelParm[4+] = spectral index, curvature...
            If giving spectral terms, add 10 to  modelParm[3]
  mrgCC......If True, then merge each CC table CCVer before
           . subtracting (one per field). In each table, all
           . components on a given cell have their fluxes summed
          .  resulting in a single entry per cell,
  PBCor......If true, apply frequency dependent primary beam
          .  corrections.  Default True
  antSize....Primary antenna size (m) to use for PBCor def. 25

  outDType..'FITS' or 'AIPS'  type of output
  outName....Output UV AIPS file name    Standard defaults.
  outClass...Output UV AIPS file class.  Standard defaults.
  outSeq.....Output UV AIPS file seq. #. 0 => highest unique.
  outDisk....Disk drive # of output UV (FITS or AIPS) NO default
            0 FITS => current directory
  Compress...If True  the output data is written in
            compressed format which can result in a substantial
            reduction in disk space needed.
  nThreads...If the Obit libraries are compiled with multiple
            thread operation enabled, this parameter sets the
            number of threads that can be used for parallel
            operations.
            NB: This only improves performance if there are
            multiple processors and/or cores in the host.
  noScrat....A list of AIPS disk numbers on which you do not
            wish scratch files
```

```
UVSim:

                          PURPOSE

     This task is used to generate specimen u-v coverage for
an interferometric array, given an array configuration
specified by the user.
                           LIMITATIONS

    At present, the program is mainly applicable only to compact
array configurations - as opposed, say, to VLBI arrays.  It
assumes, for example, that the elevation angles of all array
elements are identical.  However, it is expected that minor
embellishments will be added to UVSim, removing such
restrictions as the need arises.

                 VLA Antenna locations

# VLA A configuration: 1 missing
uvsim.nAnts=26
uvsim.arrayXYZ = [-1.601185365e+06,-5.041977547e+06,3.554875870e+06]
uvsim.antXYZ = [ \
 -10577.4465 ,  -1651.4898 ,  15618.8354, \
   1836.9848 ,   6942.0883 ,  -2665.1917, \
   3242.0079 ,  12366.0699 ,  -4767.4174, \
   1005.4658 ,  -2642.9940 ,  -1472.1762, \
  -8642.1073 ,  -1349.2745 ,  12760.7237, \
    152.7719 ,   -401.2756 ,   -223.3878, \
   4857.9405 ,  19090.2991 ,  -7275.8373, \
   3275.1757 ,  -8682.4843 ,  -4854.8454, \
   -243.5707 ,    -38.0432 ,    360.0683, \
   5349.1927 , -14224.3335 ,  -7964.4512, \
   2394.9882 ,  -6350.0140 ,  -3550.9598, \
   4083.2632 ,  15550.4716 ,  -5990.4377, \
   4259.0322 , -11311.4724 ,  -6330.0044, \
  -2629.0842 ,   -410.6900 ,   3885.6318, \
   2495.7864 ,   9491.9196 ,  -3654.6512, \
  -3854.6611 ,   -602.3057 ,   5698.9363, \
  -5271.2704 ,   -823.5763 ,   7791.9868, \
    499.8678 ,  -1317.9892 ,   -735.1819, \
   1253.2737 ,   4733.6305 ,  -1816.9039, \
   6536.2953 , -17410.1698 ,  -9755.5295, \
   1640.0547 ,  -4329.9410 ,  -2416.7002, \
    765.2475 ,   2889.4489 ,  -1108.8703, \
  -1660.4443 ,   -259.4071 ,   2454.4650, \
   -801.3897 ,   -124.9614 ,   1182.1372, \
  -6870.9088 ,  -1072.9460 ,  10148.7744, \
    377.0253 ,   1440.9950 ,   -556.1007
]
```

```
# VLA B configuration:
uvsim.nAnts=27
uvsim.arrayXYZ = [-1.601185365e+06,-5.041977547e+06,3.554875870e+06]
uvsim.antXYZ = [ \
   -3217.5611 ,   -502.6561 ,   4756.1486, \
    -801.3765 ,   -124.9634 ,   1182.1330, \
     -74.7738 ,    -11.7642 ,    111.6307, \
     998.6885 ,   3764.3286 ,  -1443.4370, \
     114.4582 ,    438.6920 ,   -169.4598, \
     377.0138 ,   1440.9986 ,   -556.0993, \
    -243.5732 ,    -38.0377 ,    360.0674, \
   -1660.4424 ,   -259.4055 ,   2454.4706, \
    1005.4526 ,  -2643.0090 ,  -1472.1602, \
    1640.0438 ,  -4329.9487 ,  -2416.6938, \
     560.1301 ,   2113.2402 ,   -810.6614, \
    2000.0969 ,  -5299.7823 ,  -2962.8613, \
      35.6570 ,    133.6496 ,    -51.0682, \
     765.2420 ,   2889.4443 ,  -1108.8804, \
     306.2086 ,   -804.5630 ,   -448.0830, \
    1534.5962 ,   5793.8966 ,  -2223.4635, \
    -489.2698 ,    -76.3170 ,    721.5282, \
   -1174.2759 ,   -183.3309 ,   1734.2316, \
    1253.2710 ,   4733.6263 ,  -1816.8921, \
      46.9538 ,   -122.0062 ,    -67.5933, \
     152.7782 ,   -401.2547 ,   -223.3981, \
     499.8570 ,  -1317.9902 ,   -735.1997, \
     733.3805 ,  -1932.9750 ,  -1078.0972, \
   -2629.0753 ,   -410.6669 ,   3885.6271, \
    1316.4566 ,  -3443.3143 ,  -1913.5336, \
   -2091.4340 ,   -326.6029 ,   3089.4287, \
     229.4939 ,    879.5950 ,   -339.8485
]
```

```
# VLA C configuration:  1 missing
uvsim.nAnts=26
uvsim.arrayXYZ = [-1.601185365e+06,-5.041977547e+06,3.554875870e+06]
uvsim.antXYZ = [ \
   -357.6659 ,      0.0000 ,    527.8260, \
    223.9773 ,   -588.4632 ,   -327.6884, \
   -801.4043 ,   -124.9802 ,   1182.1236, \
     93.5248 ,   -244.9877 ,   -136.2203, \
    306.1735 ,   -804.5809 ,   -448.0776, \
     46.9103 ,   -122.0392 ,    -67.6047, \
     14.7699 ,    -37.1317 ,    -20.2138, \
    611.7688 ,  -1613.2995 ,   -899.9988, \
    152.7598 ,   -401.2719 ,   -223.4056, \
    299.6950 ,   1145.9177 ,   -442.2658, \
   -489.3108 ,    -76.2825 ,    721.5120, \
    464.0856 ,   1763.7486 ,   -678.8806, \
    499.8440 ,  -1317.9851 ,   -735.2095, \
   -637.5117 ,    -99.3965 ,    939.9327, \
    -30.0572 ,     -4.8033 ,     45.7094, \
    229.4504 ,    879.5782 ,   -339.8561, \
     11.3010 ,     40.6734 ,    -15.1929, \
    167.3765 ,    643.3078 ,   -248.8691, \
     70.6521 ,    267.7614 ,   -102.8906, \
   -243.6045 ,    -38.0461 ,    360.0346, \
    -74.8225 ,    -11.7603 ,    111.6133, \
    114.4365 ,    438.6881 ,   -169.4949, \
    376.9899 ,   1440.9908 ,   -556.1276, \
     35.6144 ,    133.6478 ,    -51.0924, \
   2495.7479 ,   9491.9223 ,  -3654.6949, \
    398.2386 ,  -1048.1359 ,   -584.1735
]
```

```
# VLA D configuration:
uvsim.nAnts=27
uvsim.arrayXYZ = [-1.601185365e+06,-5.041977547e+06,3.554875870e+06]
uvsim.antXYZ = [ \
   -148.4695 ,    -23.2243 ,    219.9841, \
    121.6138 ,   -319.1300 ,   -177.5794, \
     51.8719 ,    195.8316 ,    -75.1046, \
     14.7867 ,    -37.1284 ,    -20.2165, \
     68.6063 ,   -179.2495 ,    -99.5131, \
     46.9103 ,   -122.0392 ,    -67.6047, \
     93.5089 ,   -244.9958 ,   -136.2176, \
    186.8100 ,   -491.1257 ,   -273.5405, \
    152.7598 ,   -401.2719 ,   -223.4056, \
     21.9946 ,     81.5319 ,    -30.9473, \
     35.5896 ,    133.6463 ,    -51.0933, \
     35.5896 ,    133.6463 ,    -51.0933, \
     91.5248 ,    348.8928 ,   -134.4470, \
     22.9971 ,      3.5052 ,    -32.4942, \
      0.6601 ,      0.0078 ,      0.4887, \
   -108.4265 ,    -17.0117 ,    161.0185, \
     45.3412 ,      6.9765 ,    -65.4789, \
     11.3010 ,     40.6734 ,    -15.1929, \
     70.6497 ,    267.7602 ,   -102.9017, \
    139.6532 ,    536.9016 ,   -207.7388, \
    -30.0482 ,     -4.7850 ,     45.7010, \
    -30.0482 ,     -4.7850 ,     45.7010, \
    114.4365 ,    438.6881 ,   -169.4949, \
   -193.6120 ,    -30.2536 ,    286.4598, \
    -74.8135 ,    -11.7504 ,    111.6187, \
    -52.4484 ,     -8.2548 ,     78.6568, \
     28.9315 ,    -74.4789 ,    -41.0539
]
```