MWA beam software standalone version
(updated on 2018-05-28 , by Marcin Sokolowski)

- **<u>Requirements :</u>**
  - python - standard python can be used, but if does not work then possibly this can be useful:
    - anaconda - might be an easy way to have many packages installed:

      https://www.continuum.io/downloads#linux
    - Installation : bash ./Anaconda2-4.4.0-Linux-x86_64.sh
  - I needed to install these two extra packages on the clean environment :
    - pip install ephem
    - pip install pyfits

Generation of beam for a given fits files

- **Script : make_beam_test.py**
  - **<u>Description :</u>**

    Generates beam images (fits files) for a given sky image
  - **<u>Parameters :</u>**

    In order to correctly generate beam model the following parameters are required :

FITS file with sky image for which the beam model is generatd (option -f or --filename), frequency (options --freq_mhz or –freq_cc), beamformer delays (from metafits file, list of delays or MWA gridpoint) and time as observation id (option --obsid) or can be derrived from metafits file name (option --metafits).

- **-f or --filename FITS.fits** : fits file for which beam model is generated (in the same resolution)
- **-m or --metafits** : metafits file containg META data for this observation (beamformer delays are read from there in particular). **In order to correctly generate beam beamformer delays must be provided in any of the three possible ways (--metafits, --delays or --gridpoint)**.
- **--beamformer or -d or --delays** : beamformer delays provided in the command line parameter as a list , for example : -b 0,0,0,0,2,2,2,2,4,4,4,4,6,6,6,6
- **-g or --gridpoint** : provides MWA gridpoint number (also called sweet spot which also defines beamformer delays). List of MWA gridpoints can be found in files mwa_sweet_spots.py or MWA_sweet_spot_gridpoints.csv.
- **--model** : which beam model to use, 2014 – oldest analytic model, 2015 – average embedded element (called AEE or advanced) model and 2016 (Full

Embedded Element or FEE) model. Default is 2016 model (the newest and the most advanced).

- **--jones** : generates Jones matrices (not just power beam)
- **--freq_cc** – frequency coarse channel (multiply by 1.28 to get MHz).
- **--freq_mhz** – frequnecy in MHz
- **-o / --obsid** – which is GPS time of the observation start provides time of the observation
- Examples :
  - Using MWA metafits file which provides actual beamformer delays as used during the observation:

python make_beam_test.py -f 2456603.366393_img_200.335MHz_chan000of032_cal001_peel_000_000_t0000XX.fits --model=2016 --freq_mhz=203.51 -m 1067806072.metafits

The metafits file can be downloaded using command:

wget http://mwa-metadata01.pawsey.org.au/metadata/fits/?obs_id= 1067806072 -O 1067806072.metafits

- Using beamformer delays as parameters:

python make_beam_test.py -f wsclean_1067806072-0002-XX-dirty.fits -d 0,0,0,0,2,2,2,2,4,4,4,4,6,6,6,6 -v --model=2016

- Using metafits file :

  python make_beam_test.py -f wsclean_1067806072-0002-XX-dirty.fits --metafits 1067806072.metafits -v --model=2016

- Using gridpoint number, which defines MWA delays (as in mwa_sweet_spots.py) :

python make_beam_test.py -f wsclean_1067806072-0002-XX-dirty.fits -g 11 -v --model=2016

- Other examples :

python make_beam_test.py -f 2456603.366393_img_200.335MHz_chan000of032_cal001_peel_000_000_t0000XX.fits -g 11 -v --model=2016 --obsid=1067806072 --freq_mhz=203.52

python make_beam_test.py -f 2456603.366393_img_200.335MHz_chan000of032_cal001_peel_000_000_t0000XX.fits -d 0,0,0,0,2,2,2,2,4,4,4,4,6,6,6,6 -v --model=2016 --obsid=1067806072 --freq_mhz=203.51

Beam correction of images to generate Stokes images

- **Script beam_correct_image.py**
  - Description :

    Beam corrects instrumental images in (four expected XX,YY,XY and XYi – as coming out of WSCLEAN or RTS) and calculates Stokes images (I,Q,U,V)

  - Parameters :

    Similar parameters are required to correctly beam correct instrumental polarisation images and obtain Stokes images :

FITS file with sky image for which the beam model is generatd (option -f or --filename), frequency (options --freq_mhz or –freq_cc), beamformer delays (from metafits file, list of delays or MWA gridpoint) and time as observation id (option --obsid) or can be derrived from metafits file name (option --metafits).
WARNING : --metafits not yet implemented

- **--xx_file , --yy_file, --xy_file, --xyi_file –** provide names of FITS files with XX, YY, XY and XYi images.
- **--out_basename** – basename (prefix) for the output files names, then _I, _Q, -U and -V are added to output filenames
- **TO BE IMPLEMENTED : -m or --metafits** : metafits file containg META data for this observation (beamformer delays are read from there in particular). **In order to correctly generate beam beamformer delays must be provided in any of the three possible ways (--metafits, --delays or --gridpoint)**.
- **--beamformer or -b or --delays** : beamformer delays provided in the command line parameter as a list , for example : -b 0,0,0,0,2,2,2,2,4,4,4,4,6,6,6,6
- **-g or --gridpoint** : provides MWA gridpoint number (also called sweet spot which also defines beamformer delays). List of MWA gridpoints can be found in files mwa_sweet_spots.py or MWA_sweet_spot_gridpoints.csv.
- **--model** : which beam model to use, 2014 – oldest analytic model, 2015 – average embedded element (called AEE or advanced) model and 2016 (Full Embedded Element or FEE) model. Default is 2016 model (the newest and the most advanced).
- **--freq_mhz** – frequnecy in MHz
- **-o / --obsid** – which is GPS time of the observation start provides time of the observation
- **--h5file** – full path to HDF5 file with the full embedded element beam model (default is local file MWA_embedded_element_pattern_V02.h5)
- **--wsclean** – if XX,YY,XY,XYi images come from WSCLEAN software
- **--rts** – if images are from Real-Time System (RTS). WARNING : WSCLEAN and RTS software are known to follow different sign convention thus it is important to provide information on the software used to obtain correct sign values.

  - Using beamformer delays passed as parameter (-b 0,0,0,0,2,2,2,2,4,4,4,4,6,6,6,6) :

python ./beam_correct_image.py --xx_file=wsclean_1067806072-0002-XX-dirty.fits --yy_file=wsclean_1067806072-0002-YY-dirty.fits --xy_file=wsclean_1067806072-0002-XY-dirty.fits --xyi_file=wsclean_1067806072-0002-XYi-dirty.fits -b 0,0,0,0,2,2,2,2,4,4,4,4,6,6,6,6 --model=2016

- Using gridpoint number (-g 11) which defines the delays :

  python ./beam_correct_image.py --xx_file=wsclean_1067806072-0002-XX-dirty.fits --yy_file=wsclean_1067806072-0002-YY-dirty.fits --xy_file=wsclean_1067806072-0002-XY-dirty.fits --xyi_file=wsclean_1067806072-0002-XYi-dirty.fits -g 11 --model=2016

- Using analytic model :

  python ./beam_correct_image.py --xx_file=wsclean_1067806072-0002-XX-dirty.fits --yy_file=wsclean_1067806072-0002-YY-dirty.fits --xy_file=wsclean_1067806072-0002-XY-dirty.fits --xyi_file=wsclean_1067806072-0002-XYi-dirty.fits --model=analytic

Sensitivity calculations

- **Sensitivity calculation (will also be provided as a webservice) uses script mwa_sensitivity.py :**
  - Description :

    script calculates MWA sensitivity for specific observing parameters and time. It calcualtes SEFD and noise expected on images of a specified integration time (default 120 seconds) and bandwidth (default 1.28 MHz). The last lines of the standard output contain this information (“Noise expected on XX images”).

  - **On-line prototype version is being developed, but currently it is not available to external users.**
  - Parameters:

    Similar parameters are required to correctly beam correct instrumental polarisation images and obtain Stokes images :

FITS file with sky image for which the beam model is generatd (option -f or --filename), frequency (options --freq_mhz or –freq_cc), beamformer delays (from metafits file, list of delays or MWA gridpoint) and time as observation id (option --obsid) or can be derrived from metafits file name (option --metafits).

- **-m or --metafits** : metafits file containg META data for this observation (beamformer delays are read from there in particular). **In order to correctly generate beam beamformer delays must be provided in any of the three possible ways (--metafits, --delays or --gridpoint)**.
- **--beamformer or -b or --delays** : beamformer delays provided in the command line parameter as a list , for example : -b 0,0,0,0,2,2,2,2,4,4,4,4,6,6,6,6
- **--gridpoint** : provides MWA gridpoint number (also called sweet spot which also defines beamformer delays). List of MWA gridpoints can be found in files mwa_sweet_spots.py or MWA_sweet_spot_gridpoints.csv.
- **--model** : which beam model to use, 2014 – oldest analytic model, 2015 – average embedded element (called AEE or advanced) model and 2016 (Full Embedded Element or FEE) model. Default is 2016 model (the newest and the most advanced).
- **--jones** : generates Jones matrices (not just power beam)
- **-c or --channel or --freq_cc** – frequency coarse channel (multiply by 1.28 to get MHz).
- **--freq_mhz or -f or --frequency** – frequnecy in MHz
- **--gps / --obsid** – which is GPS time of the observation start provides time of the observation
- **--datetimestring** – UTC time of the observation
- **-D or --date** : UT date of the observation
- **-t or --time** : UT time of the observation

- **--plottype / -p** : plotting option. Type of plot: all, beam, sky, beamsky, beamsky_scaled. **It allows to generate beams for any pointing (without providing FITS file with sky image as make_beam_test.py does).**
- **--title** : plot title
- **--ext or -e** : plot file extention/format (default png)
- **--no_zenith_norm or -n** : turns off zenith normalization of the beam model (default)
- **--size** : resolution of the used beam model
- **--dir** : directory where generated beam model files are saved
- **--pointing_za_deg** – zenith angle where sensitivity is calculated [in decimal degrees] alternatively **--pointing_elev_deg** can be used to specify elevation [in decimal degrees]
- **--pointing_az_deg** – azimuth angle where sensitivity is calculated [in decimal degrees]
- **--t_rc**v : value of receiver temperature in Kelvin [default 0], overwrites option –trcv_type
- **--trcv_type** : which receiver temperature measurement is to be used [default is the one based on EDA drift scan and HASLAM map fit], can be overwritten by the option –t_rcv
- **-x / --outsens_file** : name of output sensitivity file
- **NOT YET IMPLEMENTED :** --add_sources : if add sources to the generated beam/sky image
- **-v or --verbose** : to generate more debugging output
-

▪ **<u>Examples :</u>**

- **<u>Test case obsid = 1190024128 with expected results :</u>**
  - wget http://mwa-metadata01.pawsey.org.au/metadata/fits/?obs_id=1190024128 -O 1190024128.metafits
  - python ./mwa_sensitivity.py -c 69,93,121,145,169 -p all --metafits=1190024128.metafits -m full_EE --pointing_az_deg=328.309328 --pointing_za_deg=36.246219 --outsens_file=test_HerA_db > output_db 2>&1
  - or with option --gridpoint=80
  - **<u>Expected results in test_HerA_db_XX.txt :</u>**

88.32000000 0.00564678 3681.76 20.79009433 0.00000000
119.04000000 0.01249623 1463.97 18.29416600 0.00000000
154.88000000 0.02358699 709.51 16.73515452 0.00000000
185.60000000 0.02923237 446.79 13.06087541 0.00000000
216.32000000 0.02398064 326.35 7.82617055 0.00000000

  - **<u>Expected noise on XX and YY images (in output_db file) :</u>**

Noise expected on XX images = 0.0728 Jy

Noise expected on YY images = 0.0656 Jy

- At zenith gridpoint (--gridpoint=0) and 5 degrees off zenith (--pointing_za_deg 5 --pointing_az_deg 0), frequency 185.6 MHz, integration time 120 sec, using 60 antennas (--antnum=60), also generates beam images (option -p all) at GPS time 1193567480 (-g 1193567480) :

  python ./mwa_sensitivity.py -f 185.6 --model 2016 --pointing_za_deg 5 --pointing_az_deg 0 --antnum=60 --inttime=120 -p all --gridpoint=0 -g 1193567480

C/C++ implementations

- **C/C++ implementations of the 2016 MWA beam model** :
  - Real-time System (RTS) C implementation, for details please contact Bart Pindor (not sure if this is in the main version or a test branch in git)
  - Andre's (C++ version). In calibrate and beam (option -2016) can also use the 2016 model if it is enabled on compilation level or via parameters.
-