10th VLTI school of interferometry

…somewhere on the web!

Practice session: MATISSE data reduction

The MATISSE data reduction is built on two main features: a C library interfaced with the standard ESO scripting environment EsoRex, and additional python scripts.

The data reduction algorithms themselves rely on Fourier transforms and a careful averaging and treatment of the biases on the observables (spectrum, squared visibilities, closure phase, differential phases and coherent flux).

We will, in this practice session, go through the steps needed to reduce a MATISSE dataset, and evaluate its quality. You will have access to a machine where some MATISSE raw data and the pipeline are present.

On real datasets, that are terabytes large, we encourage you to use dedicated computing facilities like computing servers or mesocenters instead of simple laptops. The most stringent limit of the MATISSE pipeline currently is the amount of ram and disk access.

1. Looking at how the MATISSE data is organized

First of all, go to /data/drs on your assigned machine. There you will find a 2018-12-10_RAW directory containing raw MATISSE data and calibration files. You will start a terminal on the machine and type:

```
cd /data/drs/2018-12-10_RAW
ls
```

You will see a bunch of files with names "MATIS.XXX.fits" these are the MATISSE raw data you will get from the ESO archive, XXX being the date of archival. Of course, while very convenient for archival purposes, such files names are not very informative of the effective content of these files, therefore we have developed a few python scripts to make it clearer.

1.1. List stars observed during a night

For all the practice session, you will create a directory drs_practice:

```
mkdir ~/drs_practice
```

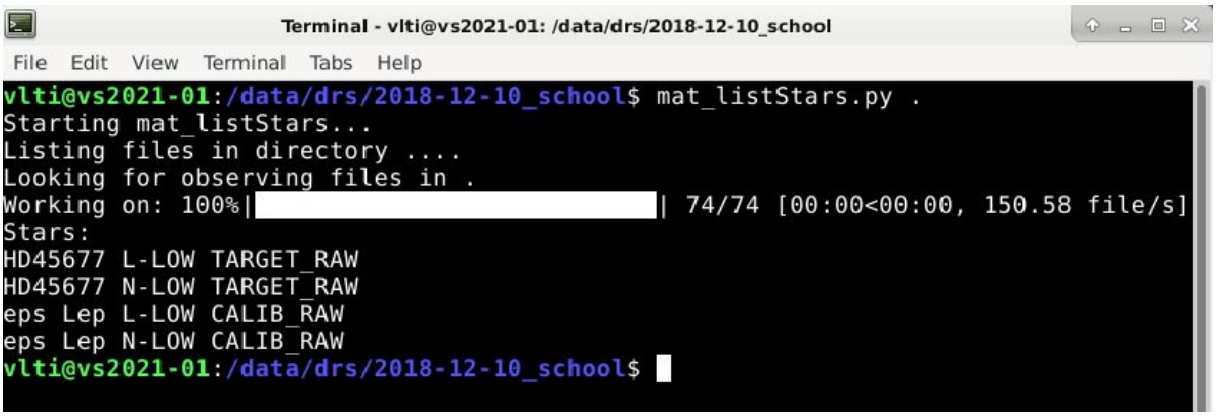
(if ~ does not function on your keyboard, try writing $HOME/drs_practice instead). then

```
cd
cd drs_practice
```

As a first, we will start by listing the stars that were observed during a night. Type:

```
mat_listStars.py /data/drs/2018-12-10_RAW
```

in the prompt. You should get the following answer:



You can see here that the dataset is made of two pointings, one on a science star: HD45677 (identified with the keyword “TARGET” followed by “RAW” meaning it is raw data) and one on a calibration star epsilon Lep, both observed simultaneously in the L-band and the N-band.

Now try typing the following command to get a short help on the command line options:

```
mat_listStars.py -h
```

All our python scripts have this little help to let you keep track f their functionalities.

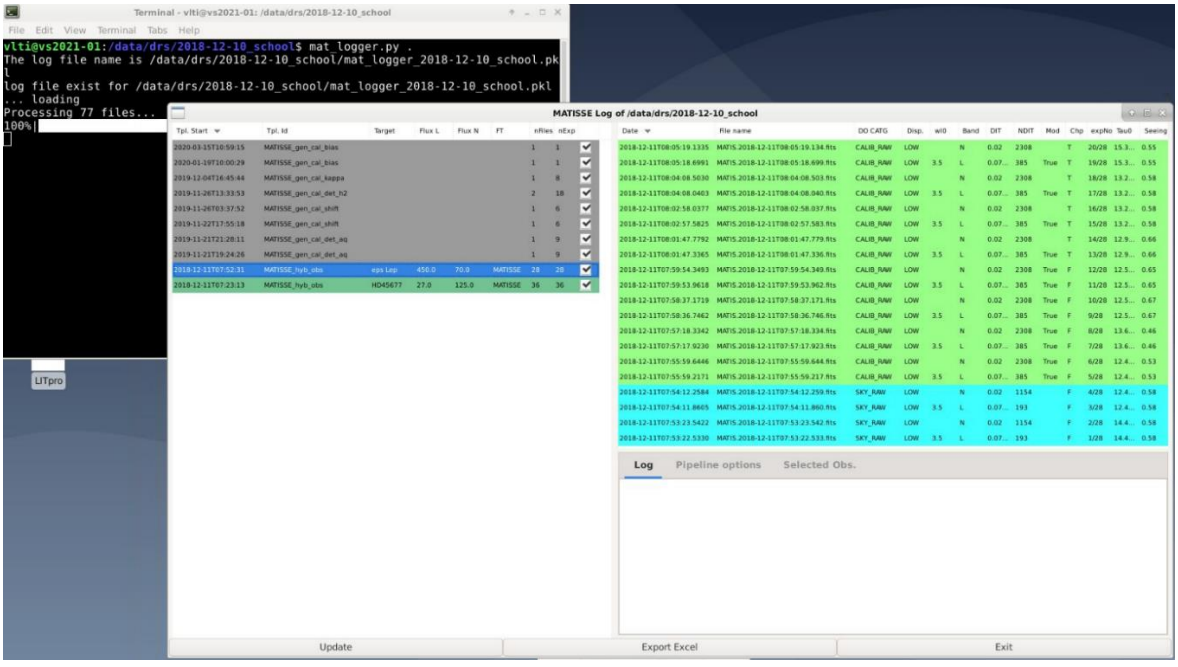
A similar little tool exists, **mat_listDITs.py**, that allows one to list the Detector Integration Times (DIT) used throughout a night. But still, we do not know with these tools to what correspond exactly which file.

1.2. Browse MATISSE data

The next tool to use is mat_logger.py. You can call it from within a directory containing raw data by typing

```
mat_logger.py /data/drs/2018-12-10_RAW
```

It should look like this:



This tool sorts the files by template number (corresponding to the execution of one observation of the target, an “OB” or “observing block” in ESO jargon), displays a bunch of useful information on each file, and more importantly shows the files in colours. This allows one to immediately identify which files correspond to what. For example here the files in green are fringe files (where the actual scientific data are recorded), while files in blue are sky files (pointing to a portion of the sky that does not contain an object, for sky background calibration purposes).

Observe the keywords displayed in the logger: they are all relevant to MATISSE data!

- “DO_CATG” is the data category (raw, reduced, calib, target, etc.),
- “Disp.” Is the prism/grating dispersion: LOW for a spectral resolution of ~30 (L and N), MED for ~500 (L), HIGH for ~300 (N) or ~1000 (L), and VHR for ~4000 (L),
- “Band” identifies the instrument arm: “L” for L and M bands and “N” for N-band,
- DIT is the Detector Integration Time, NDIT the number of frames recorded during the 1mn exposure,
- “Mod” is the OPD modulation (turned ON or OFF),
- “Chopping” tells whether the chopping is turned ON or OFF
- Finally, ambient conditions like coherence time (Tau0) and Seeing are also displayed

Quit this GUI by clicking on the “exit” button.

1.3. Display the content of raw MATISSE data

Now that you have identified a raw data file with mat_logger, you can display its content with the script:

```
mat_showRawData.py /data/drs/2018-12-10_RAW/<filename>
```

Let’s display a few raw exposures of the calibrator eps Lep, for instance :
- one L-band interferometric+photometric exposure (without chopping):

```
MATIS.2018-12-11T07:55:59.217.fits
```

- one N-band interferometric exposure (thus without chopping) :

```
MATIS.2018-12-11T07:55:59.644.fits
```

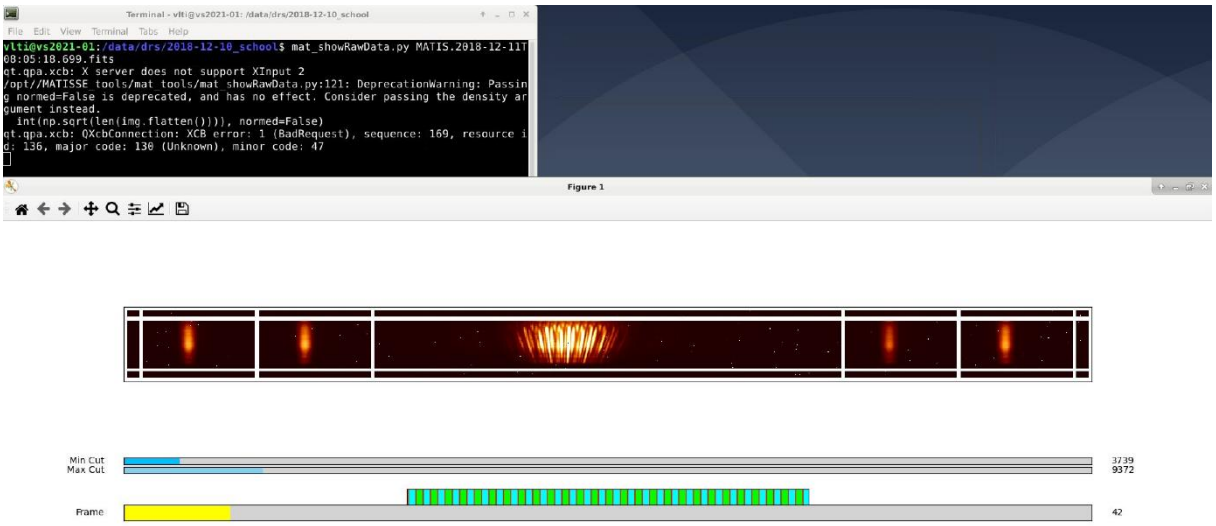
- one L-band interferometric+photometric exposure (with chopping):

```
MATIS.2018-12-11T08:01:47.336.fits
```

- one N-band photometric exposure (thus with chopping):

```
MATIS.2018-12-11T08:01:47.779.fits
```

Play with it a bit to browse through the frames, see chopped and non-chopped frames, compare L and N band files etc. The window you will get should resemble the one below, with the detector sub windows containing the 4 photometric beams (2 on the left and 2 on the right) and the interferometric beam (in the middle), in addition to calibration windows that are collected on the sides of the detector.



Back to **mat_logger.py** (see command above), by right-clicking on a file you can access to a context menu. Some of the context elements are useful:

- **Show with fv** (an external fits viewer)
- **Show raw data** (calling the mat_showRawData.py script)
- **Plot flux versus time**, to display the effect of Chopping

You can also use the NASA tool fv to visualize the fits files structure. Don't hesitate to try it also!

With all of these little tools (and more if you wish to contribute), you should be ready to go for the next step!

2. Reducing data by hand

2.1. How it is done

To test the proper installation of the MATISSE software, use the command

```
esorex --recipes
```

to check that you can find the "mat_xxx" recipes. esorex is a scripting language that enables one to set the parameters and file list to reduce data with any ESO pipeline (including the MATISSE pipeline). The recipes are scripts that take files (listed in a "sof", or "set of files" text file) and parameters (handed in the command line) as input and produces files as output. To create a "sof" file, one needs to list the files paths and their types. An example is given below:

```
/data/drs/2018-12-10_RAW/MATIS.2018-12-11T07:54:11.860.fits SKY_RAW
/data/drs/2018-12-10_RAW/MATIS.2018-12-11T08:05:18.699.fits CALIB_RAW
/data/drs/2018-12-10_RAW/KAPPA_MATRIX_HAWAII-2RG_LOW.fits KAPPA_MATRIX
/data/drs/2018-12-10_RAW/FLATFIELD_HAWAII.fits OBS_FLATFIELD
/data/drs/2018-12-10_RAW/SHIFT_MAP_HAWAII-2RG_LOW.fits SHIFT_MAP
/data/drs/2018-12-10_RAW/h2_slow_bpm_merged.fits BADPIX
/data/drs/2018-12-10_RAW/jsdc_2017_03_03.fits JSDC_CAT
```

Paste this text onto a text editor and save it into **~/drs_practice/test.sof**

Once the sof file created, one can call esorex with the proper options to reduce the data:

```
esorex mat_raw_estimates test.sof
```

It should display the progress of the reduction as it goes (should take about 2mn to reduce).

Take a look to all the files that have been created, using the **fv** tool:

- DSPtarget.fits and fringePeak.fits are diagnostic files to see if there are fringes in the dataset. You can visualize it with:
  `fv DSPtarget.fits`
  Similar diagnostic files are nrjImag.fits, nrjReal.fits, BSimag.fits, BSreal.fits for both the fringe detection and the bispectrum calculation
- CALIB_CAL_XXXX.fits contains the frames where all the detector defects should be corrected. You can visualize it with:
  `fv CALIB_CAL_0001.fits`
  It contains 3 tables, IMAGING_DATA, IMAGING_DETECTOR and ARRAY_GEOMETRY. The first one contains the actual data (open its content by clicking on the "All" button, ou can then open the individual frames
- PHOT_BEAMS_XXXX.fits contains the photometric channels that are resampled to the size of the interferometric channel
- OBJ_CORR_FLUX_0001.fits contains the correlated flux calculation
- OI_OPDWVPO_XXXX.fits contains the optical path difference computation
- RAW_VIS2_XXXX.fits, RAW_CPHASE_XXXX.fits, RAW_DPHASE_XXXXX.fits, andRAW_SPECTRUM_XXXX.fits contain the squared visibilities, closure phases, differential phases, and spectrum respectively.
- All these files are merged into one single oifits file CALIB_RAW_INT_0006.fits

2.2. Visualize the product: oifits files

We have a little tool to visualize the final oifits file:

```
mat_showOiData.py CALIB_RAW_INT_0006.fits
```

You can also use the fantastic oifitsexplorer tool from the JMMC to take a look to the oifits file content. In the command line, type

```
oifitsexplorer
```

3. Batch processing

3.1. Full throttle processing

Of course, manually editing text files and running them in command line may be cumbersome, especially for large datasets as for MATISSE!

We therefore developed a set of automatic pipeline scripts to reduce batches of data.

The first script is mat_autoPipeline.py It browses through a data directory, identifies the files to reduce, creates the sof files and runs the esorex scripts in one go. It creates all the intermediate files of the data reduction and the oifits files.

Example of command:

```
mat_autoPipeline.py --nbCore=2 --dirCalib=/data/drs/2018-12-10_RAW/ --skipN /data/drs/2018-12-10_RAW
```

It will run for about 30mn (15mn per OB). → go to next step directly

In the meantime, you can access to some reduced products in the directory

```
cd /data/drs/2018-12-10_REDUCED
```

There, you will find the same reduced data as done with esorex previously, but stored with a name corresponding to the executed recipe (mat_raw_estimates) and the detector (HAWAII for LM bands and AQUARIUS for N band), with all the intermediate files inside.

3.2. Tidying it up

For a batch data reduction, this is still not convenient for the later steps, so we designed a small script that extracts only the relevant reduced data, namely the oifits file, and renames it in a more human-readable way.

```
mat_tidyupOiFits.py /data/drs/2018-12-10_REDUCED
```

It will create a new directory 2018-12-10_REDUCED_OIFITS where all oifits files are stored. Take some time to look at the files. You will see that they are split by star, configuration, spectral resolution, Beam Commuting Device (BCD) positions (IN_IN, IN_OUT, OUT_IN, and OUT_OUT), and whether the observation used chopping or not. This separation is important for the next step(s).

3.3. Batch displaying data

First copy the directory in your home

The tool mat_showOiData.py allows one to produce a PDF file grouping all the oifits for a quick=look of whole datasets:

```
mat_showOiData.py --pdf --mergedpdf /data/drs/2018-12-10_REDUCED_OIFITS
```

Another useful tool is one that displays visibilities as a function of frequency

```
mat_showOiFreq.py /data/drs/2018-12-10_REDUCED_OIFITS
```

And also (u,v) coverage

```
mat_showUV.py --freq='as' --color='black' --showtitle=True /data/drs/2018-12-10_REDUCED_OIFITS
```

You can apply these scripts to data that have already been reduced on several nights: /data/drs/OIFITS

3.4. Displaying a transfer function

The following script displays the transfer function of a night

```
mat_showTransFunc.py /data/drs/OIFITS/2018-12-10_OIFITS
```

4. Calibrating you data

There is a recipe to calibrate data in the MATISSE pipeline that can be called with esorex in a similar way as the data reduction routine… and there is also an automatic python script to execute it conveniently on a batch of data!

First you will need to copy some of the raw oifits to your home directory

```
cp /data/drs/OIFITS/XXXX-XX-XX_OIFITS $HOME/ XXXX-XX-XX_OIFITS
```

```
mat_autoCalib.py /data/drs/OIFITS/XXXX-XX-XX_OIFITS
```

and voilà! You should have calibrated data ready for use for science, that you can visuzalize with oifitsexplorer and use un other JMMC tools like Litpro and Oimaging!