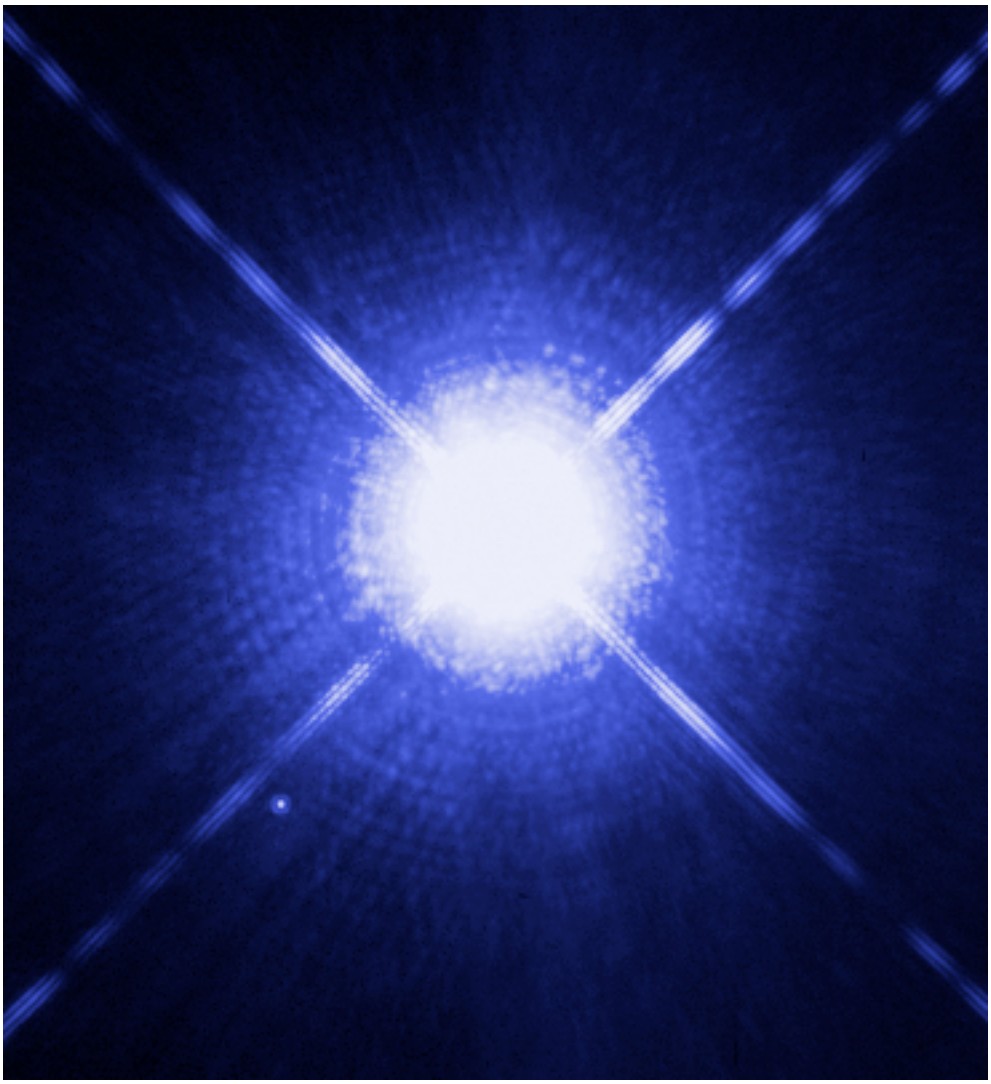


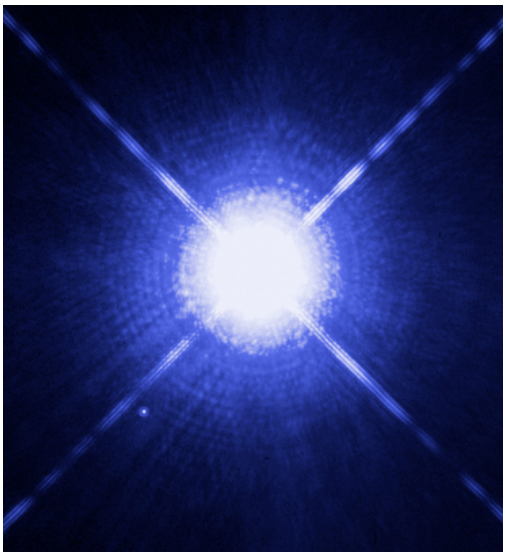
Simulation of Radio Interferometry from Unique Sources Documentation

Release 0.1b

National Radio Astronomy Observatory

Mar 16, 2022

CONTENTS



CHAPTER

ONE

INTRODUCTION: SIRIUS (SIMULATION OF RADIO INTERFEROMETRY FROM UNIQUE SOURCES)

SiRIUS is a component list radio interferometry visibilities simulator **in development** for the VLA, ALMA, and the ngVLA telescopes. It makes use of modular Common Astronomy Software Applications (CASA), the CASA Next Generation Infrastructure framework (CNGI), and dask-ms (dask-ms).

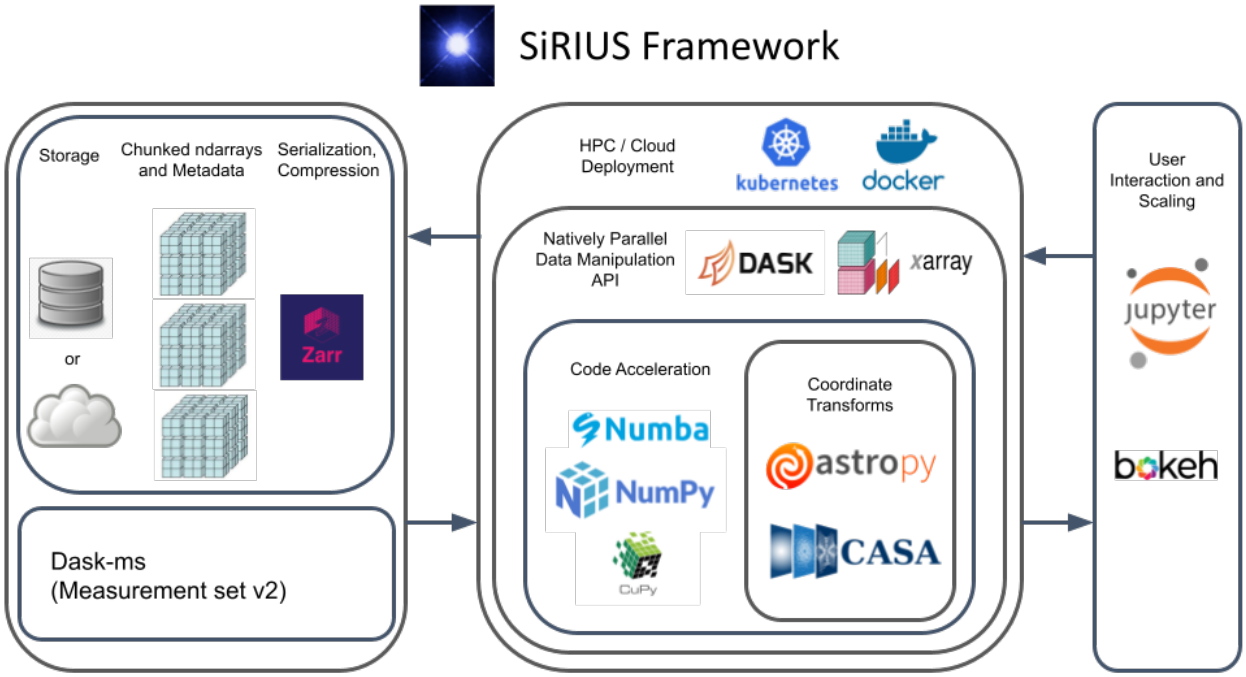
1.1 Goals

- Create a production-ready simulation package using the CNGI framework that integrates seamlessly into the modular CASA Python environment to demonstrate the framework's suitability for developing ngCASA.
- Demonstrate performance and scalability by simulating large ngVLA size datasets on a cluster computer.
- Exceed the CASA component list simulator functionality of the simulator tool, simobserve task, and simalma task.
- Validate full Stokes imaging of the mosaic, awproject, and hpg gridders in the CASA task tclean using SiRIUS simulated datasets.

1.2 Framework

SiRIUS makes use of a modified version of the functional programming paradigm in the CNGI prototype were a directed acyclic graph (DAG) of blocked algorithms composed from functions (the edges) and data (the vertices) for lazy evaluation by a scheduler process coordinating a network of (optionally distributed and heterogeneous) machine resources. SiRIUS expands on the CNGI prototype framework by incorporating dask-ms which enables reading and writing CASA tables, via python-casacore, with an xarray Dataset interface and Dask lazy execution. The addition of dask-ms allows SiRIUS to be compatible with CASA because the visibility data can now be stored in a Measurement Set.

The relationship between the libraries used in SiRIUS can be conceptualized by the following diagram:



In the framework, data is stored in either the Measurement Set v2 or the Zarr format. The Zarr format allows files to be mounted on a local disk or in the cloud, while the Measurement Set data can only be mounted on a local disk. The Dask scheduler manages N worker processes identified to the central scheduler by their unique network address and coordinating M threads. Each thread applies functions to a set of data chunks. Xarray wraps Dask functions and labels the Dask arrays for ease of interface. Any code that is wrapped with Python can be parallelized with Dask. Therefore, the option to use C++, Numba, Cupy, or another custom high-performance computing (HPC) code is retained. The CASA Measures tool and Astropy are used for coordinate conversions. Cloud deployment is enbaled using Docker and Kubernetes. User can interface with the SiRIUS using Jupyter notebooks (see the basic simulation example).

1.3 Requirements

SiRIUS should work on Mac OS and Linux using Python=3.8.

1.4 Installation

Currently, the python-casacore dependency fails to `pip install` for Mac users, consequently `conda install` must be used.

```
conda create -n sirius python=3.8
conda activate sirius
conda install -c conda-forge python-casacore
pip install sirius
```

1.5 Development Installation

```
conda create -n sirius python=3.8
conda activate sirius
conda install -c conda-forge python-casacore
git clone https://github.com/casangi/sirius.git
cd sirius
pip install -e .
```

1.6 Contact

Questions and feedback can be sent to: jsteeb@nrao.edu

Sirius A and B image by NASA, ESA, H. Bond (STScI), and M. Barstow (University of Leicester).

CHAPTER

TWO

API

Under development.

2.1 sirius.simulation

simulation(*point_source_flux*, *point_source_ra_dec*, *pointing_ra_dec*, *phase_center_ra_dec*, *phase_center_names*, *beam_parms*, *beam_models*, *beam_model_map*, *uvw_parms*, *tel_xds*, *time_xda*, *chan_xda*, *pol*, *noise_parms*, *save_parms*)

Creates a dask graph that computes a simulated measurement set and triggers a compute and saves the ms to disk.

Parameters

- **point_source_flux** (*float np.array, [n_point_sources,n_time, n_chan, n_pol], (singleton: n_time, n_chan), Janskys*) – The flux of the point sources.
- **point_source_ra_dec** (*float np.array, [n_time, n_point_sources, 2], (singleton: n_time), radians*) – The position of the point sources.
- **pointing_ra_dec** (*float np.array, [n_time, n_ant, 2], (singleton: n_time, n_ant), radians*) – Pointings of antennas, if they are different from the phase center. Set to None if no pointing offsets are required.
- **phase_center_ra_dec** (*float np.array, [n_time, 2], (singleton: n_time), radians*) – Phase center of array.
- **phase_center_names** (*str np.array, [n_time], (singleton: n_time)*) – Strings that are used to label phase centers.
- **beam_parms** (*dict*) –
- **beam_parms['mueller_selection']** (*int np.array, default=np.array([0, 5, 10, 15])*) – The elements in the 4x4 beam Mueller matrix to use. The elements are numbered row wise. For example [0, 5, 10, 15] are the diagonal elements.
- **beam_parms['pa_radius']** (*float, default=0.2, radians*) – The change in parallactic angle that will trigger the calculation of a new beam when using Zernike polynomial aperture models.
- **beam_parms['image_size']** (*int np.array, default=np.array([1000,1000])*) – Size of the beam image generated from the Zernike polynomial coefficients.

- **beam_parms['fov_scaling']** (*int, default=15*) – Used to scale the size of the beam image which is given fov_scaling*(1.22 *c/(dish_diam*frequency)).
- **beam_parms['zernike_freq_interp']** (*str, default='nearest', options=['linear', 'nearest', 'zero', 'slinear', 'quadratic', 'cubic']*) – What interpolation method to use for Zernike polynomial coefficients.
- **beam_models** (*list*) – List of beam models to use. Beam models can be any combination of function parameter dictionaries, image xr.Datasets or Zernike polynomial coefficient xr.Datasets.
- **beam_model_map** (*int np.array, [n_ant]*) – Each element in beam_model_map is an index into beam_models.
- **uvw_parms** (*dict*) –
- **uvw_parms['calc_method']** (*str, default='astropy', options=['astropy','casa']*) – Astropy coordinates or CASA tool measures can be used to calculate uvw coordinates.
- **uvw_parms['auto_corr']** (*bool, default=False*) – If True autocorrelations are also calculated.
- **tel_xds** (*xr.Dataset*) – An xarray dataset of the radio telescope array layout (see zarr files in sirius_data/telescope_layout/data/ for examples).
- **time_xda** (*xr.DataArray*) – Time series xarray array.
- **chan_xda** (*xr.DataArray*) – Channel frequencies xarray array.
- **pol** (*int np.array*) – Must be a subset of ['RR','RL','LR','LL'] => [5,6,7,8] or ['XX','XY','YX','YY'] => [9,10,11,12].
- **noise_parms** (*dict*) – Set various system parameters from which the thermal (ie, random additive) noise level will be calculated. See https://casadocs.readthedocs.io/en/stable/api/tt/casatools.simulator.html#casatools.simulator.simulator.setnoise.
- **noise_parms['mode']** (*str, default='tsys-manuel', options=['simplenoise','tsys-manuel','tsys-atm']*) – Currently only 'tsys-manuel' is implemented.
- **noise_parms['t_atmos']** (*, float, default = 250.0, Kelvin*) – Temperature of atmosphere (mode='tsys-manual')
- **noise_parms['tau']** (*float, default = 0.1*) – Zenith Atmospheric Opacity (if tsys-manual). Currently the effect of Zenith Atmospheric Opacity (Tau) is not included in the noise modeling.
- **noise_parms['ant_efficiency']** (*float, default=0.8*) – Antenna efficiency
- **noise_parms['spill_efficiency']** (*float, default=0.85*) – Forward spillover efficiency.
- **noise_parms['corr_efficiency']** (*float, default=0.88*) – Correlation efficiency.
- **noise_parms['t_receiver']** (*float, default=50.0, Kelvin*) – Receiver temp (ie, all non-atmospheric Tsys contributions).
- **noise_parms['t_ground']** (*float, default=270.0, Kelvin*) – Temperature of ground/spill.

- **noise_parms['t_cmb']** (*float, default=2.725, Kelvin*) – Cosmic microwave background temperature.
- **save_parms** (*dict*) –
- **save_parms['mode']** (*str, default='dask_ms_and_sim_tool', options=['lazy','zarr','dask_ms_and_sim_tool', 'zarr_convert_ms','dask_ms','cngi']*) –
- **save_parms['DAG_name_vis_uvw_gen']** (*str, default=False*) – Creates a DAG diagram png, named save_parms['DAG_name_write'], of how the visibilities and uvw coordinates are calculated.
- **save_parms['DAG_name_write']** (*str, default=False*) – Creates a DAG diagram png, named save_parms['DAG_name_write'], of how the ms is created with name.
- **save_parms['ms_name']** (*str, default='sirius_sim.ms'*) – If save_parms['mode']='zarr' the name sirius_sim.vis.zarr will be used.

Returns ms_xds

Return type xr.Dataset

simulation_chunk(*point_source_flux*, *point_source_ra_dec*, *pointing_ra_dec*, *phase_center_ra_dec*, *beam_parms*, *beam_models*, *beam_model_map*, *uvw_parms*, *tel_xds*, *time_chunk*, *chan_chunk*, *pol*, *noise_parms*, *uvw_precompute=None*)

Simulates uvw coordinates, interferometric visibilities and adds noise. This function does not produce a measurement set.

Parameters

- **point_source_flux** (*float np.array, [n_point_sources,n_time, n_chan, n_pol], (singleton: n_time, n_chan), Janskys*) – The flux of the point sources.
- **point_source_ra_dec** (*float np.array, [n_time, n_point_sources, 2], (singleton: n_time), radians*) – The position of the point sources.
- **pointing_ra_dec** (*float np.array, [n_time, n_ant, 2], (singleton: n_time, n_ant), radians*) – Pointings of antennas, if they are different from the phase center. Set to None if no pointing offsets are required.
- **phase_center_ra_dec** (*float np.array, [n_time, 2], (singleton: n_time), radians*) – Phase center of array.
- **beam_parms** (*dict*) –
- **beam_parms['mueller_selection']** (*int np.array, default=np.array([0, 5, 10, 15])*) – The elements in the 4x4 beam Mueller matrix to use. The elements are numbered row wise. For example [0, 5, 10, 15] are the diagonal elements.
- **beam_parms['pa_radius']** (*float, default=0.2, radians*) – The change in parallactic angle that will trigger the calculation of a new beam when using Zernike polynomial aperture models.
- **beam_parms['image_size']** (*int np.array, default=np.array([1000,1000])*) – Size of the beam image generated from the Zernike polynomial coefficients.
- **beam_parms['fov_scaling']** (*int, default=15*) – Used to scale the size of the beam image, which is given by fov_scaling*(1.22 *c/(dish_diam*frequency)).

- **beam_parms['zernike_freq_interp']** (*str, default='nearest', options=['linear', 'nearest', 'zero', 'slinear', 'quadratic', 'cubic']*) – What interpolation method to use for Zernike polynomial coefficients.
- **beam_models** (*list*) – List of beam models to use. Beam models can be any combination of function parameter dictionaries, image xr.Datasets or Zernike polynomial coefficient xr.Datasets.
- **beam_model_map** (*int np.array, [n_ant]*) – Each element in beam_model_map is an index into beam_models.
- **uvw_parms** (*dict*) –
- **uvw_parms['calc_method']** (*str, default='astropy', options=['astropy','casa']*) – Astropy coordinates or CASA tool measures can be used to calculate uvw coordinates.
- **uvw_parms['auto_corr']** (*bool, default=False*) – If True autocorrelations are also calculated.
- **tel_xds** (*xr.Dataset*) – An xarray dataset of the radio telescope array layout (see zarr files in sirius_data/telescope_layout/data/ for examples).
- **time_chunk** (*str np.array, [n_time], 'YYYY-MM-DDTHH:MM:SS.SSS'*) – Time series. Example '2019-10-03T19:00:00.000'.
- **chan_chunk** (*float np.array, [n_chan], Hz*) – Channel frequencies.
- **pol** (*int np.array*) – Must be a subset of ['RR','RL','LR','LL'] => [5,6,7,8] or ['XX','XY','YX','YY'] => [9,10,11,12].
- **noise_parms** (*dict*) – Set various system parameters from which the thermal (ie, random additive) noise level will be calculated. See https://casadocs.readthedocs.io/en/stable/api/tt/casatools.simulator.html#casatools.simulator.simulator.setnoise.
- **noise_parms['mode']** (*str, default='tsys-manuel', options=['simplenoise','tsys-manuel','tsys-atm']*) – Currently only 'tsys-manuel' is implemented.
- **noise_parms['t_atmos']** (*, float, default = 250.0, Kelvin*) – Temperature of atmosphere (mode='tsys-manual')
- **noise_parms['tau']** (*float, default = 0.1*) – Zenith Atmospheric Opacity (if tsys-manual). Currently the effect of Zenith Atmospheric Opacity (Tau) is not included in the noise modeling.
- **noise_parms['ant_efficiency']** (*float, default=0.8*) – Antenna efficiency
- **noise_parms['spill_efficiency']** (*float, default=0.85*) – Forward spillover efficiency.
- **noise_parms['corr_efficiency']** (*float, default=0.88*) – Correlation efficiency.
- **noise_parms['t_receiver']** (*float, default=50.0, Kelvin*) – Receiver temp (ie, all non-atmospheric Tsys contributions).
- **noise_parms['t_ground']** (*float, default=270.0, Kelvin*) – Temperature of ground/spill.

- **noise_parms['t_cmb']** (*float, default=2.725, Kelvin*) – Cosmic microwave background temperature.

Returns

- **vis** (*complex np.array, [n_time,n_baseline,n_chan,n_pol]*) – Visibility data.
- **uvw** (*float np.array, [n_time,n_baseline,3]*) – Spatial frequency coordinates.
- **weight** (*complex np.array, [n_time,n_baseline,n_pol]*) – Data weights.
- **sigma** (*complex np.array, [n_time,n_baseline,n_pol]*) – RMS noise of data.
- **t_arr** (*float np.array, [4]*) – Timing infromation: calculate uvw, evaluate_beam_models, calculate visibilities, calculate additive noise.

2.2 sirius.calc_uvw

calc_uvw_chunk(*tel_xds*, *time_str*, *phase_center_ra_dec*, *uvw_parms*, *check_parms=True*)

Calculates a chunk of uvw values. This function forms part of a node in a Dask graph. This function can be used on its own if no parallelism is required.

Parameters

- **tel_xds** (*xr.Dataset*) – An xarray dataset of the radio telescope array layout (see zarr files in sirius_data/telescope_layout/data/ for examples).
- **time_str** (*str np.array, [n_time], 'YYYY-MM-DDTHH:MM:SS.SSS'*) – Time serie. Example '2019-10-03T19:00:00.000'.
- **phase_center_ra_dec** (*float np.array, [n_time, 2], (singleton: n_time), radians*) – Phase center of array.
- **uvw_parms** (*dict*) –
- **uvw_parms['calc_method']** (*str, default='astropy', options=['astropy','casa']*) – The uvw coordinates can be calculated using the astropy package or the casa measures tool.
- **uvw_parms['auto_corr']** (*bool, default=False*) – If True autocorrelations are also calculated.
- **check_parms** (*bool*) – Check input parameters and asign defaults.

Returns

- **uvw** (*float np.array, [n_time,n_baseline,3]*) – The uvw coorsdinates in per wavelength units.
- **antenna1** (*int np.array, [n_baseline]*) – The indices of the first antenna in the baseline pairs.
- **antenna2** (*int np.array, [n_baseline]*) – The indices of the second antenna in the baseline pairs.

2.3 sirius.calc_beam

calc_zpc_beam(*zpc_xds*, *parallactic_angles*, *freq_chan*, *beam_parms*, *check_parms=True*)

Calculates an antenna apertures from Zernike polynomial coefficients, and then Fourier transform it to obtain the antenna beam image. The beam image dimensionality is [pa (paralactic angle), chan (channel), pol (polarization), l (orthographic/synthesis projection of directional cosine), m (orthographic/synthesis projection of directional cosine)].

Parameters

- **zpc_xds** (*xr.Dataset*) – A Zernike polynomial coefficient xr.Datasets. Available models can be found in sirius_data/zernike_dish_models/data.
- **parallactic_angles** (*float np.array, [n_pa], radians*) – An array of the parallactic angles for which to calculate the antenna beams.
- **freq_chan** (*float np.array, [n_chan], Hz*) – Channel frequencies.
- **beam_parms** (*dict*) –
- **beam_parms['mueller_selection']** (*int np.array, default=np.array([0, 5, 10, 15])*) – The elements in the 4x4 beam Mueller matrix to use. The elements are numbered row wise. For example [0, 5, 10, 15] are the diagonal elements.
- **beam_parms['pa_radius']** (*float, default=0.2, radians*) – The change in parallactic angle that will trigger the calculation of a new beam when using Zernike polynomial aperture models.
- **beam_parms['image_size']** (*int np.array, default=np.array([1000,1000])*) – This parameter should rarely be modified. Size of the beam image generated from the Zernike polynomial coefficients.
- **beam_parms['fov_scaling']** (*int, default=1.2*) – This parameter should rarely be modified. Used to determine the cell size of the beam image so that it lies within the image that is generated.
- **beam_parms['zernike_freq_interp']** (*str, default='nearest', options=['linear', 'nearest', 'zero', 'slinear', 'quadratic', 'cubic']*) – What interpolation method to use for Zernike polynomial coefficients.
- **check_parms** (*bool*) – Check input parameters and asign defaults.

Returns **J_xds** – An xds that contains the image of the per antenna beam as a function of [pa (paralactic angle), chan (channel), pol (polarization), l (orthographic/synthesis projection of directional cosine), m (orthographic/synthesis projection of directional cosine)]. Should not be confused with the primary beam, which is the beam for a baseline and is equal to the product of two antenna beams.

Return type xr.Dataset

evaluate_beam_models(*beam_models*, *time_str*, *freq_chan*, *phase_center_ra_dec*, *site_location*, *beam_parms*, *check_parms=True*)

Loops over beam_models and converts each Zernike polynomial coefficient xr.Datasets to an antenna beam image. The beam image dimensionality is [pa (paralactic angle), chan (channel), pol (polarization), l (orthographic/synthesis projection of directional cosine), m (orthographic/synthesis projection of directional cosine)]. The parallactic angles are also calculated for each date-time in time_str at the site_location and with a right ascension declination in phase_center_ra_dec. A subset of parallactic angles are used in the pa coordinate of the beam image, where all pa values are within beam_parms['pa_radius'] radians.

Parameters

- **beam_models** (*list*) – List of beam models to use. Beam models can be any combination of function parameter dictionaries, image xr.Datasets or Zernike polynomial coefficient xr.Datasets (models can be found in sirius_data/zernike_dish_models/data).
- **time_str** (*str np.array, [n_time], 'YYYY-MM-DDTHH:MM:SS.SSS'*) – Time series. Example '2019-10-03T19:00:00.000'.
- **freq_chan** (*float np.array, [n_chan], Hz*) – Channel frequencies.
- **phase_center_ra_dec** (*float np.array, [n_time, 2], (singleton: n_time), radians*) – Phase center of array.
- **site_location** (*dict*) – A dictionary with the location of telescope. For example [{'m0': {'unit': 'm', 'value': -1601185}, 'm1': {'unit': 'm', 'value': -5041977}, 'm2': {'unit': 'm', 'value': 3554875}, 'refer': 'ITRF', 'type': 'position'}]. The site location of telescopes can be found in site_pos attribute of the xarray dataset of the radio telescope array layout (see zarr files in sirius_data/telescope_layout/data/).
- **parallactic_angles** (*float np.array, radians*) – An array of the parallactic angles for which to calculate the antenna beams.
- **freq_chan** – Channel frequencies.
- **beam_parms** (*dict*) –
- **beam_parms['mueller_selection']** (*int np.array, default=np.array([0, 5, 10, 15])*) – The elements in the 4x4 beam Mueller matrix to use. The elements are numbered row wise. For example [0, 5, 10, 15] are the diagonal elements.
- **beam_parms['pa_radius']** (*float, default=0.2, radians*) – The change in parallactic angle that will trigger the calculation of a new beam when using Zernike polynomial aperture models.
- **beam_parms['image_size']** (*int np.array, default=np.array([1000,1000])*) – Size of the beam image generated from the Zernike polynomial coefficients.
- **beam_parms['fov_scaling']** (*int, default=1.2*) – This parameter should rarely be modified. Used to determine the cell size of the beam image so that it lies within the image that is generated.
- **beam_parms['zernike_freq_interp']** (*str, default='nearest', options=['linear', 'nearest', 'zero', 'slinear', 'quadratic', 'cubic']*) – What interpolation method to use for Zernike polynomial coefficients.

Returns **J_xds** – An xds that contains the image of the per antenna beam as a function of [pa (paralactic angle), chan (channel), pol (polarization), l (orthographic/synthesis projection of directional cosine), m (orthographic/synthesis projection of directional cosine)]. Should not be confused with the primary beam, which is the beam for a baseline and is equal to the product of two antenna beams.

Return type xr.Dataset

2.4 sirius.calc_vis

calc_vis_chunk(*uvw*, *vis_data_shape*, *point_source_flux*, *point_source_ra_dec*, *pointing_ra_dec*, *phase_center_ra_dec*, *antenna1*, *antenna2*, *chan_chunk*, *beam_model_map*, *beam_models*, *parallactic_angle*, *pol*, *mueller_selection*, *check_parms=True*)

Simulate interferometric visibilities.

Parameters

- **uvw** (*float np.array, [n_time,n_baseline,3], meter*) – Spatial frequency coordinates.
- **vis_data_shape** (*float np.array, [4]*) – Dimensions of visibility data [n_time, n_baseline, n_chan, n_pol]
- **point_source_flux** (*float np.array, [n_point_sources,n_time, n_chan, n_pol], (singleton: n_time, n_chan), Janskys*) – The flux of the point sources.
- **point_source_ra_dec** (*float np.array, [n_time, n_point_sources, 2], (singleton: n_time), radians*) – The position of the point sources.
- **pointing_ra_dec** (*float np.array, [n_time, n_ant, 2], (singleton: n_time, n_ant), radians*) – Pointings of antennas, if they are different from the phase center. Set to None if no pointing offsets are required.
- **phase_center_ra_dec** (*float np.array, [n_time, 2], (singleton: n_time), radians*) – Phase center of array.
- **antenna1** (*np.array of int, [n_baseline]*) – The indices of the first antenna in the baseline pairs. The _calc_baseline_indx_pair function in sirius._sirius_utils._array_utils can be used to calculate these values.
- **antenna2** (*np.array of int, [n_baseline]*) – The indices of the second antenna in the baseline pairs. The _calc_baseline_indx_pair function in sirius._sirius_utils._array_utils can be used to calculate these values.
- **chan_chunk** (*float np.array, [n_chan], Hz*) – Channel frequencies.
- **beam_model_map** (*int np.array, [n_ant]*) – Each element in beam_model_map is an index into beam_models.
- **beam_models** (*list*) – List of beam models to use. Beam models can be any combination of function parameter dictionaries or image xr.Datasets. Any Zernike polynomial coefficient xr.Datasets models must be converted to images using sirius.calc_beam.evaluate_beam_models.
- **parallactic_angle** (*float np.array, [n_time], radians*) – Parallactic angle over time at the array center.
- **pol** (*int np.array*) – Must be a subset of ['RR','RL','LR','LL'] => [5,6,7,8] or ['XX','XY','YX','YY'] => [9,10,11,12].
- **mueller_selection** (*int np.array*) – The elements in the 4x4 beam Mueller matrix to use. The elements are numbered row wise. For example [0, 5, 10, 15] are the diagonal elements.
- **check_parms** (*bool*) – Check input parameters and asign defaults.

Returns **vis** – Visibility data.

Return type complex np.array, [n_time,n_baseline,n_chan,n_pol]

2.5 sirius.calc_a_noise

calc_a_noise_chunk(*vis_shape*, *uvw*, *beam_model_map*, *beam_models*, *antenna1*, *antenna2*, *noise_parms*, *check_parms=True*)

Add noise to visibilities.

Parameters

- **vis_data_shape** (*float np.array, [4]*) – Dimensions of visibility data [n_time, n_baseline, n_chan, n_pol].
- **uvw** (*float np.array, [n_time,n_baseline,3]*) – Spatial frequency coordinates. Can be None if no autocorrelations are present.
- **beam_model_map** (*int np.array, [n_ant]*) – Each element in beam_model_map is an index into beam_models.
- **beam_models** (*list*) – List of beam models to use. Beam models can be any combination of function parameter dictionaries, image xr.Datasets or Zernike polynomial coefficient xr.Datasets.
- **antenna1** (*np.array of int, [n_baseline]*) – The indices of the first antenna in the baseline pairs. The _calc_baseline_indx_pair function in sirius._sirius_utils._array_utils can be used to calculate these values.
- **antenna2** (*np.array of int, [n_baseline]*) – The indices of the second antenna in the baseline pairs. The _calc_baseline_indx_pair function in sirius._sirius_utils._array_utils can be used to calculate these values.
- **noise_parms** (*dict*) – Set various system parameters from which the thermal (ie, random additive) noise level will be calculated. See https://casadocs.readthedocs.io/en/stable/api/tt/casatools.simulator.html#casatools.simulator.simulator.setnoise.
- **noise_parms['mode']** (*str, default='tsys-manual', options=['simplenoise','tsys-manual','tsys-atm']*) – Currently only 'tsys-manual' is implemented.
- **noise_parms['t_atmos']** (*, float, default = 250.0, Kelvin*) – Temperature of atmosphere (mode='tsys-manual')
- **noise_parms['tau']** (*float, default = 0.1*) – Zenith Atmospheric Opacity (if tsys-manual). Currently the effect of Zenith Atmospheric Opacity (Tau) is not included in the noise modeling.
- **noise_parms['ant_efficiency']** (*float, default=0.8*) – Antenna efficiency.
- **noise_parms['spill_efficiency']** (*float, default=0.85*) – Forward spillover efficiency.
- **noise_parms['corr_efficiency']** (*float, default=0.88*) – Correlation efficiency.
- **noise_parms['t_receiver']** (*float, default=50.0, Kelvin*) – Receiver temp (ie, all non-atmospheric Tsys contributions).
- **noise_parms['t_ground']** (*float, default=270.0, Kelvin*) – Temperature of ground/spill.

- **noise_parms['t_cmb']** (*float, default=2.725, Kelvin*) – Cosmic microwave background temperature.
- **noise_parms['auto_corr']** (*bool, default=False*) – If True autocorrelations are also calculated.
- **noise_parms['freq_resolution']** (*float, Hz*) – Width of a single channel.
- **noise_parms['time_delta']** (*float, s*) – Integration time.
- **check_parms** (*bool*) – Check input parameters and asign defaults.

Returns

- **noise** (*complex np.array, [n_time, n_baseline, n_chan, n_pol]*)
- **weight** (*float np.array, [n_time, n_baseline, n_pol]*)
- **sigma** (*float np.array, [n_time, n_baseline, n_pol]*)

CHAPTER

THREE

DEVELOPMENT

[3]:
```
import os
try:
    import sirius
    print('SiRIUS version',sirius.__version__,'already installed.')
except ImportError as e:
    print(e)
    print('Installing SiRIUS')
    os.system("pip install sirius")
    import sirius
    print('SiRIUS version',sirius.__version__,' installed.')
```

```
SiRIUS version 0.0.19 already installed.
```

[4]:
```
import pkg_resources
import xarray as xr
import numpy as np
from astropy.coordinates import SkyCoord
xr.set_options(display_style="html")
import os
try:
    from google.colab import output
    output.enable_custom_widget_manager()
    IN_COLAB = True
except:
    IN_COLAB = False
%matplotlib widget
```

3.1 Organization

3.2 Architecture

3.3 Data Structures

3.3.1 tel.zarr

[7]:
```
########## Telescope layout ##########
tel_dir = pkg_resources.resource_filename('sirius_data', 'telescope_layout/data/vla.d.
↪tel.zarr')
```

(continues on next page)

(continued from previous page)

```
tel_xds = xr.open_zarr(tel_dir,consolidated=False)
tel_xds
```

[7]:
```
<xarray.Dataset>
Dimensions:         (ant_name: 27, pos_coord: 3)
Coordinates:
  * ant_name        (ant_name) <U3 'W01' 'W02' 'W03' 'W04' ... 'N07' 'N08' 'N09'
  * pos_coord       (pos_coord) int64 0 1 2
Data variables:
    ANT_POS         (ant_name, pos_coord) float64 dask.array<chunksize=(27, 3),␣
↪meta=np.ndarray>
    DISH_DIAMETER   (ant_name) float64 dask.array<chunksize=(27,), meta=np.ndarray>
Attributes:
    site_pos:         [{'m0': {'unit': 'm', 'value': -1601185.3650000016}, 'm1...
    telescope_name:   VLA
```

3.3.2 zpc.zarr

[8]:
```
########## Zernike Polynomial Dish Aperture Models ##########
zpc_dir = pkg_resources.resource_filename('sirius_data', 'zernike_dish_models/data/
↪EVLA_avg_zcoeffs_SBand_lookup.zpc.zarr')
zpc_xds = xr.open_zarr(zpc_dir,consolidated=False)
zpc_xds
```

[8]:
```
<xarray.Dataset>
Dimensions:     (pol: 4, chan: 16, coef_indx: 66)
Coordinates:
  * chan        (chan) float64 2.052e+09 2.18e+09 ... 3.844e+09 3.972e+09
  * coef_indx   (coef_indx) int64 0 1 2 3 4 5 6 7 8 ... 58 59 60 61 62 63 64 65
  * pol         (pol) int64 5 6 7 8
Data variables:
    ETA         (pol, chan, coef_indx) float64 dask.array<chunksize=(4, 16, 66),␣
↪meta=np.ndarray>
    ZC          (pol, chan, coef_indx) complex128 dask.array<chunksize=(4, 16, 66),␣
↪meta=np.ndarray>
Attributes:
    conversion_date:  2022-01-03
    dish_diam:        25
    max_rad_1GHz:     0.014946999714079439
    telescope_name:   EVLA
    zpc_file_name:    EVLA_avg_zcoeffs_SBand_lookup.csv
```

3.3.3 bim.zarr

CHAPTER

FOUR

MODELING ANTENNAS

Under Development

Too see plots run in google colab: https://colab.research.google.com/github/casangi/sirius/blob/main/docs/antenna_beams.ipynb

4.1 Load Packages

[1]:
```
import os
try:
    import sirius
    print('SiRIUS version',sirius.__version__,'already installed.')
except ImportError as e:
    print(e)
    print('Installing SiRIUS')
    os.system("pip install sirius")
    import sirius
    print('SiRIUS version',sirius.__version__,' installed.')

import pkg_resources
import xarray as xr
import numpy as np
from ipywidgets import interactive, fixed
from sirius.display_tools import display_J, display_M
from sirius.calc_beam import make_mueler_mat

from astropy.coordinates import SkyCoord
xr.set_options(display_style="html")
import os
try:
    from google.colab import output
    output.enable_custom_widget_manager()
    IN_COLAB = True
except:
    IN_COLAB = False
%matplotlib widget
#%matplotlib inline
```

```
SiRIUS version 0.0.22 already installed.
```

4.2 Voltage Pattern and Primary Beam

The Voltage Pattern (VP) is the image plane response of a single antenna and the Primary Beam (PB) is the image plane response of pair of antennas (baselines) and is given by:

$$PB = VP \odot VP^*$$

where * is the complex conjugate and $\odot$ is the Hadamard product (element wise product).

^{T} transpose. ^{H} Hermitian or complex conjugate transpose.

Aperture illumination function (AIP)

[2]:
```
from sirius.calc_beam import calc_beam, calc_zpc_beam
import numpy as np
from sirius_data.beam_1d_func_models.airy_disk import aca, alma, vla
print(vla)

#freq_chan = np.array([60*10**9,90*10**9,100*10**9,120*10**9])
freq_chan = np.array([90*10**9])
#freq_chan = np.array([3.052*10**9])
beam_parms = {}
J_xds = calc_beam(alma,freq_chan,beam_parms,check_parms=True)
#J_xds = calc_beam(vla,freq_chan,beam_parms,check_parms=True)
J_xds
```

```
{'func': 'casa_airy', 'dish_diam': 24.5, 'blockage_diam': 0.0, 'max_rad_1GHz': 0.
↪014946999714079439}
Setting default fov_scaling  to  4.0
Setting default mueller_selection  to  [ 0  5 10 15]
Setting default zernike_freq_interp  to  nearest
Setting default pa_radius  to  0.2
Setting default image_size  to  [1000 1000]
[1000 1000] [-1.38385217e-06  1.38385217e-06] 4.0 0.03113667385557884
```

[2]:
```
<xarray.Dataset>
Dimensions:  (chan: 1, pa: 1, pol: 1, l: 1000, m: 1000)
Coordinates:
  * chan     (chan) int64 90000000000
  * pa       (pa) int64 0
  * pol      (pol) int64 0
  * l        (l) float64 0.0006919 0.0006905 0.0006892 ... -0.0006892 -0.0006905
  * m        (m) float64 -0.0006919 -0.0006905 ... 0.0006892 0.0006905
Data variables:
    J        (pa, chan, pol, l, m) float64 0.01527 0.01577 ... 0.0168 0.0163
```

[3]:
```
from sirius.display_tools import display_J

interactive_plot_1 = interactive(display_J, J_xds=fixed(J_xds), pa=fixed(0), chan=(0,
↪J_xds.dims['chan']-1), val_type=['abs','phase','real','imag'],units=['rad','arcsec',
↪'arcmin','deg'])
output_1 = interactive_plot_1.children[-1]
output_1.layout.auto_scroll_threshold = 9999;
interactive_plot_1
```

```
interactive(children=(IntSlider(value=0, description='chan', max=0),␣
↪Dropdown(description='val_type', options=...
```

4.3 EVLA Polynomial Model

[4]:
```
%load_ext autoreload
%autoreload 2
```

[5]:
```
pc_dir = pkg_resources.resource_filename('sirius_data', 'beam_polynomial_coefficient_
↪models/data/EVLA_.bpc.zarr')
pc_xds = xr.open_zarr(pc_dir,consolidated=False)

pc_xds = pc_xds.where(pc_xds.band=='S',drop=True)
#pc_xds = pc_xds.isel(chan=[36,37])
print(pc_xds.band.values)
print(pc_xds.chan.values)
pc_xds
```

```
['S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S' 'S']
[2.052e+09 2.180e+09 2.436e+09 2.564e+09 2.692e+09 2.820e+09 2.948e+09
 3.052e+09 3.180e+09 3.308e+09 3.436e+09 3.564e+09 3.692e+09 3.820e+09
 3.948e+09]
```

[5]:
```
<xarray.Dataset>
Dimensions:    (chan: 15, pol: 1, coef_indx: 5)
Coordinates:
    band       (chan) <U1 dask.array<chunksize=(15,), meta=np.ndarray>
  * chan       (chan) float64 2.052e+09 2.18e+09 ... 3.82e+09 3.948e+09
  * coef_indx  (coef_indx) int64 0 1 2 3 4
  * pol        (pol) int64 5
Data variables:
    BPC        (chan, pol, coef_indx) float64 dask.array<chunksize=(15, 1, 5),␣
↪meta=np.ndarray>
    ETA        (chan, pol, coef_indx) float64 dask.array<chunksize=(15, 1, 5),␣
↪meta=np.ndarray>
Attributes:
    conversion_date:  2022-01-27
    dish_diam:        25
    max_rad_1GHz:     0.014946999714079439
    pc_file_name:     EVLA_.txt
    telescope_name:   EVLA
```

[6]:
```
from sirius.calc_beam import calc_bpc_beam

J_xds = calc_bpc_beam(pc_xds,pc_xds.chan.values,beam_parms={})
J_xds
```

```
Setting default fov_scaling  to  4.0
Setting default mueller_selection  to  [ 0  5 10 15]
Setting default zernike_freq_interp  to  nearest
Setting default pa_radius  to  0.2
Setting default image_size  to  [1000 1000]
cell_size  [-2.91364517e-05  2.91364517e-05]
```

[6]:
```
<xarray.Dataset>
Dimensions:  (chan: 15, pa: 1, pol: 1, l: 1000, m: 1000)
Coordinates:
  * chan     (chan) float64 2.052e+09 2.18e+09 2.436e+09 ... 3.82e+09 3.948e+09
  * pa       (pa) int64 0
  * pol      (pol) int64 0
```

(continues on next page)

(continued from previous page)

```
  * l          (l) float64 0.01457 0.01454 0.01451 ... -0.01448 -0.01451 -0.01454
  * m          (m) float64 -0.01457 -0.01454 -0.01451 ... 0.01448 0.01451 0.01454
Data variables:
    J          (pa, chan, pol, l, m) float64 0.0 0.0 0.0 0.0 ... 0.0 0.0 0.0 0.0
```

[7]:
```
from sirius.display_tools import display_J

interactive_plot_1 = interactive(display_J, J_xds=fixed(J_xds), pa=fixed(0), chan=(0,
↪J_xds.dims['chan']-1), val_type=['abs','phase','real','imag'],units=['rad','arcsec',
↪'arcmin','deg'])
output_1 = interactive_plot_1.children[-1]
output_1.layout.auto_scroll_threshold = 9999;
interactive_plot_1
```

```
interactive(children=(IntSlider(value=7, description='chan', max=14),␣
↪Dropdown(description='val_type', options...
```

4.4 Jones Matrix

Sky Jones matrix as a function of direction is given by

$$\mathbf{J}_i^{sky}(\mathbf{s}) = \begin{bmatrix} J_i^p & -J_i^{p\to q} \\ J_i^{q\to p} & J_i^q \end{bmatrix}$$

Simplified case of the Airy disk (real valued and symmetric)

$$\mathbf{J}_i^{sky}(\mathbf{s}) = \begin{bmatrix} J_i^p & 0 \\ 0 & J_i^p \end{bmatrix}$$

No leakage and p=q.

SiRIUS stores Jones matrices row wise

$$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}$$

[8]:
```
import pkg_resources

zpc_dir = pkg_resources.resource_filename('sirius_data', 'aperture_polynomial_
↪coefficient_models/data/EVLA_avg_zcoeffs_SBand_lookup.apc.zarr')
zpc_xds = xr.open_zarr(zpc_dir,consolidated=False)
freq_chan = zpc_xds.chan.values#np.array([2.1*10**9,2.4*10**9,2.8*10**9,3.4*10**9])

zpc_xds
```

[8]:
```
<xarray.Dataset>
Dimensions:    (chan: 16, pol: 4, coef_indx: 66)
Coordinates:
  * chan       (chan) float64 2.052e+09 2.18e+09 ... 3.844e+09 3.972e+09
  * coef_indx  (coef_indx) int64 0 1 2 3 4 5 6 7 8 ... 58 59 60 61 62 63 64 65
  * pol        (pol) int64 5 6 7 8
Data variables:
    ETA        (chan, pol, coef_indx) float64 dask.array<chunksize=(16, 4, 66),␣
↪meta=np.ndarray>
```

(continues on next page)

(continued from previous page)

```
    ZPC        (chan, pol, coef_indx) complex128 dask.array<chunksize=(16, 4, 66),␣
↪meta=np.ndarray>
Attributes:
    apc_file_name:    EVLA_avg_zcoeffs_SBand_lookup.csv
    conversion_date:  2022-01-27
    dish_diam:        25
    max_rad_1GHz:     0.014946999714079439
    telescope_name:   EVLA
```

[9]:
```
beam_parms={}
beam_parms['mueller_selection'] = np.arange(16)
J_xds_zpc = calc_zpc_beam(zpc_xds,np.array([0,np.pi/4,np.pi*2/4,np.pi*3/4,np.pi]),
↪freq_chan,beam_parms)
J_xds_zpc
```

```
Setting default fov_scaling  to  4.0
Setting default zernike_freq_interp  to  nearest
Setting default pa_radius  to  0.2
Setting default image_size  to  [1000 1000]
```

[9]:
```
<xarray.Dataset>
Dimensions:  (chan: 16, pa: 5, pol: 4, l: 1000, m: 1000)
Coordinates:
  * chan     (chan) float64 2.052e+09 2.18e+09 2.308e+09 ... 3.844e+09 3.972e+09
  * pa       (pa) float64 0.0 0.7854 1.571 2.356 3.142
  * pol      (pol) int64 5 6 7 8
  * l        (l) float64 0.01457 0.01454 0.01451 ... -0.01448 -0.01451 -0.01454
  * m        (m) float64 -0.01457 -0.01454 -0.01451 ... 0.01448 0.01451 0.01454
Data variables:
    J        (pa, chan, pol, l, m) complex128 (-0.12695041266544269-0.0096647...
```

[10]:
```
interactive_plot_1 = interactive(display_J, J_xds=fixed(J_xds_zpc), pa=(0,J_xds_zpc.
↪dims['pa']-1), chan=(0,J_xds_zpc.dims['chan']-1), val_type=['abs','phase','real',
↪'imag'],units=['rad','arcsec','arcmin','deg'])
output_1 = interactive_plot_1.children[-1]
output_1.layout.auto_scroll_threshold = 9999;
interactive_plot_1
```

```
interactive(children=(IntSlider(value=2, description='pa', max=4), IntSlider(value=7,␣
↪description='chan', max=...
```

4.5 Mueller Matrix

$$\mathbf{M}_{ij}^{sky}(\mathbf{s}) = \mathbf{J}_{i}^{sky}(\mathbf{s}) \otimes \mathbf{J}_{j}^{sky}(\mathbf{s})$$

SiRIUS stores Jones matrices row wise

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \end{bmatrix}$$

[11]:
```
#Het array support
#make_mueler_mat(J_xds1, J_xds2, mueller_selection)

M_xds = make_mueler_mat(J_xds_zpc,J_xds_zpc,beam_parms['mueller_selection']) #np.
↪array([ 0, 5, 10, 15])
M_xds
```

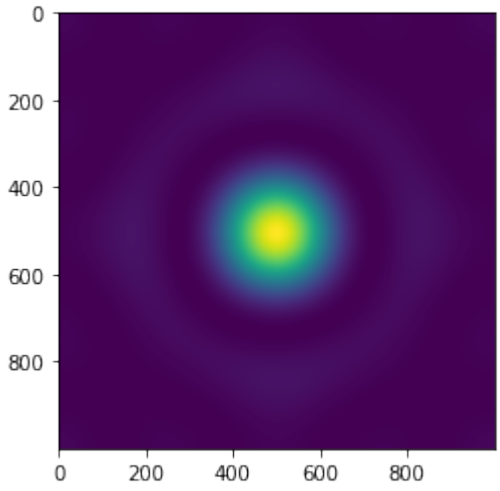
```
(5, 16, 4, 1000, 1000) (5, 16, 4, 1000, 1000)
```

[11]:
```
<xarray.Dataset>
Dimensions:  (chan: 16, pa: 5, m_sel: 16, l: 1000, m: 1000)
Coordinates:
  * chan     (chan) float64 2.052e+09 2.18e+09 2.308e+09 ... 3.844e+09 3.972e+09
  * pa       (pa) float64 0.0 0.7854 1.571 2.356 3.142
  * m_sel    (m_sel) int64 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
  * l        (l) float64 0.01457 0.01454 0.01451 ... -0.01448 -0.01451 -0.01454
  * m        (m) float64 -0.01457 -0.01454 -0.01451 ... 0.01448 0.01451 0.01454
    pol1     (m_sel) float64 5.0 5.0 6.0 6.0 5.0 5.0 ... 8.0 8.0 7.0 7.0 8.0 8.0
    pol2     (m_sel) float64 5.0 6.0 5.0 6.0 7.0 8.0 ... 5.0 6.0 7.0 8.0 7.0 8.0
Data variables:
    M        (pa, chan, m_sel, l, m) complex128 (0.01620981518531084+0j) ... ...
```

[12]:
```
interactive_plot_1 = interactive(display_M, M_xds=fixed(M_xds), pa=(0,J_xds_zpc.dims[
↪'pa']-1), chan=(0,J_xds_zpc.dims['chan']-1), val_type=['abs','phase','real','imag'],
↪units=['rad','arcsec','arcmin','deg'])
output_1 = interactive_plot_1.children[-1]
output_1.layout.auto_scroll_threshold = 9999;
interactive_plot_1
```

```
interactive(children=(IntSlider(value=2, description='pa', max=4), IntSlider(value=7,␣
↪description='chan', max=...
```

[13]:
```
import matplotlib.pyplot as plt
plt.figure()
plt.imshow(np.real(M_xds.M[0,0,0,:,:]))
plt.show()
```



[14]:
```
%load_ext autoreload
%autoreload 2
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

CHAPTER

FIVE

BASIC SIMULATION

[1]:
```
import os
try:
    import sirius
    print('SiRIUS version',sirius.__version__,'already installed.')
except ImportError as e:
    print(e)
    print('Installing SiRIUS')
    os.system("pip install sirius")
    import sirius
    print('SiRIUS version',sirius.__version__,' installed.')
```

```
SiRIUS version 0.0.28 already installed.
```

5.1 Load Packages

[2]:
```
import pkg_resources
import xarray as xr
import numpy as np
from astropy.coordinates import SkyCoord
xr.set_options(display_style="html")
import os
try:
    from google.colab import output
    output.enable_custom_widget_manager()
    IN_COLAB = True
    %matplotlib widget
except:
    IN_COLAB = False
    %matplotlib inline
```

5.2 Load Telescope Layout

[3]:
```
########## Get telescope layout ##########
tel_dir = pkg_resources.resource_filename('sirius_data', 'telescope_layout/data/vla.d.
↪tel.zarr')
tel_xds = xr.open_zarr(tel_dir,consolidated=False)
n_ant = tel_xds.dims['ant_name']
#tel_xds.attrs['telescope_name'] = 'EVLA'
tel_xds
```

[3]:
```
<xarray.Dataset>
Dimensions:         (ant_name: 27, pos_coord: 3)
Coordinates:
  * ant_name        (ant_name) <U3 'W01' 'W02' 'W03' 'W04' ... 'N07' 'N08' 'N09'
  * pos_coord       (pos_coord) int64 0 1 2
Data variables:
    ANT_POS         (ant_name, pos_coord) float64 dask.array<chunksize=(27, 3),␣
↪meta=np.ndarray>
    DISH_DIAMETER   (ant_name) float64 dask.array<chunksize=(27,), meta=np.ndarray>
Attributes:
    site_pos:         [{'m0': {'unit': 'm', 'value': -1601185.3650000016}, 'm1...
    telescope_name:   VLA
```

5.3 Create Time and Freq Xarrays

The chunking of time_xda and chan_xda determines the number of branches in the DAG (maximum parallelism = n_time_chunks x n_chan_chunks).

[4]:
```
from sirius.dio import make_time_xda
#time_xda = make_time_xda(time_start='2019-10-03T19:00:00.000',time_delta=3600,n_
↪samples=10,n_chunks=4)
time_xda = make_time_xda(time_start='2019-10-03T19:00:00.000',time_delta=3600,n_
↪samples=10,n_chunks=1)
time_xda
```

```
Number of chunks  1
```

[4]:
```
<xarray.DataArray 'array-51a16e79cab65fc8a0bc3efd397119f7' (time: 10)>
dask.array<array, shape=(10,), dtype=<U23, chunksize=(10,), chunktype=numpy.ndarray>
Dimensions without coordinates: time
Attributes:
    time_delta:  3600.0
```

[5]:
```
from sirius.dio import make_chan_xda
spw_name = 'SBand'
#chan_xda = make_chan_xda(freq_start = 3*10**9, freq_delta = 0.4*10**9, freq_
↪resolution=0.01*10**9, n_channels=3, n_chunks=3)
chan_xda = make_chan_xda(freq_start = 3*10**9, freq_delta = 0.4*10**9, freq_
↪resolution=0.01*10**9, n_channels=3, n_chunks=1)
chan_xda
```

```
Number of chunks  1
```

[5]:
```
<xarray.DataArray 'array-4bc1685db34e1d77709ed218dffb158a' (chan: 3)>
dask.array<array, shape=(3,), dtype=float64, chunksize=(3,), chunktype=numpy.ndarray>
Dimensions without coordinates: chan
Attributes:
    freq_resolution:  10000000.0
    spw_name:         sband
    freq_delta:       400000000.0
```

5.4 Beam Models

[6]:
```
from sirius_data.beam_1d_func_models.airy_disk import vla #Get airy disk parameters␣
↪for VLA dish (obtained from https://open-bitbucket.nrao.edu/projects/CASA/repos/
↪casa6/browse/casatools/src/code/synthesis/TransformMachines/PBMath.cc)
airy_disk_parms = vla
print(airy_disk_parms)

beam_models = [airy_disk_parms]
beam_model_map = np.zeros(n_ant,dtype=int) #Maps the antenna index to a model in beam_
↪models.
beam_parms = {} #Use default beam parms.

#If no beam should be used:
#none_model = {'pb_func':'none','dish_diameter':0.0,'blockage_diameter':0.0}

'''
#If Zernike Polynomial should be used:
zpc_dir = pkg_resources.resource_filename('sirius_data', 'aperture_polynomial_
↪coefficient_models/data/EVLA_avg_zcoeffs_SBand_lookup.apc.zarr')
zpc_xds = xr.open_zarr(zpc_dir,consolidated=False)

beam_models = [zpc_xds]
zpc_xds

beam_models = [zpc_xds,airy_disk_parms]

beam_model_map[0] = 1

zpc_xds
'''
```

```
{'func': 'casa_airy', 'dish_diam': 24.5, 'blockage_diam': 0.0, 'max_rad_1GHz': 0.
↪014946999714079439}
```

[6]:
```
"\n#If Zernike Polynomial should be used:\nzpc_dir = pkg_resources.resource_filename(
↪'sirius_data', 'aperture_polynomial_coefficient_models/data/EVLA_avg_zcoeffs_SBand_
↪lookup.apc.zarr')\nzpc_xds = xr.open_zarr(zpc_dir,consolidated=False)\n\nbeam_
↪models = [zpc_xds]\nzpc_xds\n\nbeam_models = [zpc_xds,airy_disk_parms]\n\nbeam_
↪model_map[0] = 1\n\nzpc_xds\n"
```

5.5 Polarization Setup

[7]:
```
#https://github.com/casacore/casacore/blob/dbf28794ef446bbf4e6150653dbe404379a3c429/
↪measures/Measures/Stokes.h
# ['RR','RL','LR','LL'] => [5,6,7,8], ['XX','XY','YX','YY'] => [9,10,11,12]
pol = [5,8]
```

5.6 UVW Parameters

[8]:
```
# If using uvw_parms['calc_method'] = 'casa' .casarc must have directory of casadata.
import pkg_resources
casa_data_dir = pkg_resources.resource_filename('casadata', '__data__')
rc_file = open(os.path.expanduser("~/.casarc"), "a+")  # append mode
rc_file.write("\n measures.directory: " + casa_data_dir)
rc_file.close()

uvw_parms = {}
uvw_parms['calc_method'] = 'casa' #'astropy' or 'casa'
uvw_parms['auto_corr'] = False
```

5.7 Sources

5.7.1 point_source_ra_dec: [n_time, n_point_sources, 2] (singleton: n_time)

[9]:
```
point_source_skycoord = SkyCoord(ra='19h59m50.51793355s',dec='+40d48m11.3694551s',
↪frame='fk5')
point_source_ra_dec = np.array([point_source_skycoord.ra.rad,point_source_skycoord.
↪dec.rad])[None,None,:]
```

5.7.2 point_source_flux: [n_point_sources, n_time, n_chan, n_pol] (singleton: n_time, n_chan)

All 4 instramental pol values must be given even if only RR and LL are requested.

[10]:
```
point_source_flux = np.array([1.0, 0, 0, 1.0])[None,None,None,:]
```

5.8 Telescope Setup

5.8.1 phase_center: [n_time, 2] (singleton: n_time)

5.8.2 phase_center_names: [n_time] (singleton: n_time)

[11]:
```
phase_center = SkyCoord(ra='19h59m28.5s',dec='+40d44m01.5s',frame='fk5')
phase_center_ra_dec = np.array([phase_center.ra.rad,phase_center.dec.rad])[None,:]
phase_center_names = np.array(['field1'])
```

[12]:
```
%load_ext autoreload
%autoreload 2
```

[13]:
```
from sirius._sirius_utils._coord_transforms import _calc_rotation_mats, _directional_
↪cosine,  _sin_project

print(phase_center_ra_dec[0,:],point_source_ra_dec[0,:,:])
lm_sin = _sin_project(phase_center_ra_dec[0,:],point_source_ra_dec[0,:,:])[0,:]
```

(continues on next page)

(continued from previous page)

```
print('%.15f' % lm_sin[0],'%.15f' % lm_sin[1])
```

```
[5.23369701 0.71093805] [[5.2352982  0.71214946]]
0.001212034202758 0.001212034202764
```

5.8.3 pointing_ra_dec:[n_time, n_ant, 2] (singleton: n_time, n_ant) or None

[14]:
```
pointing_ra_dec = None #No pointing offsets
```

5.9 Noise

[15]:
```
noise_parms = None
#noise_parms ={}
```

5.10 Run Simulation

Write to disk as a measuremnt set and create DAGs.

[16]:
```
%load_ext autoreload
%autoreload 2
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

[17]:
```
from sirius import simulation
save_parms = {'ms_name':'simple_sim.ms','write_to_ms':True,'DAG_name_vis_uvw_gen':
↪'DAG_vis_uvw_gen.png','DAG_name_write':'DAG_write.png'}
ms_xds = simulation(point_source_flux,
                    point_source_ra_dec,
                    pointing_ra_dec,
                    phase_center_ra_dec,
                    phase_center_names,
                    beam_parms,beam_models,
                    beam_model_map,uvw_parms,
                    tel_xds, time_xda, chan_xda, pol, noise_parms, save_parms)
ms_xds
```

```
Setting default fov_scaling  to  4.0
Setting default mueller_selection  to  [ 0  5 10 15]
Setting default zernike_freq_interp  to  nearest
Setting default pa_radius  to  0.2
Setting default image_size  to  [1000 1000]
Setting default mode  to  dask_ms_and_sim_tool
10 351 3 2
<xarray.Dataset>
Dimensions:  (time: 10, pol: 2, chan: 3, baseline: 351, uvw: 3, time_chunk: 1,
              chan_chunk: 1, 4: 4)
Coordinates:
  * time     (time) <U23 '2019-10-03T19:00:00.000' ... '2019-10-04T04:00:00.000'
```

(continues on next page)

(continued from previous page)

```
  * pol      (pol) int64 5 8
  * chan     (chan) float64 3e+09 3.4e+09 3.8e+09
Dimensions without coordinates: baseline, uvw, time_chunk, chan_chunk, 4
Data variables:
    DATA     (time, baseline, chan, pol) complex128 dask.array<chunksize=(10, 351, 3,␣
↪2), meta=np.ndarray>
    UVW      (time, baseline, uvw) complex128 dask.array<chunksize=(10, 351, 3),␣
↪meta=np.ndarray>
    WEIGHT   (time, baseline, pol) float64 dask.array<chunksize=(10, 351, 2), meta=np.
↪ndarray>
    SIGMA    (time, baseline, pol) float64 dask.array<chunksize=(10, 351, 2), meta=np.
↪ndarray>
    TIMING   (time_chunk, chan_chunk, 4) float64 dask.array<chunksize=(1, 1, 4),␣
↪meta=np.ndarray>
Meta data creation  0.4570884704589844
reshape time  0.0011935234069824219
*** Dask compute time 4.647282600402832
```

[17]:
```
<xarray.Dataset>
Dimensions:           (polarization_ids: 1, spw_ids: 1)
Coordinates:
  * polarization_ids  (polarization_ids) int64 0
  * spw_ids           (spw_ids) int64 0
Data variables:
    *empty*
Attributes:
    xds0:              <xarray.Dataset>\nDimensions:         (row: 3510, uvw_...
    SPECTRAL_WINDOW:   <xarray.Dataset>\nDimensions:          (row: 1, d1: 3)...
    POLARIZATION:      <xarray.Dataset>\nDimensions:       (row: 1, d1: 2, d2...
    DATA_DESCRIPTION:  <xarray.Dataset>\nDimensions:             (row: 1)\nCo...
```

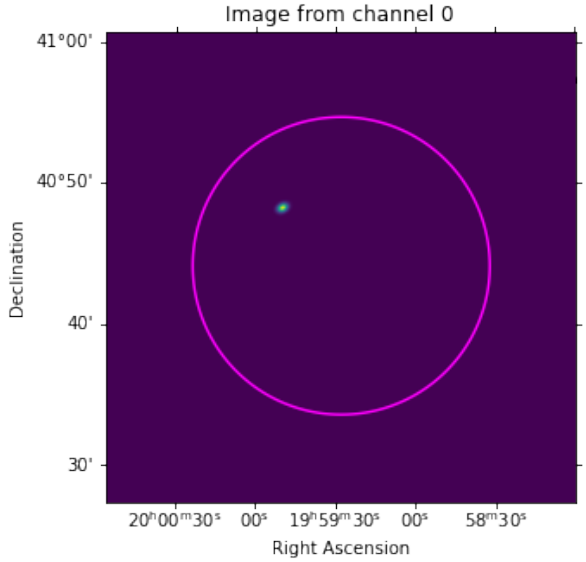
5.11 Image Simulated Dataset

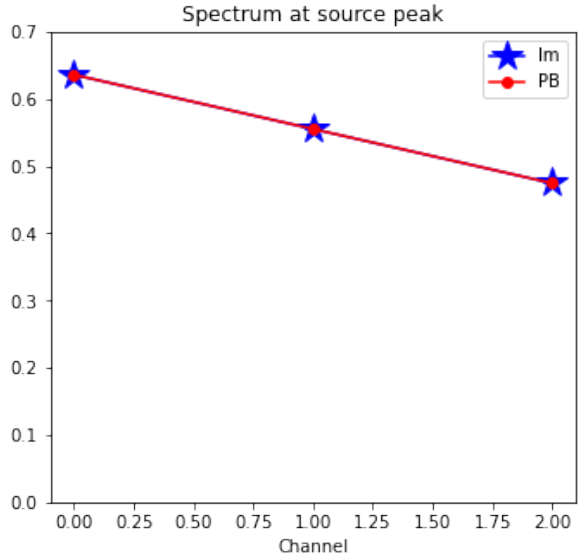
[18]:
```
from casatasks import tclean
os.system('rm -rf simple.*')
tclean(vis=save_parms['ms_name'],imagename='simple',imsize=[400,400],cell=[5.0,5.0],
↪specmode='cube',niter=1000,pblimit=0.1,pbmask=0.1,gridder='standard',stokes='RR')
```

[18]:
```
{}
```

[21]:
```
chan=0
from sirius.display_tools import display_image
print(np.abs(ms_xds.xds0.DATA[0,chan,0].values))
display_image(imname='simple.image',pbname='simple.pb',resname='simple.residual',
↪ylim=[0.0,0.7],chan=chan)
```

```
0.6360993
Peak Intensity (chan0) : 0.6360901
PB at location of Intensity peak (chan0) : 0.6360983
max pixel location  (array([150]), array([250]), array([0]), array([0]))
Residual RMS : 0.0001055
```





[21]: (0.636090099811554, 0.00010554427406769702)

[ ]:

CHAPTER

SIX

HETEROGENEOUS ARRAY MOSAIC SIMULATIONS AND IMAGING

[1]:
```
import os
try:
    import sirius
    print('SiRIUS version',sirius.__version__,'already installed.')
except ImportError as e:
    print(e)
    print('Installing SiRIUS')
    os.system("pip install sirius")
    import sirius
    print('SiRIUS version',sirius.__version__,' installed.')
```

```
SiRIUS version 0.0.21 already installed.
```

[2]:
```
import pkg_resources
import xarray as xr
import numpy as np
from astropy.coordinates import SkyCoord
xr.set_options(display_style="html")
import os
try:
    from google.colab import output
    output.enable_custom_widget_manager()
    IN_COLAB = True
    %matplotlib widget
except:
    IN_COLAB = False
    %matplotlib inline

#working_dir = '/lustre/cv/users/jsteeb/simulation/'
working_dir = ''
```

6.1 Telescope Layout

[3]:
```
########## Get telescope layout ##########
tel_dir = pkg_resources.resource_filename('sirius_data', 'telescope_layout/data/alma.
↪all.tel.zarr')
tel_xds = xr.open_zarr(tel_dir,consolidated=False).sel(ant_name = ['N601','N606','J505
↪','J510', 'A001', 'A012','A025', 'A033','A045', 'A051','A065', 'A078'])
n_ant = tel_xds.dims['ant_name']
tel_xds
```

[3]:
```
<xarray.Dataset>
Dimensions:         (ant_name: 12, pos_coord: 3)
Coordinates:
  * ant_name        (ant_name) <U7 'N601' 'N606' 'J505' ... 'A051' 'A065' 'A078'
  * pos_coord       (pos_coord) int64 0 1 2
Data variables:
    ANT_POS         (ant_name, pos_coord) float64 dask.array<chunksize=(12, 3),␣
↪meta=np.ndarray>
    DISH_DIAMETER   (ant_name) float64 dask.array<chunksize=(12,), meta=np.ndarray>
Attributes:
    site_pos:         [{'m0': {'unit': 'm', 'value': 2225142.180268967}, 'm1':...
    telescope_name:   ALMA
```

6.2 Create Time and Freq Xarrays

The chunking of time_xda and chan_xda determines the number of branches in the DAG (maximum parallelism = n_time_chunks x n_chan_chunks).

[4]:
```
from sirius.dio import make_time_xda

#time_xda = make_time_xda(time_start='2020-10-03T18:56:36.44',time_delta=2000,n_
↪samples=18,n_chunks=4)
#time_xda = make_time_xda(time_start='2020-10-03T18:57:29.09',time_delta=2000,n_
↪samples=18,n_chunks=4)
#time_xda = make_time_xda(time_start='2020-10-04T00:00:00.000',time_delta=2000,n_
↪samples=18,n_chunks=4)

#time_xda = make_time_xda(time_start='2020-10-03T18:57:28.95',time_delta=2000,n_
↪samples=18,n_chunks=2)
time_xda = make_time_xda(time_start='2020-10-03T18:57:28.95',time_delta=2000,n_
↪samples=18,n_chunks=1)
time_xda
```

```
Number of chunks  1
```

[4]:
```
<xarray.DataArray 'array-a26c3c2e3ac33048ef565ef252d802d9' (time: 18)>
dask.array<array, shape=(18,), dtype=<U23, chunksize=(18,), chunktype=numpy.ndarray>
Dimensions without coordinates: time
Attributes:
    time_delta:  2000.0
```

[5]:
```
from sirius.dio import make_chan_xda    #n_channels = 5
#chan_xda = make_chan_xda(spw_name = 'Band3',freq_start = 90*10**9, freq_delta =␣
↪2*10**9, freq_resolution=1*10**6, n_channels=5, n_chunks=2)
chan_xda = make_chan_xda(spw_name = 'Band3',freq_start = 90*10**9, freq_delta =␣
↪2*10**9, freq_resolution=1*10**6, n_channels=5, n_chunks=1)
chan_xda
```

```
Number of chunks  1
```

[5]:
```
<xarray.DataArray 'array-0abbfd50a841ad468a340137153842c0' (chan: 5)>
dask.array<array, shape=(5,), dtype=float64, chunksize=(5,), chunktype=numpy.ndarray>
Dimensions without coordinates: chan
Attributes:
    freq_resolution:  1000000.0
    spw_name:         Band3
    freq_delta:       2000000000.0
```

Beam Models

[6]:
```
from sirius_data.beam_1d_func_models.airy_disk import alma, aca
airy_disk_parms_alma =  alma
airy_disk_parms_aca =  aca

print(alma)
print(aca)

beam_models = [airy_disk_parms_aca, airy_disk_parms_alma]

# beam_model_map maps the antenna index to a model in beam_models.
beam_model_map = tel_xds.DISH_DIAMETER.values
beam_model_map[beam_model_map == 7] = 0
beam_model_map[beam_model_map == 12] = 1
beam_model_map= beam_model_map.astype(int)

beam_parms = {} #Use default beam parms.
```

```
{'func': 'casa_airy', 'dish_diam': 10.7, 'blockage_diam': 0.75, 'max_rad_1GHz': 0.
↪03113667385557884}
{'func': 'casa_airy', 'dish_diam': 6.25, 'blockage_diam': 0.75, 'max_rad_1GHz': 0.
↪06227334771115768}
```

6.3 Polarization Setup

[7]:
```
#https://github.com/casacore/casacore/blob/dbf28794ef446bbf4e6150653dbe404379a3c429/
↪measures/Measures/Stokes.h
# ['RR','RL','LR','LL'] => [5,6,7,8], ['XX','XY','YX','YY'] => [9,10,11,12]
pol = [9,12]
```

6.4 UVW Parameters

[8]:
```
# If using uvw_parms['calc_method'] = 'casa' .casarc must have directory of casadata.
import pkg_resources
casa_data_dir = pkg_resources.resource_filename('casadata', '__data__')
rc_file = open(os.path.expanduser("~/.casarc"), "a+")  # append mode
rc_file.write("\n measures.directory: " + casa_data_dir)
rc_file.close()

uvw_parms = {}
uvw_parms['calc_method'] = 'casa' #'astropy' or 'casa'Heterogeneous Array Mosaic␣
↪Simulations and Imaging
```

6.5 Sources

6.5.1 point_source_ra_dec: [n_time, n_point_sources, 2] (singleton: n_time)

[9]:
```
point_source_skycoord = SkyCoord(ra='19h59m28.5s',dec='-40d44m21.5s',frame='fk5')
point_source_ra_dec = np.array([point_source_skycoord.ra.rad,point_source_skycoord.
↪dec.rad])[None,None,:]
```

6.5.2 point_source_flux: [n_point_sources, n_time, n_chan, n_pol] (singleton: n_time, n_chan)

All 4 instramental pol values must be given even if only RR and LL are requested.

[10]:
```
point_source_flux = np.array([1.0, 0, 0, 1.0])[None,None,None,:]
```

6.6 Telescope Setup

6.6.1 phase_center: [n_time, 2] (singleton: n_time)

6.6.2 phase_center_names: [n_time] (singleton: n_time)

[11]:
```
n_time = len(time_xda)
n_time_per_field = int(n_time/2)
phase_center_1 = SkyCoord(ra='19h59m28.5s',dec='-40d44m01.5s',frame='fk5')
phase_center_2 = SkyCoord(ra='19h59m28.5s',dec='-40d44m51.5s',frame='fk5')
phase_center_names = np.array(['field1']*n_time_per_field + ['field2']*n_time_per_
↪field)
phase_center_ra_dec = np.array([[phase_center_1.ra.rad,phase_center_1.dec.rad]]*n_
↪time_per_field + [[phase_center_2.ra.rad,phase_center_2.dec.rad]]*n_time_per_field)
```

6.6.3 pointing_ra_dec:[n_time, n_ant, 2] (singleton: n_time, n_ant) or None

[12]:
```
pointing_ra_dec = None #No pointing offsets
```

6.7 Run Simulation

[13]:
```
noise_parms = None

from sirius import simulation
save_parms = {'ms_name':working_dir+'het_mosaic_sim.ms','write_to_ms':True}
vis_xds = simulation(point_source_flux,
                     point_source_ra_dec,
                     pointing_ra_dec,
                     phase_center_ra_dec,
                     phase_center_names,
```

(continues on next page)

(continued from previous page)

```
                    beam_parms,beam_models,
                    beam_model_map,uvw_parms,
                    tel_xds, time_xda, chan_xda, pol, noise_parms, save_parms)
vis_xds
```

```
Setting default auto_corr  to  False
Setting default fov_scaling  to  4.0
Setting default mueller_selection  to  [ 0  5 10 15]
Setting default zernike_freq_interp  to  nearest
Setting default pa_radius  to  0.2
Setting default image_size  to  [1000 1000]
Setting default DAG_name_vis_uvw_gen  to  False
Setting default DAG_name_write  to  False
Meta data creation  30.654367208480835
reshape time  0.00202178955078125
*** Dask compute time 10.297820568084717
compute and save time  10.298502206802368
```

[13]:
```
[<xarray.Dataset>
 Dimensions:          (row: 594, chan: 5, corr: 2, uvw: 3)
 Coordinates:
     ROWID            (row) int32 dask.array<chunksize=(594,), meta=np.ndarray>
 Dimensions without coordinates: row, chan, corr, uvw
 Data variables: (12/21)
     TIME_CENTROID    (row) float64 dask.array<chunksize=(594,), meta=np.ndarray>
     SCAN_NUMBER      (row) int32 dask.array<chunksize=(594,), meta=np.ndarray>
     ANTENNA1         (row) int32 dask.array<chunksize=(594,), meta=np.ndarray>
     FEED1            (row) int32 dask.array<chunksize=(594,), meta=np.ndarray>
     INTERVAL         (row) float64 dask.array<chunksize=(594,), meta=np.ndarray>
     FLAG_ROW         (row) bool dask.array<chunksize=(594,), meta=np.ndarray>
     ...               ...
     UVW              (row, uvw) float64 dask.array<chunksize=(594, 3), meta=np.
↪ndarray>
     TIME             (row) float64 dask.array<chunksize=(594,), meta=np.ndarray>
     FLAG             (row, chan, corr) bool dask.array<chunksize=(594, 5, 2), meta=np.
↪ndarray>
     STATE_ID         (row) int32 dask.array<chunksize=(594,), meta=np.ndarray>
     ARRAY_ID         (row) int32 dask.array<chunksize=(594,), meta=np.ndarray>
     CORRECTED_DATA   (row, chan, corr) complex64 dask.array<chunksize=(594, 5, 2),␣
↪meta=np.ndarray>
 Attributes:
     __daskms_partition_schema__:  (('FIELD_ID', 'int32'), ('DATA_DESC_ID', 'i...
     FIELD_ID:                     0
     DATA_DESC_ID:                 0,
 <xarray.Dataset>
 Dimensions:          (row: 594, chan: 5, corr: 2, uvw: 3)
 Coordinates:
     ROWID            (row) int32 dask.array<chunksize=(594,), meta=np.ndarray>
 Dimensions without coordinates: row, chan, corr, uvw
 Data variables: (12/21)
     TIME_CENTROID    (row) float64 dask.array<chunksize=(594,), meta=np.ndarray>
     SCAN_NUMBER      (row) int32 dask.array<chunksize=(594,), meta=np.ndarray>
     ANTENNA1         (row) int32 dask.array<chunksize=(594,), meta=np.ndarray>
     FEED1            (row) int32 dask.array<chunksize=(594,), meta=np.ndarray>
     INTERVAL         (row) float64 dask.array<chunksize=(594,), meta=np.ndarray>
     FLAG_ROW         (row) bool dask.array<chunksize=(594,), meta=np.ndarray>
     ...               ...
     UVW              (row, uvw) float64 dask.array<chunksize=(594, 3), meta=np.
↪ndarray>
```

(continues on next page)

(continued from previous page)

```
    TIME            (row) float64 dask.array<chunksize=(594,), meta=np.ndarray>
    FLAG            (row, chan, corr) bool dask.array<chunksize=(594, 5, 2), meta=np.
↪ndarray>
    STATE_ID        (row) int32 dask.array<chunksize=(594,), meta=np.ndarray>
    ARRAY_ID        (row) int32 dask.array<chunksize=(594,), meta=np.ndarray>
    CORRECTED_DATA  (row, chan, corr) complex64 dask.array<chunksize=(594, 5, 2),␣
↪meta=np.ndarray>
Attributes:
    __daskms_partition_schema__:  (('FIELD_ID', 'int32'), ('DATA_DESC_ID', 'i...
    FIELD_ID:                     1
    DATA_DESC_ID:                 0]
```

6.8 Noise simulation

[14]:
```
noise_parms = {'t_receiver':5000} #Increased receiver noise so that effect of noise␣
↪is easy to see in plots.

from sirius import simulation
save_parms_noisy = {'ms_name':working_dir+'het_mosaic_sim_noisy.ms','write_to_ms':
↪True}
vis_xds = simulation(point_source_flux,
                     point_source_ra_dec,
                     pointing_ra_dec,
                     phase_center_ra_dec,
                     phase_center_names,
                     beam_parms,beam_models,
                     beam_model_map,uvw_parms,
                     tel_xds, time_xda, chan_xda, pol, noise_parms, save_parms_noisy)
vis_xds
```

```
Setting default auto_corr  to  False
Setting default fov_scaling  to  4.0
Setting default mueller_selection  to  [ 0  5 10 15]
Setting default zernike_freq_interp  to  nearest
Setting default pa_radius  to  0.2
Setting default image_size  to  [1000 1000]
Setting default t_atmos  to  250.0
Currently the effect of Zenith Atmospheric Opacity (Tau) is not included in the noise␣
↪modeling.
Setting default tau  to  0.1
Setting default ant_efficiency  to  0.8
Setting default spill_efficiency  to  0.85
Setting default corr_efficiency  to  0.88
Setting default t_ground  to  270.0
Setting default t_cmb  to  2.725
Setting default DAG_name_vis_uvw_gen  to  False
Setting default DAG_name_write  to  False
Meta data creation  15.82155966758728
reshape time  0.0019490718841552734
*** Dask compute time 0.3486812114715576
compute and save time  0.3488438129425049
```

[14]:
```
[<xarray.Dataset>
 Dimensions:         (row: 594, chan: 5, corr: 2, uvw: 3)
 Coordinates:
```

(continues on next page)

(continued from previous page)

```
    ROWID           (row) int32 dask.array<chunksize=(594,), meta=np.ndarray>
Dimensions without coordinates: row, chan, corr, uvw
Data variables: (12/21)
    TIME_CENTROID   (row) float64 dask.array<chunksize=(594,), meta=np.ndarray>
    SCAN_NUMBER     (row) int32 dask.array<chunksize=(594,), meta=np.ndarray>
    ANTENNA1        (row) int32 dask.array<chunksize=(594,), meta=np.ndarray>
    FEED1           (row) int32 dask.array<chunksize=(594,), meta=np.ndarray>
    INTERVAL        (row) float64 dask.array<chunksize=(594,), meta=np.ndarray>
    FLAG_ROW        (row) bool dask.array<chunksize=(594,), meta=np.ndarray>
    ...              ...
    UVW             (row, uvw) float64 dask.array<chunksize=(594, 3), meta=np.
↪ndarray>
    TIME            (row) float64 dask.array<chunksize=(594,), meta=np.ndarray>
    FLAG            (row, chan, corr) bool dask.array<chunksize=(594, 5, 2), meta=np.
↪ndarray>
    STATE_ID        (row) int32 dask.array<chunksize=(594,), meta=np.ndarray>
    ARRAY_ID        (row) int32 dask.array<chunksize=(594,), meta=np.ndarray>
    CORRECTED_DATA  (row, chan, corr) complex64 dask.array<chunksize=(594, 5, 2),␣
↪meta=np.ndarray>
Attributes:
    __daskms_partition_schema__:  (('FIELD_ID', 'int32'), ('DATA_DESC_ID', 'i...
    FIELD_ID:                     0
    DATA_DESC_ID:                 0,
<xarray.Dataset>
Dimensions:         (row: 594, chan: 5, corr: 2, uvw: 3)
Coordinates:
    ROWID           (row) int32 dask.array<chunksize=(594,), meta=np.ndarray>
Dimensions without coordinates: row, chan, corr, uvw
Data variables: (12/21)
    TIME_CENTROID   (row) float64 dask.array<chunksize=(594,), meta=np.ndarray>
    SCAN_NUMBER     (row) int32 dask.array<chunksize=(594,), meta=np.ndarray>
    ANTENNA1        (row) int32 dask.array<chunksize=(594,), meta=np.ndarray>
    FEED1           (row) int32 dask.array<chunksize=(594,), meta=np.ndarray>
    INTERVAL        (row) float64 dask.array<chunksize=(594,), meta=np.ndarray>
    FLAG_ROW        (row) bool dask.array<chunksize=(594,), meta=np.ndarray>
    ...              ...
    UVW             (row, uvw) float64 dask.array<chunksize=(594, 3), meta=np.
↪ndarray>
    TIME            (row) float64 dask.array<chunksize=(594,), meta=np.ndarray>
    FLAG            (row, chan, corr) bool dask.array<chunksize=(594, 5, 2), meta=np.
↪ndarray>
    STATE_ID        (row) int32 dask.array<chunksize=(594,), meta=np.ndarray>
    ARRAY_ID        (row) int32 dask.array<chunksize=(594,), meta=np.ndarray>
    CORRECTED_DATA  (row, chan, corr) complex64 dask.array<chunksize=(594, 5, 2),␣
↪meta=np.ndarray>
Attributes:
    __daskms_partition_schema__:  (('FIELD_ID', 'int32'), ('DATA_DESC_ID', 'i...
    FIELD_ID:                     1
    DATA_DESC_ID:                 0]
```

6.9 Analysis

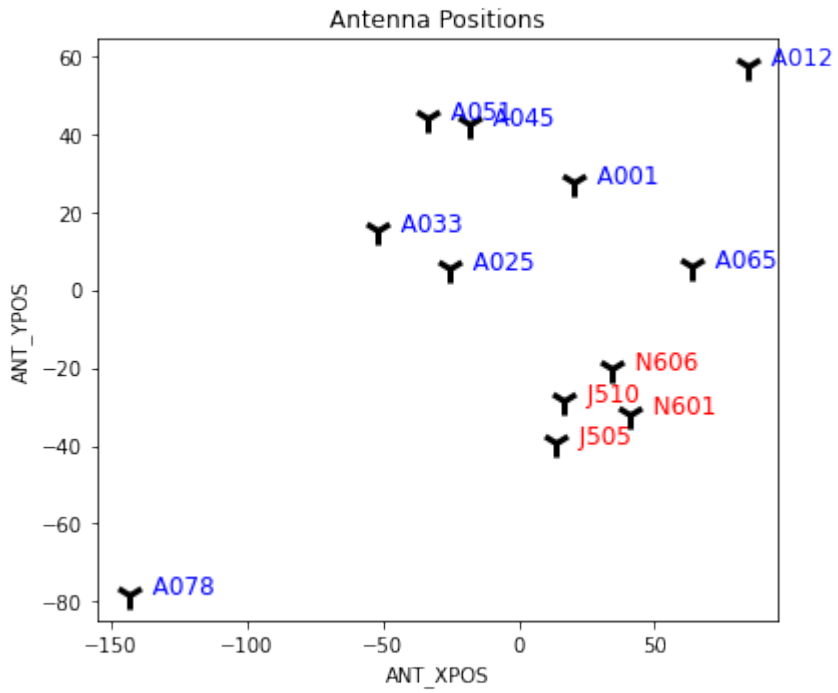
6.9.1 Plot Antennas

[15]:
```
%load_ext autoreload
%autoreload 2
```

[16]:
```
from sirius.display_tools import x_plot
```

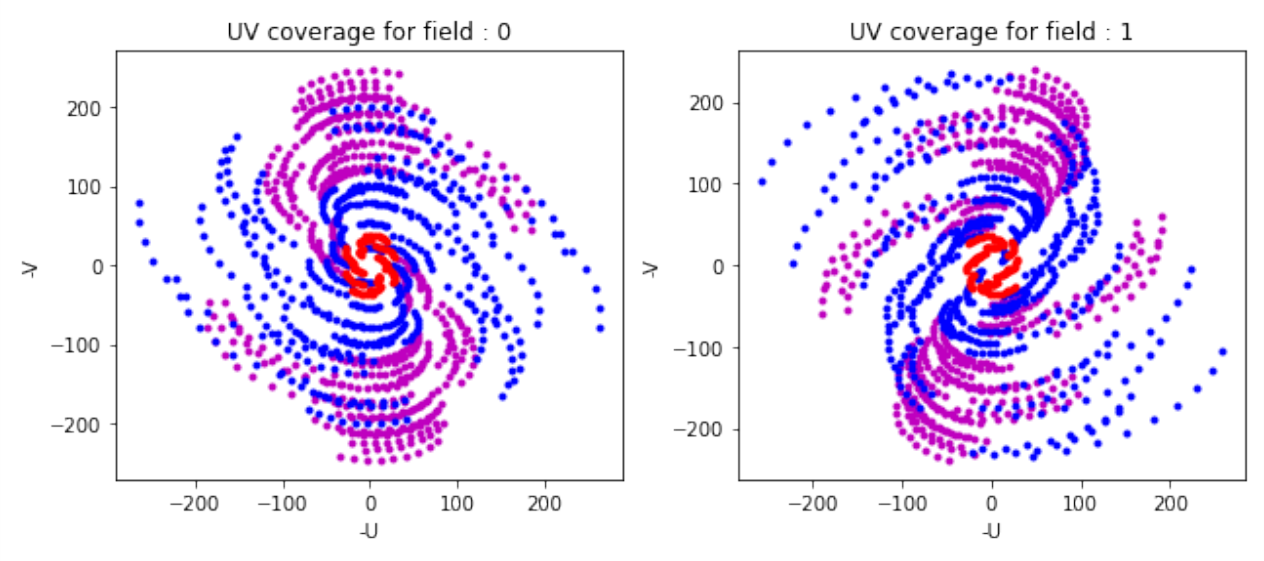
[17]:
```
x_plot(vis=save_parms['ms_name'], ptype='plotants')
```

```
Completed ddi 0  process time 4.26 s.ATA...SPECTRAL_WINDOW_ID...
Completed subtables  process time 6.16 s...

overwrite_encoded_chunks True
```



[18]:
```
from sirius.display_tools import x_plot
x_plot(vis=save_parms['ms_name'], ptype='uvcov',forceconvert=True)
```

```
Completed ddi 0  process time 4.28 s.ATA...SPECTRAL_WINDOW_ID...
Completed subtables  process time 6.53 s...

overwrite_encoded_chunks True
```

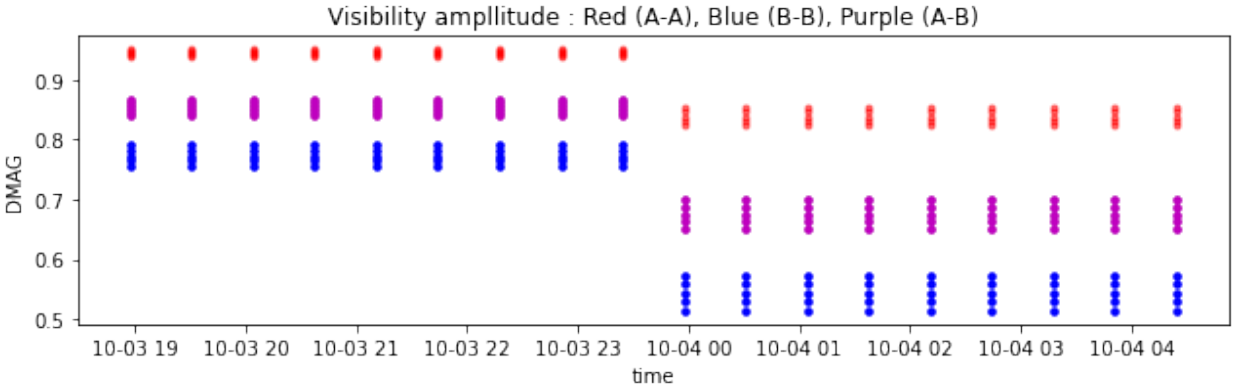


[19]:
```
from sirius.display_tools import listobs_jupyter
listobs_jupyter(vis=save_parms['ms_name'])
```

```
================================================================================
           MeasurementSet Name:  /lustre/cv/users/jsteeb/simulation/het_mosaic_sim.ms
      MS Version 2
================================================================================
   Observer: CASA simulator     Project: CASA simulation
Observation: ALMA(12 antennas)
Data records: 1188       Total elapsed time = 36000 seconds
   Observed from   03-Oct-2020/18:40:49.0   to   04-Oct-2020/04:40:48.9 (UTC)

Fields: 2
  ID   Code Name                RA               Decl           Epoch   SrcId
  nRows
  0          field1              19:59:28.500000 -40.44.01.50000 J2000   0
  594
  1          field2              19:59:28.500000 -40.44.51.50000 J2000   1
  594
Spectral Windows:  (1 unique spectral windows and 1 unique polarization setups)
  SpwID  Name   #Chans   Frame   Ch0(MHz)  ChanWid(kHz)  TotBW(kHz) CtrFreq(MHz)
  Corrs
  0      Band3      5   LSRK   90000.000   2000000.000  10000000.0  94000.0000   XX
  YY
Antennas: 12 'name'='station'
   ID=   0-4: 'N601'='P', 'N606'='P', 'J505'='P', 'J510'='P', 'A001'='P',
   ID=   5-9: 'A012'='P', 'A025'='P', 'A033'='P', 'A045'='P', 'A051'='P',
   ID= 10-11: 'A065'='P', 'A078'='P'
Dish diameter : [ 7.  7.  7.  7. 12. 12. 12. 12. 12. 12. 12. 12.]
Antenna name : ['N601' 'N606' 'J505' 'J510' 'A001' 'A012' 'A025' 'A033' 'A045' 'A051'
 'A065' 'A078']
```

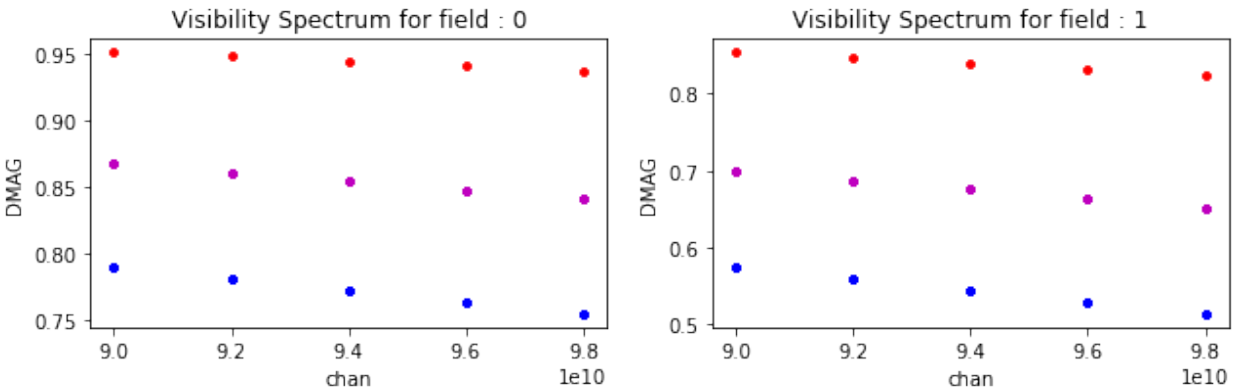
[20]:
```
x_plot(vis=save_parms['ms_name'], ptype='amp-time')
```

```
overwrite_encoded_chunks True
```



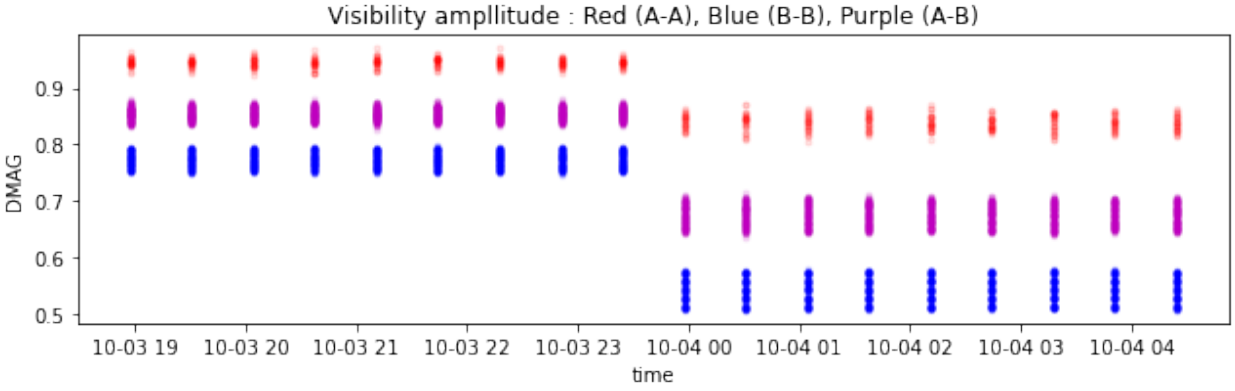
[21]:
```
from sirius.display_tools import x_plot
x_plot(vis=save_parms['ms_name'], ptype='amp-freq')
```

```
overwrite_encoded_chunks True
```



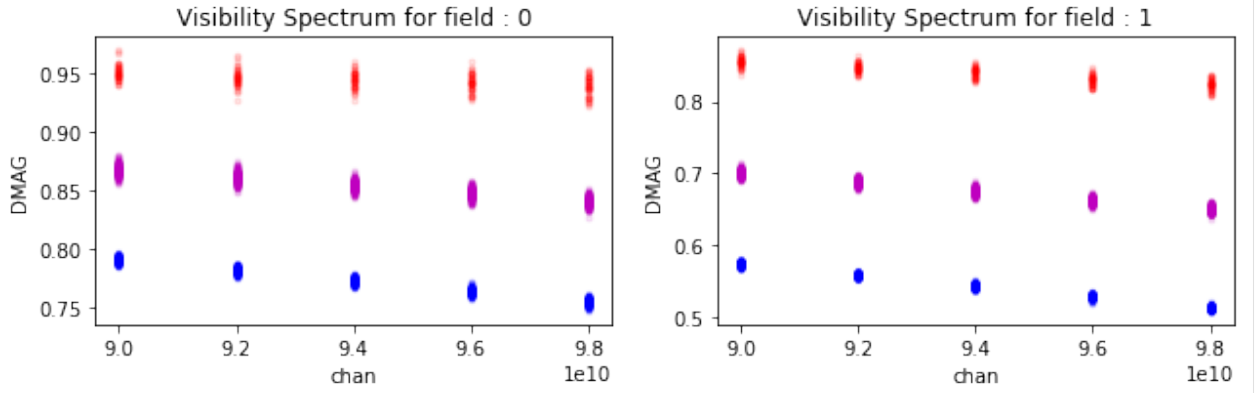
[22]:
```
from sirius.display_tools import x_plot
x_plot(vis=save_parms_noisy['ms_name'], ptype='amp-time')
```

```
Completed ddi 0  process time 4.20 s.ATA...SPECTRAL_WINDOW_ID...
Completed subtables  process time 6.73 s...

overwrite_encoded_chunks True
```



[23]:
```
from sirius.display_tools import x_plot
x_plot(vis=save_parms_noisy['ms_name'], ptype='amp-freq')
```

```
overwrite_encoded_chunks True
```



6.10 Imaging

This will take a while.

[24]:
```
from sirius.display_tools import image_ants

image_ants(vis=save_parms_noisy['ms_name'],imname=working_dir+'try_ALMA', field='0',
↪antsel='A')
image_ants(vis=save_parms_noisy['ms_name'],imname=working_dir+'try_ALMA', field='0',
↪antsel='B')
image_ants(vis=save_parms_noisy['ms_name'],imname=working_dir+'try_ALMA', field='0',
↪antsel='cross')
image_ants(vis=save_parms_noisy['ms_name'],imname=working_dir+'try_ALMA', field='0',
↪antsel='all')
```

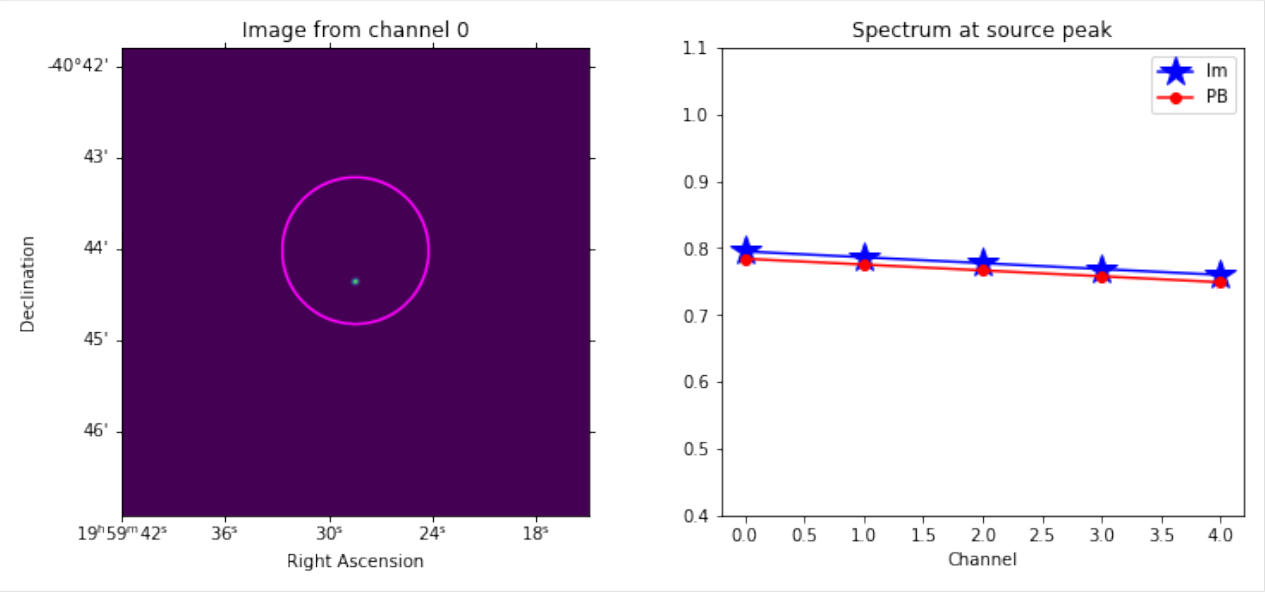
```
antsels[antsel]2  A*&
antsels[antsel]2  J*,N*&
antsels[antsel]2  A* && J*,N*
antsels[antsel]2  *
```

6.10.1 12m-12m

Image only the 12m-12m baselines, with one pointing. The source is located at about the 0.78 gain level of the PB. For this 1 Jy source, the image and PB values should match.

[26]:
```
from sirius.display_tools import display_image
peak_12m,rms_12m = display_image(imname=working_dir+'try_ALMA_A_single.image',
↪pbname=working_dir+'try_ALMA_A_single.pb',resname=working_dir+'try_ALMA_A_single.
↪residual')
```

```
Peak Intensity (chan0) : 0.7950903
PB at location of Intensity peak (chan0) : 0.7841024
max pixel location  (array([512]), array([512]), array([0]), array([0]))
Residual RMS : 0.0002087
```

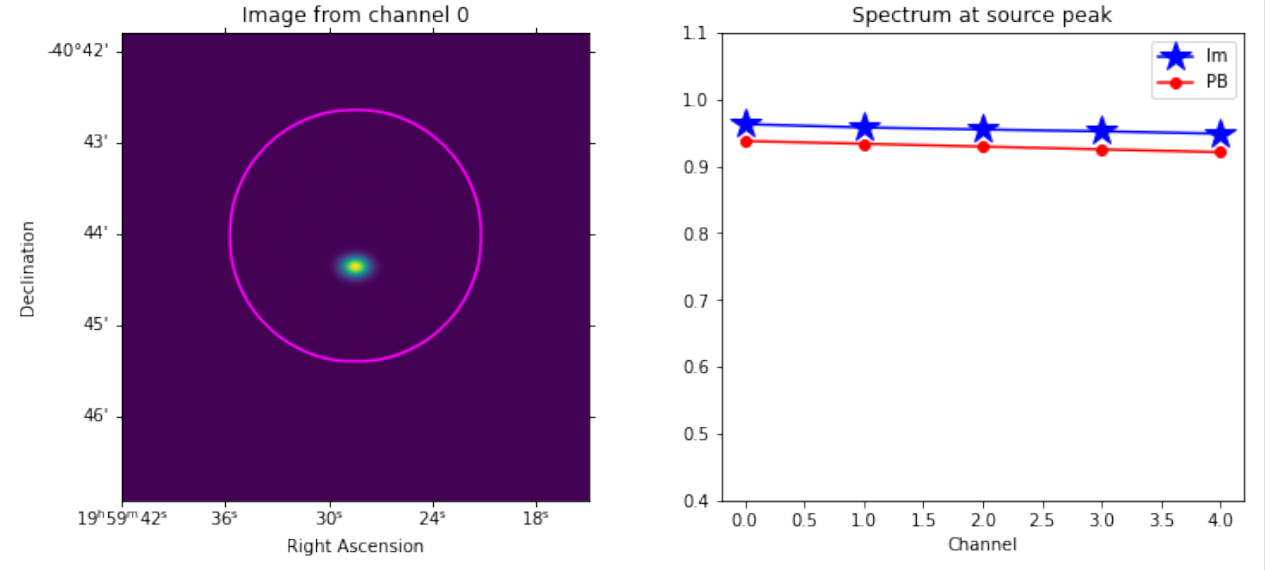


6.10.2 7m-7m

Image only the 7m-7m baselines, with one pointing. The PB is bigger than with the 12m and the source is located at about the 0.92 gain level of the PB. For this 1 Jy source, the image and PB values should match.

[27]:
```
from sirius.display_tools import display_image
peak_7m,rms_7m = display_image(imname=working_dir+'try_ALMA_B_single.image',
↪pbname=working_dir+'try_ALMA_B_single.pb',resname=working_dir+'try_ALMA_A_single.
↪residual')
```

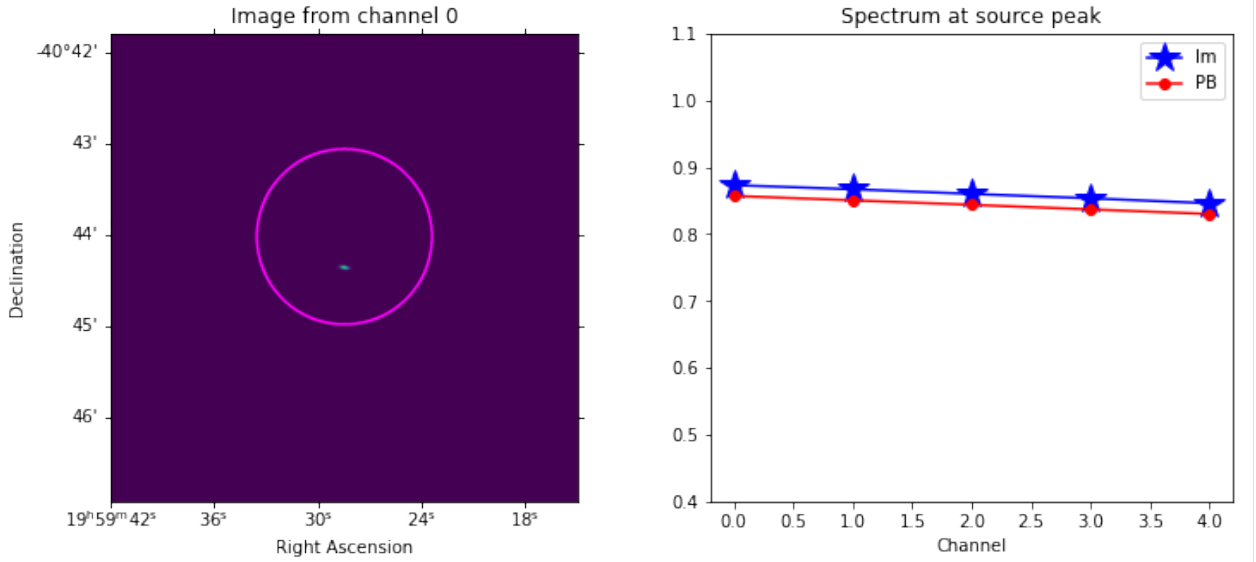
```
Peak Intensity (chan0) : 0.9630291
PB at location of Intensity peak (chan0) : 0.9377817
max pixel location  (array([512]), array([512]), array([0]), array([0]))
Residual RMS : 0.0002087
```



6.10.3 12m-7m

[29]:
```
peak_cross, rms_cross = display_image(imname=working_dir+'try_ALMA_cross_single.image
↪',pbname=working_dir+'try_ALMA_cross_single.pb',resname=working_dir+'try_ALMA_cross_
↪single.residual')
```

```
Peak Intensity (chan0) : 0.8732699
PB at location of Intensity peak (chan0) : 0.8572145
max pixel location  (array([512]), array([512]), array([0]), array([0]))
Residual RMS : 0.0002631
```

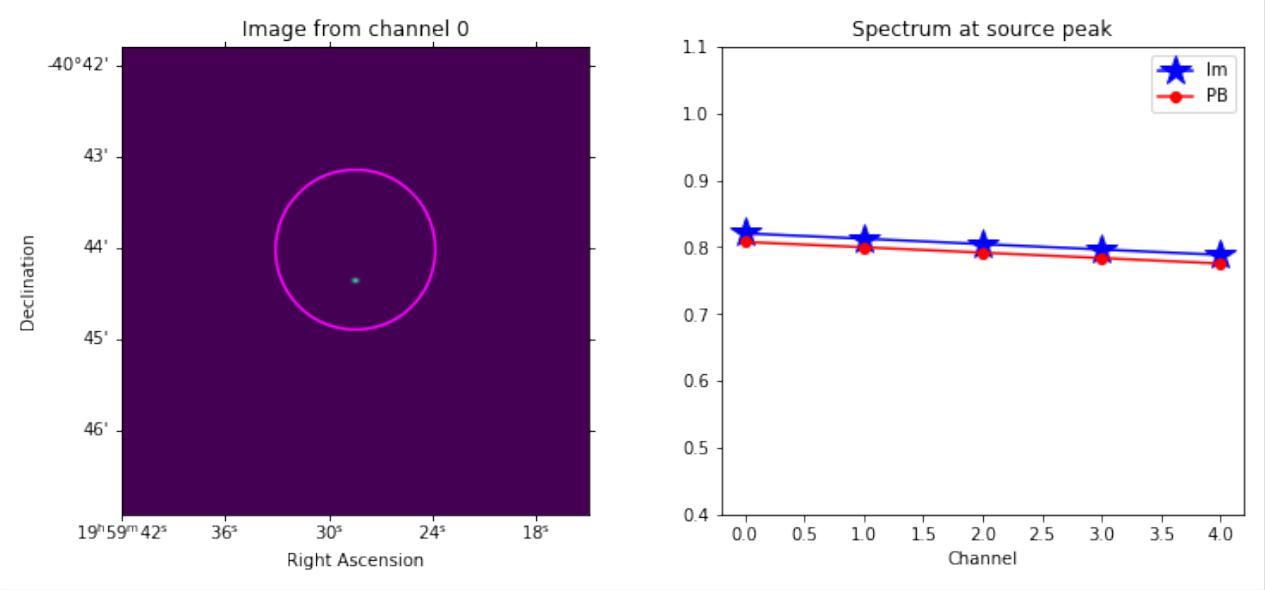


6.10.4 All baselines together

Image all baselines.

[28]:
```
peak_all, rms_all = display_image(imname=working_dir+'try_ALMA_all_single.image',
↪pbname=working_dir+'try_ALMA_all_single.pb',resname=working_dir+'try_ALMA_all_
↪single.residual')
```

```
Peak Intensity (chan0) : 0.8205988
PB at location of Intensity peak (chan0) : 0.8077468
max pixel location  (array([512]), array([512]), array([0]), array([0]))
Residual RMS : 0.0001830
```



6.10.5 Verify the measured intensity, PB and noise levels.

[30]:
```
from sirius.display_tools import get_baseline_types, check_vals
from casatasks import mstransform
 ## Add the WEIGHT_SPECTRUM column. This is needed only if you intend to use task_
↪visstat on this dataset.
os.system('rm -rf het_sim_weight_spectrum.ms')
mstransform(vis=save_parms_noisy['ms_name'],outputvis=working_dir+'het_sim_weight_
↪spectrum.ms',datacolumn='DATA',usewtspectrum=True)


antsels = get_baseline_types()
meas_peaks = {'A':peak_12m, 'B':peak_7m, 'cross':peak_cross, 'all':peak_all}
meas_rms = {'A':rms_12m, 'B':rms_7m, 'cross':rms_cross, 'all':rms_all}
check_vals(vis=working_dir+'het_sim_weight_spectrum.ms',field='0',spw='0:0',
↪antsels=antsels,meas_peaks=meas_peaks, meas_rms=meas_rms)
```

```
<IPython.core.display.HTML object>
```

```
Calculated PB size for type A (dia=12.00) : 0.95493 arcmin
Calculated PB size for type B (dia=7.00) : 1.63702 arcmin
```

6.11 Run the Imaging tests for a Mosaic

The tests below cover joint mosaic imaging.

[31]:
```
image_ants(vis=save_parms_noisy['ms_name'],imname=working_dir+'try_ALMA', field='0,1',
↪antsel='A')
image_ants(vis=save_parms_noisy['ms_name'],imname=working_dir+'try_ALMA', field='0,1',
↪antsel='B')
image_ants(vis=save_parms_noisy['ms_name'],imname=working_dir+'try_ALMA', field='0,1',
↪antsel='cross')
image_ants(vis=save_parms_noisy['ms_name'],imname=working_dir+'try_ALMA', field='0,1',
↪antsel='all')
```

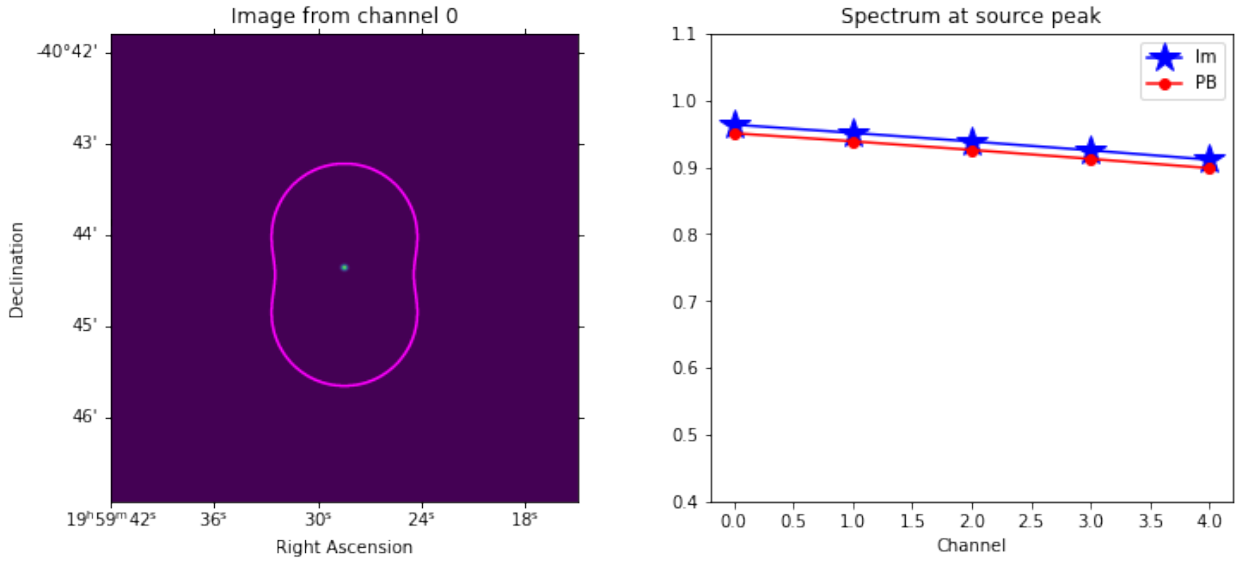
```
antsels[antsel]2  A*&
antsels[antsel]2  J*,N*&
antsels[antsel]2  A* && J*,N*
antsels[antsel]2  *
```

6.11.1 12m-12m

Image only the 12m-12m baselines for a joint mosaic. For this 1 Jy source, the image and the PB values match.

[32]:
```
mospeak_12m,mosrms_12m = display_image(imname=working_dir+'try_ALMA_A_mosaic.image',
↪pbname=working_dir+'try_ALMA_A_mosaic.pb',resname=working_dir+'try_ALMA_A_mosaic.
↪residual')
```

```
Peak Intensity (chan0) : 0.9638311
PB at location of Intensity peak (chan0) : 0.9509934
max pixel location  (array([512]), array([512]), array([0]), array([0]))
Residual RMS : 0.0001933
```

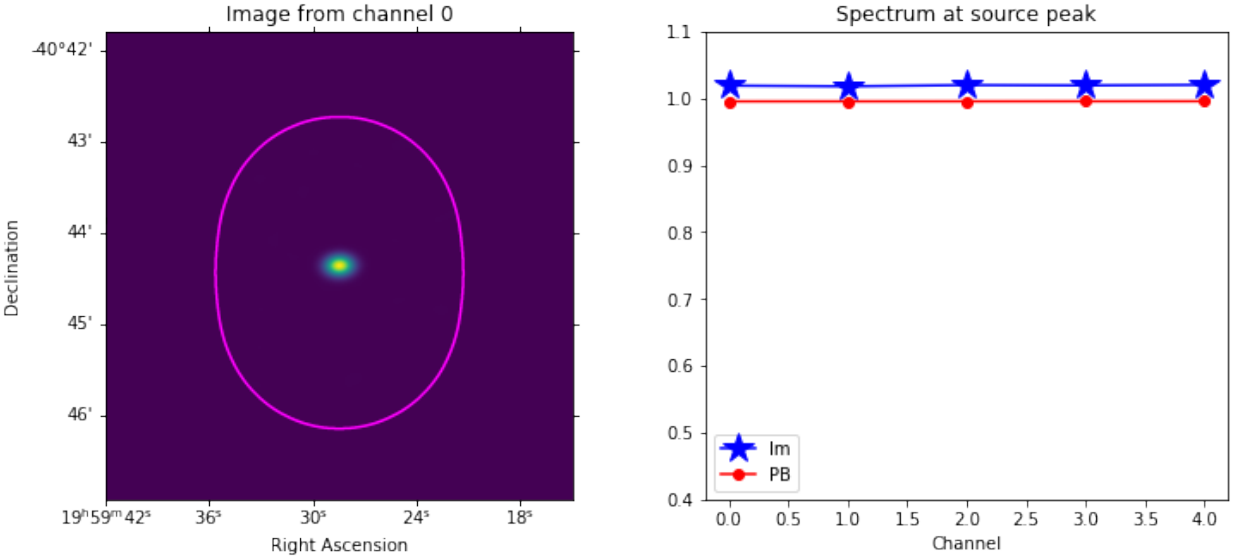


6.11.2 7m-7m

Image only the 7m-7m baselines for a joint mosaic. For this 1 Jy source, the image and the PB values match.

[33]:
```
mospeak_7m,mosrms_7m = display_image(imname=working_dir+'try_ALMA_B_mosaic.image',
↪pbname=working_dir+'try_ALMA_B_mosaic.pb',resname=working_dir+'try_ALMA_B_mosaic.
↪residual')
```

```
Peak Intensity (chan0) : 1.0191978
PB at location of Intensity peak (chan0) : 0.9956154
max pixel location  (array([512]), array([512]), array([0]), array([4]))
Residual RMS : 0.0009187
```

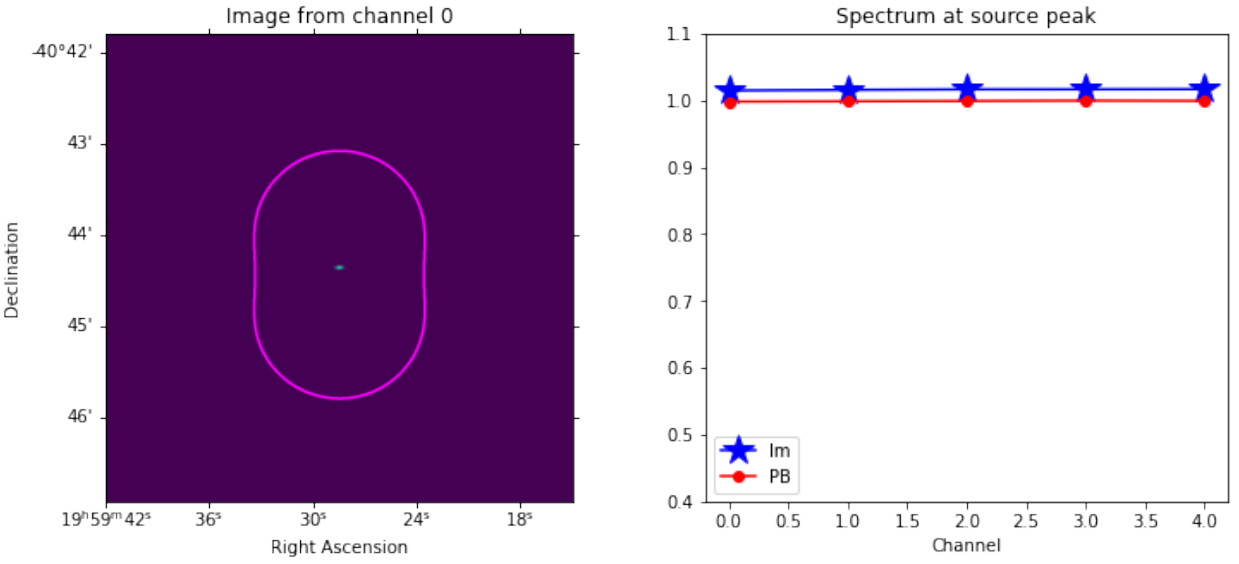


6.11.3 12m-7m

Image the cross baselines and check that the values of Sky, PB and RMS match the predictions.

[34]:
```
mospeak_cross,mosrms_cross = display_image(imname=working_dir+'try_ALMA_cross_mosaic.
↪image',pbname=working_dir+'try_ALMA_cross_mosaic.pb',resname=working_dir+'try_ALMA_
↪cross_mosaic.residual')
```

```
Peak Intensity (chan0) : 1.0148087
PB at location of Intensity peak (chan0) : 0.9982338
max pixel location  (array([512]), array([512]), array([0]), array([4]))
Residual RMS : 0.0002494
```

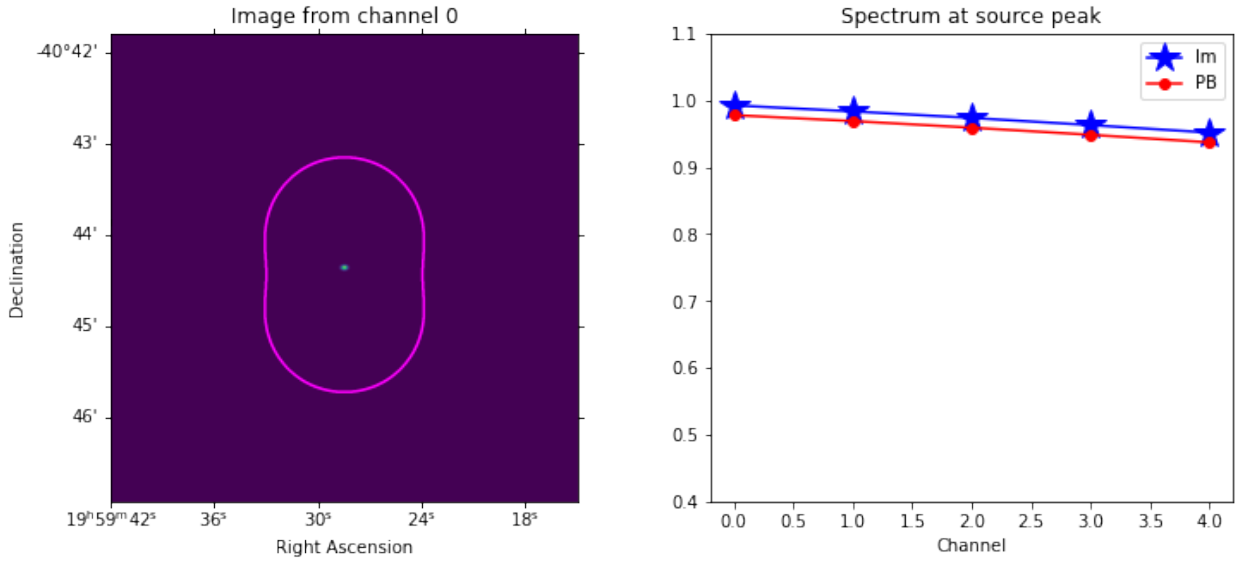


6.11.4 All baselines together

Image all baselines together with a mosaic.

[35]:
```
mospeak_all,mosrms_all = display_image(imname=working_dir+'try_ALMA_all_mosaic.image',
↪pbname=working_dir+'try_ALMA_all_mosaic.pb',resname=working_dir+ 'try_ALMA_all_
↪mosaic.residual')
```

```
Peak Intensity (chan0) : 0.9926163
PB at location of Intensity peak (chan0) : 0.9780214
max pixel location  (array([512]), array([512]), array([0]), array([0]))
Residual RMS : 0.0001611
```



[ ]:

CHAPTER

SEVEN

SINGLE FIELD ALMA SIMULATION

ALMA simulation setup: - Single field (phase center: ra=19h59m28.5s,dec=-40d44m51.5s) - Single point source (1 Jy, ra=19h59m28.5s,dec=-40d44m51.5s) - Frequencies - Time (180 time samples)

7.1 Install SiRIUS and Import Packages

[1]:
```
import os
try:
    import sirius
    print('SiRIUS version',sirius.__version__,'already installed.')
except ImportError as e:
    print(e)
    print('Installing SiRIUS')
    os.system("pip install sirius")
    import sirius
    print('SiRIUS version',sirius.__version__,' installed.')
```

```
SiRIUS version 0.0.28 already installed.
```

[2]:
```
import pkg_resources
import xarray as xr
import numpy as np
from sirius_data import _constants as cnt
from astropy import units as u
from astropy.coordinates import SkyCoord
from sirius.dio import make_time_xda, make_chan_xda
from sirius_data.beam_1d_func_models.airy_disk import aca, alma
from sirius._sirius_utils._coord_transforms import _sin_pixel_to_celestial_coord
from sirius.display_tools import display_image
from casatasks import tclean

xr.set_options(display_style="html")
import os
try:
    from google.colab import output
    output.enable_custom_widget_manager()
    IN_COLAB = True
    %matplotlib widget
except:
    IN_COLAB = False
    %matplotlib inline
```

(continues on next page)

(continued from previous page)

```
# If using uvw_parms['calc_method'] = 'casa' .casarc must have directory of casadata.
import pkg_resources
casa_data_dir = pkg_resources.resource_filename('casadata', '__data__')
rc_file = open(os.path.expanduser("~/.casarc"), "a+")  # append mode
rc_file.write("\n measures.directory: " + casa_data_dir)
rc_file.close()

alma['func'] = 'airy'
alma['dish_diam'] = 10.0
print(alma)

print('complete')
```

```
{'func': 'airy', 'dish_diam': 10.0, 'blockage_diam': 0.75, 'max_rad_1GHz': 0.
↪03113667385557884}
complete
```

7.2 Setup Simulation

[3]:
```
# Image Parameters
#working_dir = '/lustre/cv/users/jsteeb/simulation/'
working_dir = ''
image_size = np.array([1024,1024])
#cell_size = np.array([-0.3,0.3])
cell_size = np.array([-0.15,0.15])
cell = np.array([str(-cell_size[0]) + 'arcsec',str(cell_size[1]) + 'arcsec']) #Used␣
↪in tclean command
cell_size = cell_size*cnt.arcsec_to_rad

# Get telescope layout
tel_dir = pkg_resources.resource_filename('sirius_data', 'telescope_layout/data/alma.
↪all.tel.zarr')
tel_xds = xr.open_zarr(tel_dir,consolidated=False)
dish_diameter = 12.0
tel_xds = tel_xds.where(tel_xds.DISH_DIAMETER == dish_diameter,drop=True) #Only␣
↪select 12m dishes

# Time and Frequency Axis
time_xda = make_time_xda(time_start='2020-10-03T18:57:28.95',time_delta=2000,n_
↪samples=180,n_chunks=1)
chan_xda = make_chan_xda(spw_name = 'Band3',freq_start = 90*10**9, freq_delta =␣
↪2*10**9, freq_resolution=1*10**6, n_channels=5, n_chunks=1)

# Beam model
beam_models = [alma]
beam_model_map = tel_xds.DISH_DIAMETER.values
beam_model_map[beam_model_map == dish_diameter] = 0
beam_model_map= beam_model_map.astype(int) # beam_model_map maps the antenna index to␣
↪a model in beam_models.
beam_parms = {} #Use default beam parms.

# Polarizations
pol = [9,12] # ['RR','RL','LR','LL'] => [5,6,7,8], ['XX','XY','YX','YY'] => [9,10,11,
↪12]
```

(continues on next page)

(continued from previous page)

```
# UVW parameters
uvw_parms = {} # use defaults

# Setup Field
phase_center_skycoord = SkyCoord(ra='19h59m28.5s',dec='-40d44m51.5s',frame='fk5')
phase_center_ra_dec = np.array([phase_center_skycoord.ra.rad,phase_center_skycoord.
↪dec.rad])[None,:] #[n_time, 2] (singleton: n_time)
phase_center_names = np.array(['field1']) #[n_time] (singleton: n_time)

# Setup Point Source
pixel = np.array([512,612])[None,:] # The source will be positioned at the center of␣
↪a pixel.
point_source_ra_dec = _sin_pixel_to_celestial_coord(phase_center_ra_dec[0,:],image_
↪size,cell_size,pixel)[None,:,:] #[n_time, n_point_sources, 2] (singleton: n_time)
point_source_skycoord = SkyCoord(ra=point_source_ra_dec[0,0,0]*u.rad,dec=point_source_
↪ra_dec[0,0,1]*u.rad,frame='fk5')
print('The position of the source is ', point_source_skycoord.ra.hms,point_source_
↪skycoord.dec.dms)
point_source_flux = np.array([1.0, 0, 0, 1.0])[None,None,None,:]  #[n_point_sources,␣
↪n_time, n_chan, n_pol] (singleton: n_time, n_chan)

# Pointing Offsets
pointing_ra_dec = None #No pointing offsets

# Noise parameters
noise_parms = None
```

```
Number of chunks  1
Number of chunks  1
The position of the source is  hms_tuple(h=19.0, m=59.0, s=28.50000000000307) dms_
↪tuple(d=-40.0, m=-44.0, s=-36.49999998675412)
```

7.3 Run Simulation

[4]:
```
from sirius import simulation
save_parms = {'ms_name':working_dir+'alma_sim.ms','write_to_ms':True}
vis_xds = simulation(point_source_flux,
                     point_source_ra_dec,
                     pointing_ra_dec,
                     phase_center_ra_dec,
                     phase_center_names,
                     beam_parms,beam_models,
                     beam_model_map,uvw_parms,
                     tel_xds, time_xda, chan_xda, pol, noise_parms, save_parms)
vis_xds
```

```
Setting default calc_method  to  astropy
Setting default auto_corr  to  False
Setting default fov_scaling  to  4.0
Setting default mueller_selection  to  [ 0  5 10 15]
Setting default zernike_freq_interp  to  nearest
Setting default pa_radius  to  0.2
```

(continues on next page)

(continued from previous page)

```
Setting default image_size  to  [1000 1000]
Setting default mode  to  dask_ms_and_sim_tool
Setting default DAG_name_vis_uvw_gen  to  False
Setting default DAG_name_write  to  False
180 15225 5 2
<xarray.Dataset>
Dimensions:  (time: 180, pol: 2, chan: 5, baseline: 15225, uvw: 3,
              time_chunk: 1, chan_chunk: 1, 4: 4)
Coordinates:
  * time     (time) <U23 '2020-10-03T18:57:28.950' ... '2020-10-07T22:24:08.950'
  * pol      (pol) int64 9 12
  * chan     (chan) float64 9e+10 9.2e+10 9.4e+10 9.6e+10 9.8e+10
Dimensions without coordinates: baseline, uvw, time_chunk, chan_chunk, 4
Data variables:
    DATA     (time, baseline, chan, pol) complex128 dask.array<chunksize=(180, 15225,␣
↪5, 2), meta=np.ndarray>
    UVW      (time, baseline, uvw) complex128 dask.array<chunksize=(180, 15225, 3),␣
↪meta=np.ndarray>
    WEIGHT   (time, baseline, pol) float64 dask.array<chunksize=(180, 15225, 2),␣
↪meta=np.ndarray>
    SIGMA    (time, baseline, pol) float64 dask.array<chunksize=(180, 15225, 2),␣
↪meta=np.ndarray>
    TIMING   (time_chunk, chan_chunk, 4) float64 dask.array<chunksize=(1, 1, 4),␣
↪meta=np.ndarray>
Meta data creation  25.977193355560303
reshape time  0.0007262229919433594
*** Dask compute time 13.41493558883667
```

[4]:
```
<xarray.Dataset>
Dimensions:           (polarization_ids: 1, spw_ids: 1)
Coordinates:
  * polarization_ids  (polarization_ids) int64 0
  * spw_ids           (spw_ids) int64 0
Data variables:
    *empty*
Attributes:
    xds0:              <xarray.Dataset>\nDimensions:         (row: 2740500, u...
    SPECTRAL_WINDOW:   <xarray.Dataset>\nDimensions:          (row: 1, d1: 5)...
    POLARIZATION:      <xarray.Dataset>\nDimensions:       (row: 1, d1: 2, d2...
    DATA_DESCRIPTION:  <xarray.Dataset>\nDimensions:             (row: 1)\nCo...
```

7.4 Imaging niter = 0

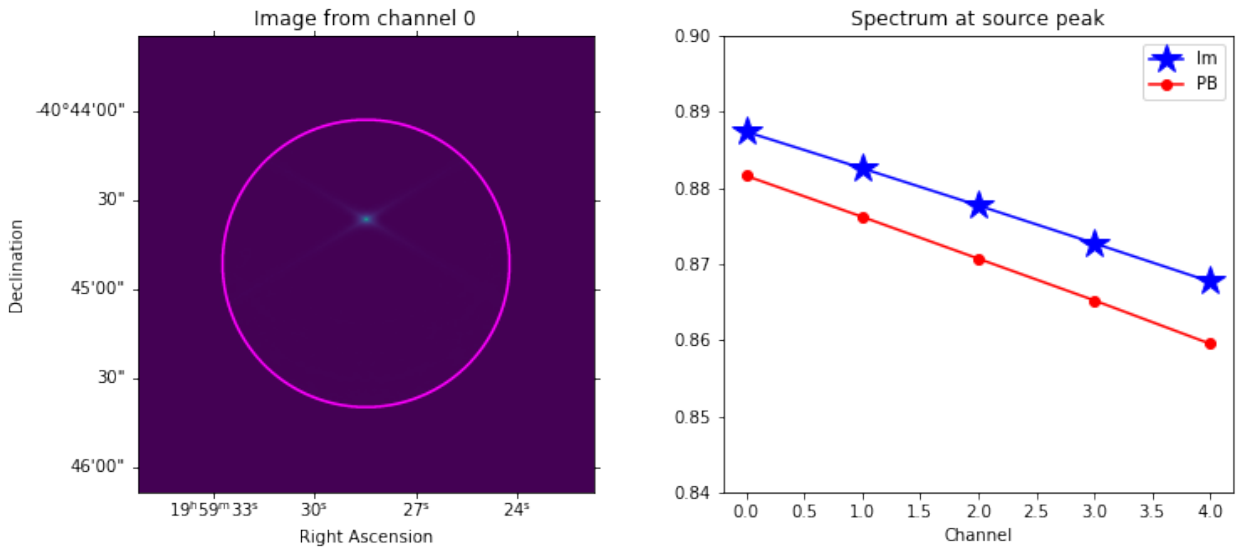
[5]:
```
image_name = working_dir+'im_alma_sim'
os.system('rm -rf '+image_name+'_niter0_*')

tclean(vis=save_parms['ms_name'],imagename=image_name+'_niter0_std',imsize=image_size,
↪cell=cell,specmode='cube',niter=0,pblimit=0.2,pbmask=0.2,gridder='standard',stokes=
↪'XX',weighting='natural',datacolumn='data')

tclean(vis=save_parms['ms_name'],imagename=image_name+'_niter0_m',imsize=image_size,
↪cell=cell,specmode='cube',niter=0,pblimit=0.2,pbmask=0.2,gridder='mosaic',stokes='XX
↪',weighting='natural',datacolumn='data')
```

```
0%...10...20...30...40...50...60...70...80...90...100%

0%...10...20...30...40...50...60...70...80...90...100%
2022-03-16 17:23:19     WARN    task_tclean::SIImageStore::restore (file src/code/
↪synthesis/ImagerObjects/SIImageStore.cc, line 2246)  Restoring with an empty model␣
↪image. Only residuals will be processed to form the output restored image.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.

0%...10...20...30...40...50...60...70...80...90...100%
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.

0%...10...20...30...40...50...60...70...80...90...100%
2022-03-16 17:23:56     WARN    task_tclean::SIImageStore::restore (file src/code/
↪synthesis/ImagerObjects/SIImageStore.cc, line 2246)  Restoring with an empty model␣
↪image. Only residuals will be processed to form the output restored image.
```

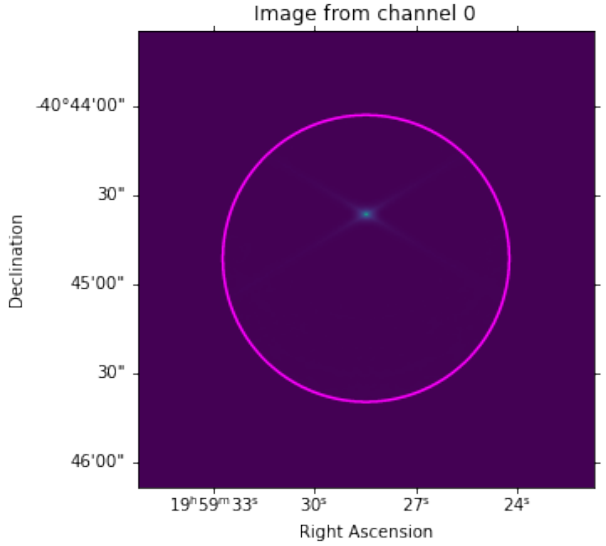
```
[5]: {}
```

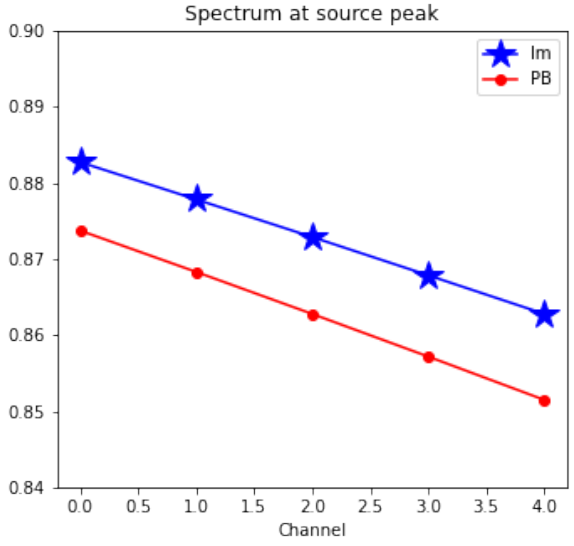
[7]:
```
chan=0
print('Abs of visibility for channel ' , chan , ':', abs(np.abs(vis_xds.xds0.DATA[0,
↪chan,0].values)))
print('#'*100)
display_image(imname=image_name+'_niter0_std.image',pbname=image_name+'_niter0_std.pb
↪',resname=image_name+'_niter0_std.residual',ylim=[0.84,0.9],chan=chan)
print('#'*100)
display_image(imname=image_name+'_niter0_m.image',pbname=image_name+'_niter0_m.pb',
↪resname=image_name+'_niter0_m.residual',ylim=[0.84,0.9],chan=chan)
```

```
Abs of visibility for channel  0 : 0.8873939
####################################################################################
↪###############
Peak Intensity (chan0) : 0.8873690
PB at location of Intensity peak (chan0) : 0.8815771
max pixel location  (array([512]), array([612]), array([0]), array([0]))
Residual RMS : 0.0100675
```




```
####################################################################################
↪###############
Peak Intensity (chan0) : 0.8827165
PB at location of Intensity peak (chan0) : 0.8737198
max pixel location  (array([512]), array([612]), array([0]), array([0]))
Residual RMS : 0.0100281
```





[7]:
```
(0.8827165365219116, 0.01002805296120395)
```

7.5 Imaging niter = 1000

[8]:
```
niter = 1000
os.system('rm -rf '+image_name+'_niter'+str(niter)+'_*')
tclean(vis=save_parms['ms_name'],imagename=image_name+'_niter'+str(niter)+'_std',
↪imsize=image_size,cell=cell,specmode='cube',niter=niter,pblimit=0.2,pbmask=0.2,
↪gridder='standard',stokes='XX',weighting='natural',datacolumn='data')

tclean(vis=save_parms['ms_name'],imagename=image_name+'_niter'+str(niter)+'_m',
↪imsize=image_size,cell=cell,specmode='cube',niter=niter,pblimit=0.2,pbmask=0.2,
↪gridder='mosaic',stokes='XX',weighting='natural',datacolumn='data')
```

```
0%...10...20...30...40...50...60...70...80...90...100%

0%...10...20...30...40...50...60...70...80...90...100%

0%...10...20...30...40...50...60...70...80...90...100%

0%...10...20...30...40...50...60...70...80...90...100%

0%...10...20...30...40...50...60...70...80...90...100%

0%...10...20...30...40...50...60...70...80...90...100%

0%...10...20...30...40...50...60...70...80...90...100%

0%...10...20...30...40...50...60...70...80...90...100%

0%...10...20...30...40...50...60...70...80...90...100%
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
```

(continues on next page)

(continued from previous page)

```
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.

0%...10...20...30...40...50...60...70...80...90...100%
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.

0%...10...20...30...40...50...60...70...80...90...100%
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
```

(continues on next page)

(continued from previous page)

```
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.

0%...10...20...30...40...50...60...70...80...90...100%
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.

0%...10...20...30...40...50...60...70...80...90...100%
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.

0%...10...20...30...40...50...60...70...80...90...100%
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
```

(continues on next page)

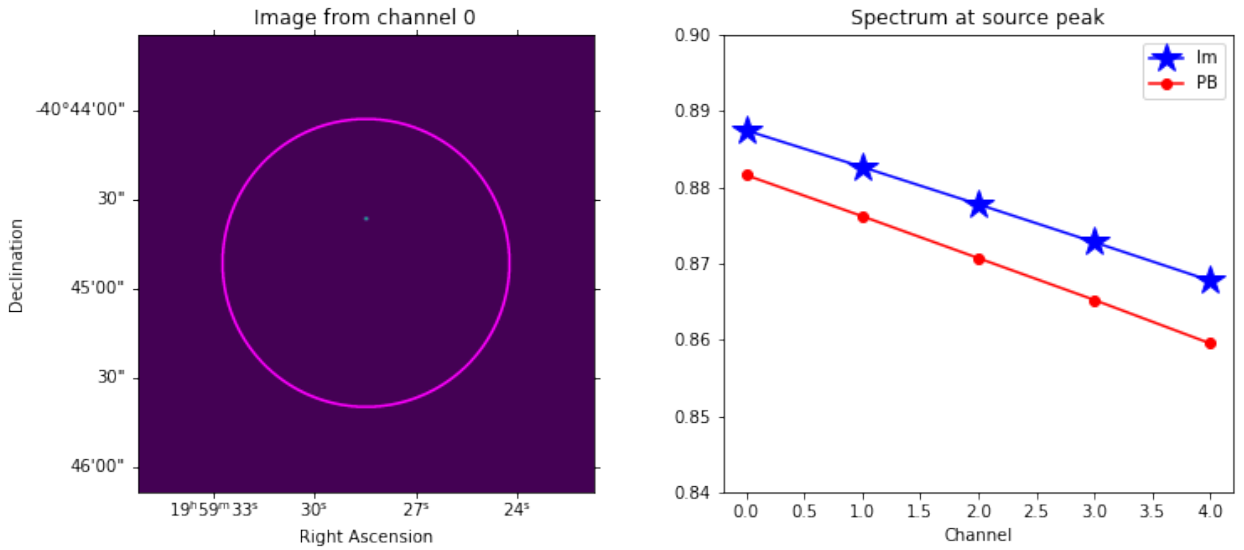
(continued from previous page)

```
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.

0%...10...20...30...40...50...60...70...80...90...100%
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.

0%...10...20...30...40...50...60...70...80...90...100%
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.

0%...10...20...30...40...50...60...70...80...90...100%
```

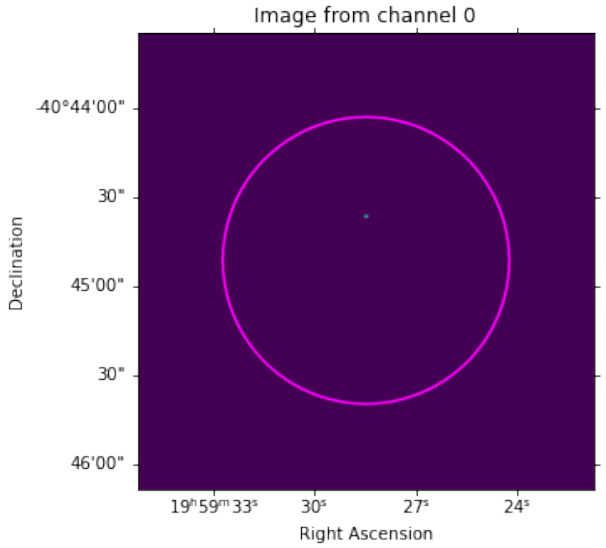
[8]:
```
{}
```

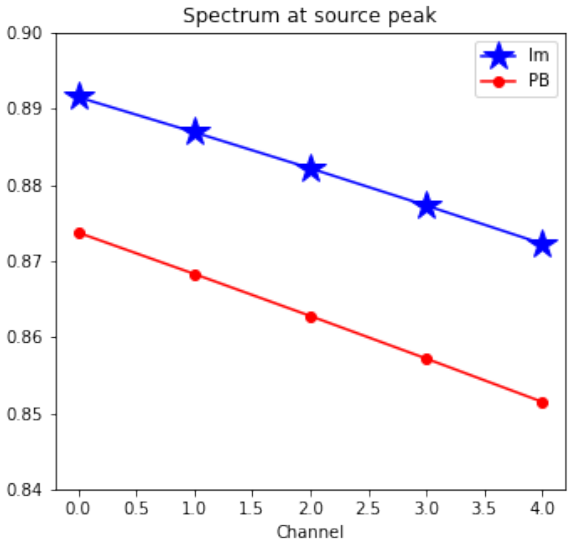
[9]:
```
chan=0
print('Abs of visibility for channel ' , chan , ':', abs(np.abs(vis_xds.xds0.DATA[0,
↪chan,0].values)))
print('#'*100)
display_image(imname=image_name+'_niter'+str(niter)+'_std.image',pbname=image_name+'_
↪niter'+str(niter)+'_std.pb',resname=image_name+'_niter'+str(niter)+'_std.residual',
↪ylim=[0.84,0.9],chan=chan)
print('#'*100)
display_image(imname=image_name+'_niter'+str(niter)+'_m.image',pbname=image_name+'_
↪niter'+str(niter)+'_m.pb',resname=image_name+'_niter'+str(niter)+'_m.residual',
↪ylim=[0.84,0.9],chan=chan)
```

```
Abs of visibility for channel  0 : 0.8873939
####################################################################################
↪###############
Peak Intensity (chan0) : 0.8874448
PB at location of Intensity peak (chan0) : 0.8815771
max pixel location  (array([512]), array([612]), array([0]), array([0]))
Residual RMS : 0.0000041
```




```
####################################################################################
↪###############
Peak Intensity (chan0) : 0.8915214
PB at location of Intensity peak (chan0) : 0.8737198
max pixel location  (array([512]), array([612]), array([0]), array([0]))
Residual RMS : 0.0000291
```





[9]:
```
(0.8915213942527771, 2.9058321458074734e-05)
```

7.6 Increase number of Sources

[10]:
```
# Setup Point Source
pixel = np.array([[512,612],[512,512],[650,650],[450,473]])
point_source_ra_dec = _sin_pixel_to_celestial_coord(phase_center_ra_dec[0,:],image_
↪size,cell_size,pixel)[None,:,:] #[n_time, n_point_sources, 2] (singleton: n_time)
point_source_skycoord = SkyCoord(ra=point_source_ra_dec[0,0,0]*u.rad,dec=point_source_
↪ra_dec[0,0,1]*u.rad,frame='fk5')
print('The position of the source is ', point_source_skycoord.ra.hms,point_source_
↪skycoord.dec.dms)
point_source_flux = np.array([[1.0, 0, 0, 1.0],[0.5, 0, 0, 0.5],[0.65, 0, 0, 0.65],[0.
↪34, 0, 0, 0.34]])[:,None,None,:]  #[n_point_sources, n_time, n_chan, n_pol]␣
↪(singleton: n_time, n_chan)

from sirius import simulation
save_parms = {'ms_name':working_dir+'msource_alma_sim.ms','write_to_ms':True}
vis_xds = simulation(point_source_flux,
                     point_source_ra_dec,
                     pointing_ra_dec,
                     phase_center_ra_dec,
                     phase_center_names,
                     beam_parms,beam_models,
                     beam_model_map,uvw_parms,
                     tel_xds, time_xda, chan_xda, pol, noise_parms, save_parms)
vis_xds
```

```
The position of the source is  hms_tuple(h=19.0, m=59.0, s=28.50000000000307) dms_
↪tuple(d=-40.0, m=-44.0, s=-36.49999998675412)
Setting default calc_method  to  astropy
Setting default auto_corr  to  False
Setting default fov_scaling  to  4.0
Setting default mueller_selection  to  [ 0  5 10 15]
```

(continues on next page)

(continued from previous page)

```
Setting default zernike_freq_interp  to  nearest
Setting default pa_radius  to  0.2
Setting default image_size  to  [1000 1000]
Setting default mode  to  dask_ms_and_sim_tool
Setting default DAG_name_vis_uvw_gen  to  False
Setting default DAG_name_write  to  False
180 15225 5 2
<xarray.Dataset>
Dimensions:  (time: 180, pol: 2, chan: 5, baseline: 15225, uvw: 3,
              time_chunk: 1, chan_chunk: 1, 4: 4)
Coordinates:
  * time     (time) <U23 '2020-10-03T18:57:28.950' ... '2020-10-07T22:24:08.950'
  * pol      (pol) int64 9 12
  * chan     (chan) float64 9e+10 9.2e+10 9.4e+10 9.6e+10 9.8e+10
Dimensions without coordinates: baseline, uvw, time_chunk, chan_chunk, 4
Data variables:
    DATA     (time, baseline, chan, pol) complex128 dask.array<chunksize=(180, 15225,␣
↪5, 2), meta=np.ndarray>
    UVW      (time, baseline, uvw) complex128 dask.array<chunksize=(180, 15225, 3),␣
↪meta=np.ndarray>
    WEIGHT   (time, baseline, pol) float64 dask.array<chunksize=(180, 15225, 2),␣
↪meta=np.ndarray>
    SIGMA    (time, baseline, pol) float64 dask.array<chunksize=(180, 15225, 2),␣
↪meta=np.ndarray>
    TIMING   (time_chunk, chan_chunk, 4) float64 dask.array<chunksize=(1, 1, 4),␣
↪meta=np.ndarray>
Meta data creation  25.98868727684021
reshape time  0.0008723735809326172
*** Dask compute time 33.98268532752991
```

[10]:
```
<xarray.Dataset>
Dimensions:           (polarization_ids: 1, spw_ids: 1)
Coordinates:
  * polarization_ids  (polarization_ids) int64 0
  * spw_ids           (spw_ids) int64 0
Data variables:
    *empty*
Attributes:
    xds0:              <xarray.Dataset>\nDimensions:         (row: 2740500, u...
    SPECTRAL_WINDOW:   <xarray.Dataset>\nDimensions:          (row: 1, d1: 5)...
    POLARIZATION:      <xarray.Dataset>\nDimensions:       (row: 1, d1: 2, d2...
    DATA_DESCRIPTION:  <xarray.Dataset>\nDimensions:             (row: 1)\nCo...
```

[11]:
```
image_name = working_dir+'im_alma_sim'
os.system('rm -rf '+image_name+'_niter0_*')

tclean(vis=save_parms['ms_name'],imagename=image_name+'_niter0_std',imsize=image_size,
↪cell=cell,specmode='cube',niter=0,pblimit=0.2,pbmask=0.2,gridder='standard',stokes=
↪'XX',weighting='natural',datacolumn='data')

tclean(vis=save_parms['ms_name'],imagename=image_name+'_niter0_m',imsize=image_size,
↪cell=cell,specmode='cube',niter=0,pblimit=0.2,pbmask=0.2,gridder='mosaic',stokes='XX
↪',weighting='natural',datacolumn='data')
```

```
0%...10...20...30...40...50...60...70...80...90...100%
```

(continues on next page)

(continued from previous page)

```
0%...10...20...30...40...50...60...70...80...90...100%
2022-03-16 17:42:27     WARN    task_tclean::SIImageStore::restore (file src/code/
↪synthesis/ImagerObjects/SIImageStore.cc, line 2246)  Restoring with an empty model␣
↪image. Only residuals will be processed to form the output restored image.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.

0%...10...20...30...40...50...60...70...80...90...100%
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.

0%...10...20...30...40...50...60...70...80...90...100%
2022-03-16 17:43:04     WARN    task_tclean::SIImageStore::restore (file src/code/
↪synthesis/ImagerObjects/SIImageStore.cc, line 2246)  Restoring with an empty model␣
↪image. Only residuals will be processed to form the output restored image.
```

[11]:
```
{}
```

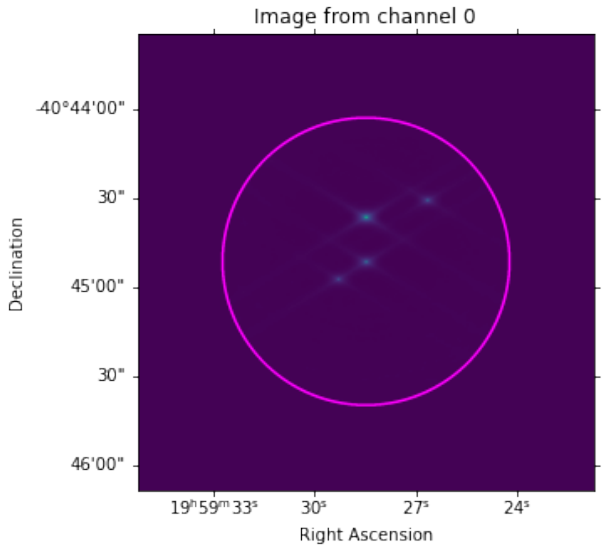
[12]:
```
chan=0
print('Abs of visibility for channel ' , chan , ':', abs(np.abs(vis_xds.xds0.DATA[0,
↪chan,0].values)))
```

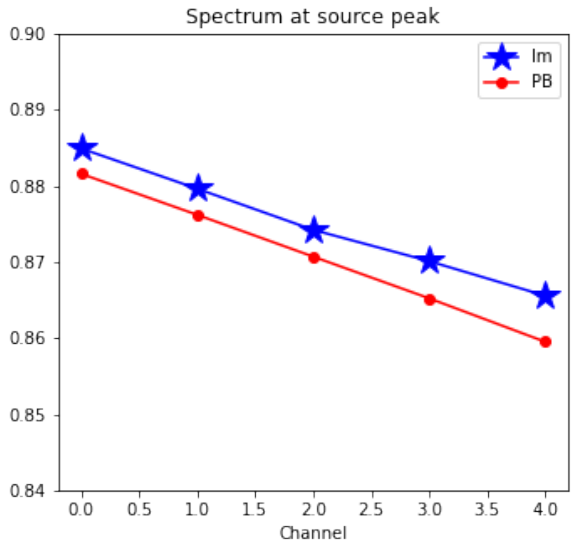
(continues on next page)

(continued from previous page)

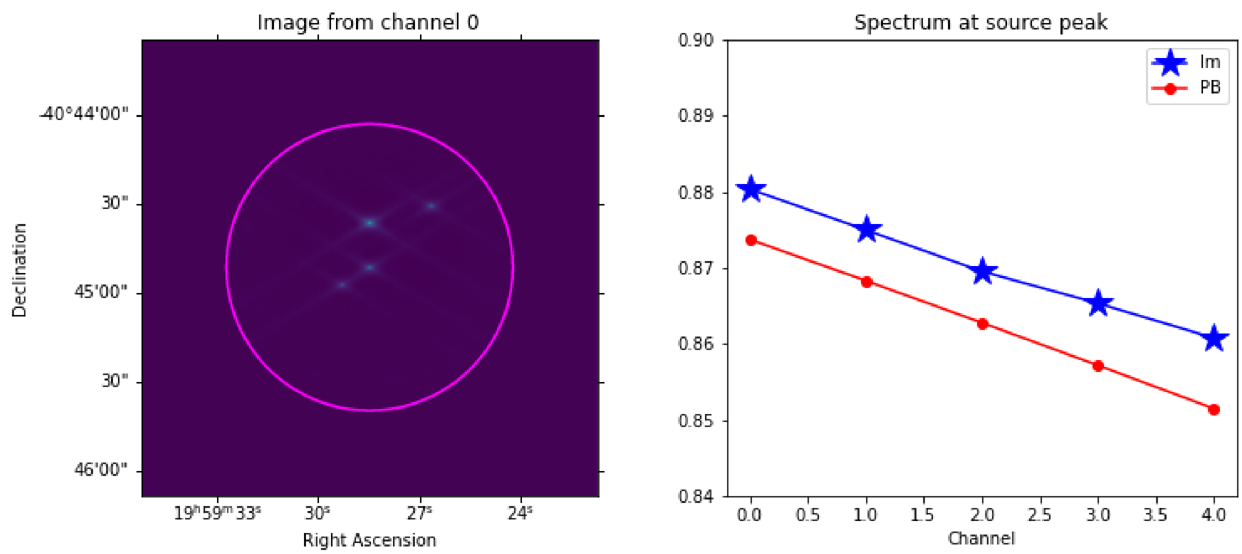
```
print('#'*100)
display_image(imname=image_name+'_niter0_std.image',pbname=image_name+'_niter0_std.pb
↪',resname=image_name+'_niter0_std.residual',ylim=[0.84,0.9],chan=chan)
print('#'*100)
display_image(imname=image_name+'_niter0_m.image',pbname=image_name+'_niter0_m.pb',
↪resname=image_name+'_niter0_m.residual',ylim=[0.84,0.9],chan=chan)
```

```
Abs of visibility for channel  0 : 1.3650753
##########################################################################################
↪###############
Peak Intensity (chan0) : 0.8849095
PB at location of Intensity peak (chan0) : 0.8815771
max pixel location  (array([512]), array([612]), array([0]), array([0]))
Residual RMS : 0.0128337
```






```
##########################################################################################
↪###############
Peak Intensity (chan0) : 0.8803405
PB at location of Intensity peak (chan0) : 0.8737198
max pixel location  (array([512]), array([612]), array([0]), array([0]))
Residual RMS : 0.0127897
```



[12]:
```
(0.8803405165672302, 0.012789708181056035)
```

[13]:
```
niter = 1000
os.system('rm -rf '+image_name+'_niter'+str(niter)+'_*')
tclean(vis=save_parms['ms_name'],imagename=image_name+'_niter'+str(niter)+'_std',
↪imsize=image_size,cell=cell,specmode='cube',niter=niter,pblimit=0.2,pbmask=0.2,
↪gridder='standard',stokes='XX',weighting='natural',datacolumn='data')

tclean(vis=save_parms['ms_name'],imagename=image_name+'_niter'+str(niter)+'_m',
↪imsize=image_size,cell=cell,specmode='cube',niter=niter,pblimit=0.2,pbmask=0.2,
↪gridder='mosaic',stokes='XX',weighting='natural',datacolumn='data')
```

```
0%...10...20...30...40...50...60...70...80...90...100%

0%...10...20...30...40...50...60...70...80...90...100%

0%...10...20...30...40...50...60...70...80...90...100%

0%...10...20...30...40...50...60...70...80...90...100%

0%...10...20...30...40...50...60...70...80...90...100%

0%...10...20...30...40...50...60...70...80...90...100%
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
```

(continues on next page)

(continued from previous page)

```
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.

0%...10...20...30...40...50...60...70...80...90...100%
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.

0%...10...20...30...40...50...60...70...80...90...100%
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.

0%...10...20...30...40...50...60...70...80...90...100%
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
```

(continues on next page)

(continued from previous page)

```
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.

0%...10...20...30...40...50...60...70...80...90...100%
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.

0%...10...20...30...40...50...60...70...80...90...100%
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
```

(continues on next page)

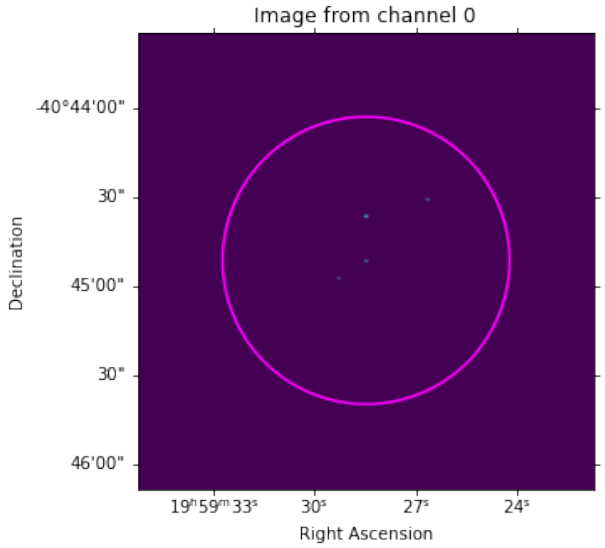
(continued from previous page)

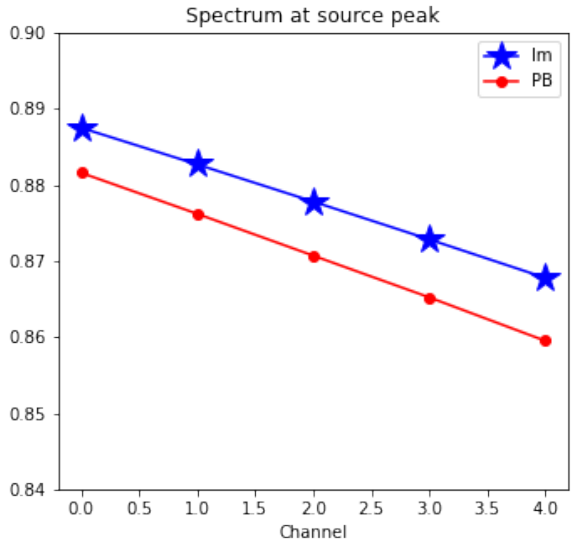
```
0%...10...20...30...40...50...60...70...80...90...100%
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.
OMP: Info #276: omp_set_nested routine deprecated, please use omp_set_max_active_
↪levels instead.

0%...10...20...30...40...50...60...70...80...90...100%
```

[13]:
```
{}
```

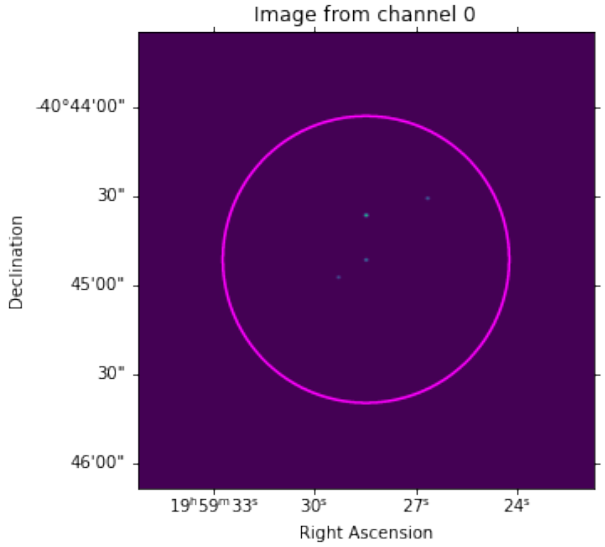
[14]:
```
chan=0
#niter=1000
print('Abs of visibility for channel ' , chan , ':', abs(np.abs(vis_xds.xds0.DATA[0,
↪chan,0].values)))
print('#'*100)
display_image(imname=image_name+'_niter'+str(niter)+'_std.image',pbname=image_name+'_
↪niter'+str(niter)+'_std.pb',resname=image_name+'_niter'+str(niter)+'_std.residual',
↪ylim=[0.84,0.9],chan=chan)
print('#'*100)
display_image(imname=image_name+'_niter'+str(niter)+'_m.image',pbname=image_name+'_
↪niter'+str(niter)+'_m.pb',resname=image_name+'_niter'+str(niter)+'_m.residual',
↪ylim=[0.84,0.9],chan=chan)
```

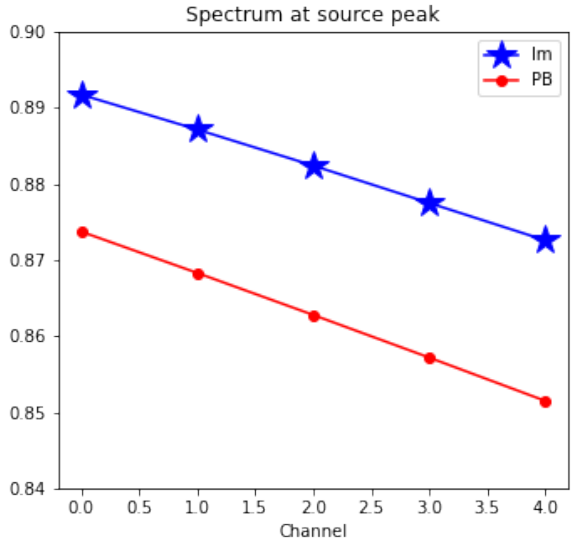
```
Abs of visibility for channel  0 : 1.3650753
######################################################################################
↪##############
Peak Intensity (chan0) : 0.8874919
PB at location of Intensity peak (chan0) : 0.8815771
max pixel location  (array([512]), array([612]), array([0]), array([0]))
Residual RMS : 0.0000202
```






```
################################################################################################
↪###############
Peak Intensity (chan0) : 0.8917176
PB at location of Intensity peak (chan0) : 0.8737198
max pixel location  (array([512]), array([612]), array([0]), array([0]))
Residual RMS : 0.0000386
```





[14]:
```
(0.8917175531387329, 3.856849138287457e-05)
```

CHAPTER

EIGHT

ABOUT THIS PROJECT

Under development. Questions and feedback can be sent to: **jsteeb@nrao.edu**

PYTHON MODULE INDEX

S

INDEX