OSKAR MATLAB Interface

Version history:

Revision	Date	Modification
1	2013-03-04	Created from the description of the OSKAR MATLAB interface previously in the OSKAR Applications document.
2	2013-03-05	Fixed typos in the syntax for four commands.
3	2013-03-11	Added a number of comments addressing a number of issues of building and running the OSKAR MATLAB interface.
4	2014-07-16	Updated installation path. Fixed name of “vis” package. Added section describing “convert” package, and moved some existing functions here from “sky”.

1 Introduction

This document briefly summarises the **experimental** MATLAB interface functionality provided by OSKAR. It assumes that OSKAR has already been built and installed with the MATLAB build dependency satisfied. Please note that if MATLAB is updated, the OSKAR installation procedure should be repeated to ensure that the version of OSKAR is linked against the correct MATLAB libraries.

2 Configuring MATLAB

The OSKAR MATLAB toolbox is a **very experimental** set of MATLAB functions for interfacing with OSKAR data products in a MATLAB workspace.

To import this toolbox into MATLAB all that is required is to add the OSKAR toolbox directory to the MATLAB path. By default this toolbox is installed in the following location:

```
/usr/local/share/oskar/matlab
```

Adding a new MATLAB path can be achieved by using the **Set Path…** option in the **File** menu of the MATAB interface, or by using the `addPath` command from a MATLAB command window.

Once the OSKAR toolbox has been successfully added, the MATLAB command

```
>> what oskar
```

can be used to list the packages which are available.

In general, functions in the OSKAR toolbox can be run with the following syntax:

```
>> return_values = oskar.package.function(arguments)
```

2.1 Known Issues

2.1.1 Compatibility with MATLAB versions of libstdc++ & libgfortran

On some Linux systems, including Ubuntu, the versions of libstdc++ and libgfortan provided with MATLAB are incompatible to the versions linked by the OSKAR MATLAB interface.

As a temporary work-around, we suggest that the MATLAB installation is modified by updating the symbolic links found in `<MATLAB_ROOT>/sys/os/glnxa64`, for 64-bit Linux installations (or `<MATLAB_ROOT>/sys/os/glnx86`, for 32-bit Linux installations), to the equivalent system installed versions of these libraries. If necessary, this work-around can be easily reverted by changing the symbolic links back to their original targets in the MATLAB installation.

3 OSKAR MATLAB Functions

The OSKAR toolbox consists of six packages, each containing one or more functions and/or classes, listed in the following sections. Usage information for a function can be obtained from within MATLAB by running the function with no arguments.

3.1 oskar.binary_file.*: Functions for reading OSKAR binary files

`[index, headers] = oskar.binary_file.query(<filename>)`

Function to query the contents of an OSKAR binary file, returning a MATLAB cell array index table (in the variable index) describing the contents of the file, as well as an array of structures for each of the headers in the binary file.

`[record] = oskar.binary_file.read_record(<filename>, <group>, <tag>, [index = 0])`

Function to read a specified record from an OSKAR binary file, returning the record as a MATLAB structure.

`[records] = oskar.binary_file.read_group(<filename>, <group>)`

Function to read a group of records from an OSKAR binary file. The group is returned as an array of record structures.

`[records] = oskar.binary_file.read_all(<filename>)`

Function to read the contents of an OSKAR binary file and return an array of record structures.

3.2 oskar.cuda.*: Functions for obtaining CUDA system information

`[info] = oskar.cuda.device_info()`

Function to return a structure of basic information on the current state of CUDA devices.

3.3 oskar.image.*: Functions handling OSKAR images

`oskar.image.type (`*`enumeration class`*`)`

Enumeration class used for describing the image type.

`oskar.image.transform (`*`enumeration class`*`)`

Enumeration class used for describing the image transform used when making the image.

`[image] = oskar.image.read(<filename>, [index = 0])`

Function to read an OSKAR binary image file into a MATLAB structure. If more than one image is present in the binary file, the optional index argument can be used to specify which one is loaded.

`oskar.image.plot(<image data>, <FOV in deg.>, [figure title])`

Function to create MATLAB image plot labelled according to the specified field of view (FOV). Image data supplied to this function should be in the form of a square matrix.

`[settings] = oskar.image.init_settings()`

Initialise a MATLAB OSKAR image settings structure with default values. This structure is used to configure the OSKAR imager, run using the function `oskar.image.make()`.

`[image] = oskar.image.make(<visibilities>, <image settings>)`

Function to make an image or image cube from a set of visibilities. Visibilities should be supplied in the form of an OSKAR MATLAB visibility structure and image settings are represented as a structure of values as created by `oskar.image.init_settings()`. The image or image cube returned is in the form of an OSKAR image MATLAB structure.

`[image] = oskar.image.make(<uu>, <vv>, <amps>, <frequency in Hz>, <no. pixels>, <FOV in deg.>)`

Function to make an image from supplied visibility data. Input arguments consist of the baseline coordinates (uu, vv), in metres, complex visibility amplitudes, the frequency of the observation, in Hz, the number of pixels along each side of the image and the field of view, in degrees. The image created by this function is returned as an OSKAR image MATLAB structure.

3.4 oskar.convert.*: Functions for coordinate conversion

`[MJD UTC] = oskar.convert.date_time_to_mjd(<year>, <month>, <day>, <day_fraction>)`

Function to convert a date and time, in fractional days, to modified Julian days (UTC).

`[GAST] = oskar.convert.mjd_to_gast_fast(<MJD UT1>)`

Function to convert a modified Julian date to Greenwich apparent sidereal time (GAST).

3.5 oskar.settings.*: Functions to interface with OSKAR settings files

`oskar.settings.set(<filename>, <key>, <value>)`

Function to set the value of the specified settings key in the specified settings file.

3.6 oskar.sky.*: Functions for handling OSKAR sky models

`[sky] = oskar.sky.load(<filename>)`

Reads an ASCII format OSKAR sky model file into an OSKAR sky MATLAB structure.

`[sky] = oskar.sky.read(<filename>)`

Reads an OSKAR binary format sky model into an OSKAR sky MATLAB structure.

`oskar.sky.write(<filename>, <sky>)`

Function to write an OSKAR binary sky model file from an OSKAR sky MATLAB structure.

`oskar.sky.save(<filename>, <sky>)`

Function to write an OSKAR ASCII sky model file from an OSKAR sky MATLAB structure.

3.7 oskar.vis.*: Functions for handling OSKAR visibility data

`[vis] = oskar.vis.read(<file name>)`

A function that reads an OSKAR visibility binary file into an OSKAR MATLAB visibility workspace structure.

oskar.vis.write(<file name>, <vis>)

A function that writes an OSKAR visibility binary file from the specified OSKAR MATLAB visibility workspace structure.

oskar.vis.plot_amp(<vis>, [plot type], [polarisation], [time range], [channel], [baseline range])

Plots the specified OSKAR MATLAB visibility workspace structure, according to a number of options specified as function arguments. The plot type option takes a value of 1 or 2; if plot type = 1, a scatter plot of the baseline coordinates (uu, vv) coloured by the absolute value of the complex amplitude is generated; if plot type = 2, a scatter plot of uu-vv distance against complex amplitude is plotted. Polarisation is a string specifying the polarisation to plot, with allowed values of xx, yy, yx, xy, I, Q, U, or V. The time range, channel and baseline range allow selection of the data to be plotted. By default, time range = [1 vis.num_times], channel = 1, and baseline range = [1 vis.num_baselines].

oskar.vis.plot_uv(<vis>, [time range], [baseline range])

Plots the baseline (uu, vv) coordinates of the specified OSKAR MATLAB visibility workspace structure. Command line options for time range and baseline range are optional, and if not specified the entire data set is plotted (i.e. time range = [1 vis.num_times], and baseline range = [1 vis.num_baselines]).

[uvw] = oskar.vis.evaluate_baseline_uvw(<layout file>, <lon>, <lat>, <alt>, <RA>, <Dec>, <start time>, <no. times>, <dt>)

Generates the baseline coordinates for a given station and observation parameters. The station is specified as an OSKAR station layout file at a given longitude, latitude and altitude in degrees and metres respectively. Observation parameters are specified as: right ascension and declination in degrees; start time as a UTC modified Julian date; the number observation steps; and the integration length of each step, in seconds.

4 Example: Loading and Imaging Visibility Data

In order to load and image an OSKAR visibility data file, one can use the following steps:

1. Load an OSKAR visibility data structure into MATLAB:

```
>> vis = oskar.vis.read('example.vis');
```

2. Make a scatter plot of the baseline coordinates (see Figure 1):

```
>> scatter(vis.uu(:), vis.vv(:));
```

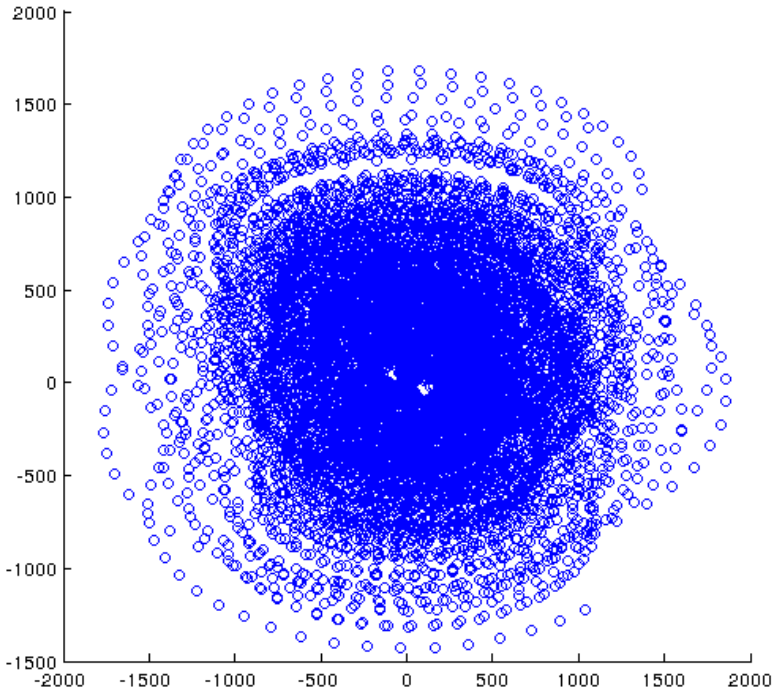

Figure 1: Scatter plot of baseline coordinates

3. Create an image settings structure and modify the field-of-view, image size, and image type settings:

```
>> settings = oskar.image.init_settings();

>> settings.fov_deg = 4;

>> settings.size = 256;

>> settings.image_type = oskar.image.type.linear
```

4. Invoke the OSKAR imager to make an image cube:

```
>> image = oskar.image.make(vis, settings);
```

5. Plot the XX dipole response for the first channel (as shown in Figure 2):

```
>> oskar.image.plot(image.data(:,:,1,1,1), image.fov_ra_deg);
```

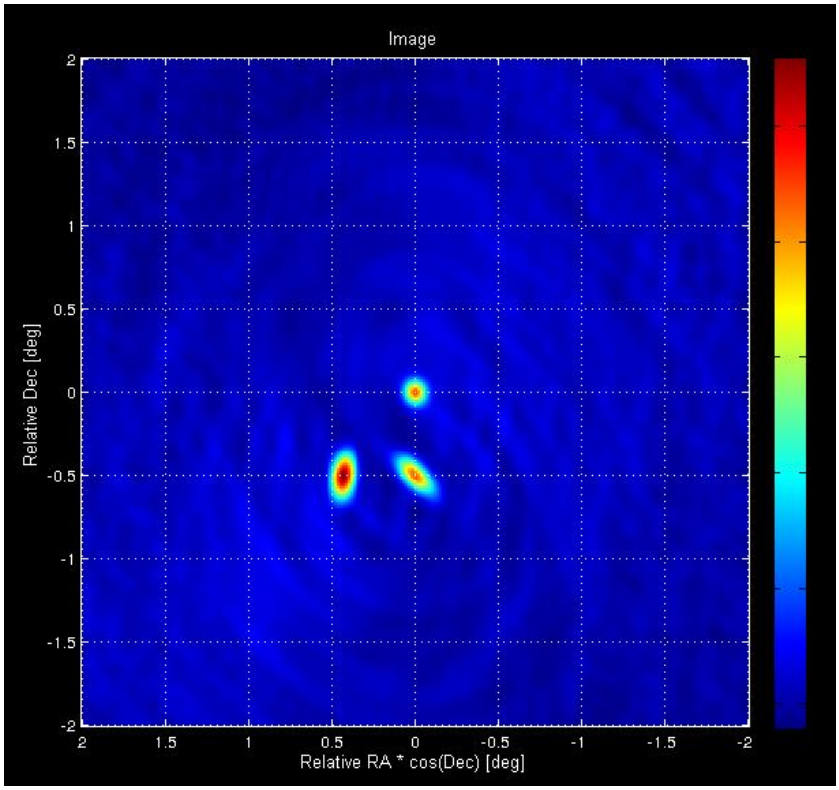

Figure 2: Raw image of XX polarisation for channel 1.