SHORT MANUAL TO GLM-MMIA DELAY & PEAK COMPARISON SOFTWARE

Last Revision June-2022

Contents

1 Overview

This document serves as a brief manual to using a program that compares GLM to MMIA signal based on input MMIA triggers as *.cdf* files. As main points, this program automatically gets:

1. Given MMIA triggers ordered by event in different directories
2. Downloaded corresponding GLM files from Google Cloud servers as *.nc* files
3. Extracted MMIA and GLM data for those files as *.mat* and *.txt* files, respectively
4. Top Cloud Energy - converted GLM and MMIA signals
5. Cross-correlated GLM-MMIA signals (and thus, MMIA delay with respect to GLM)
6. Peaks found in both signals

7. Matching peaks between both signals (GLM peaks present in MMIA signal)

8. Some statistics about MMIA delays and peak-matching

In next sections, how to use the script is explained in detail, from setting the environment (§2) to actually running it and outputting results (§3).

2 Setup and Dependencies

This script is mostly written in Python although some parts require MATLAB to be installed (with a valid license). *It is important to note that this script was designed to be run on a macOS or Linux-like machine.* Next steps are presented in groups as for Python-specific setup, MATLAB-specific setup and general setup.

2.1 Python Setup

As this script is mainly written in Python, a Python interpreter needs to be installed (Python 3.9.8 was used in the development). Some packages are needed and their installation is shown below (Listing 1). Apart from those packages, no more Python-specific setup needs to be done in order to run the code properly.

Listing 1: Necessary Python packages to install

```
# Numpy
pip install numpy

# Matplotlib (for plotting)
pip install matplotlib

# Scipy
pip install scipy

# Pandas
pip install pandas

# Pickle (for binaries storage)
pip install pickleshare

# netCDF4 for reading GLM .nc files
pip install netCDF4

# Google storage packages
pip install gsutil
pip install google
pip install --upgrade google-api-python-client
pip install google.cloud.storage
```

2.2 MATLAB Setup

As some part of the process requires a MATLAB run (specifically MMIA data outputting from *.cdf* files into *.mat* files for further processing), this software needs to be installed and properly activated. For that extraction to be done, a MATLAB patch for *.cdf* files must be downloaded, found here for macOS, Linux or Windows systems. Apart from that patch, no more setup is needed for MATLAB issues.

2.3 General Setup

In order for this script to run correctly, some previous minimal setup has to be considered.

First of all, at least 3 different directories should exist previous to the execution of the script. Those directories can be wherever in the computer's file system, separate or nested into one another, but must exist as three different folders for optimal functioning:

1. *Main Directory*: This directory should include both `main.py` and `library.py` files, as well as `MMIA_extraction.m` file (all three explained in §3) and the downloaded MATLAB CDF patch directory explained before.

2. *MMIA Files Directory*: Where all MMIA *.cdf* files to compare are located (all together).

3. *Results Directory*: Where all results will be outputted.[1]

Another important step in order to download GLM *.nc* files is to download the key for the GLM bucket, store the JSON key wherever in the filesystem and define its path as an environment variable with the name `GOOGLE_APPLICATION_CREDENTIALS`.[2]

3 Code Revision

The developed code can be found in this public GitHub repository.

3.1 Global structure

The global structure of the program consists on an executable `main.py` script which serves as a complete "list of tasks", being the data-input file and enumerating all steps by calling their respective functions. Those functions are found in `library.py` in order to keep the main script as clean as possible, being this second file just definitions of functions and having no need to enter it (unless willing to understand or change a process, of course).

A third MATLAB script called `MMIA_extraction.m` is needed for MMIA data extraction from *.cdf* to *.mat* files. In order to set it up and make it work with the main python script, it is needed to add the path to the MATLAB CDF patch at the beginning of the *.m* file (clearly marked inside the script).

[1] Important to notice that this directory's name should NOT include strings "_s", "_e" or "_c", as it may interfere with GLM filename system for *start* time of the file, *end* time and *creation* time.

[2] Example: Add line `export GOOGLE_APPLICATION_CREDENTIALS=<path-to-json-key>` to file `~/.bashrc` in Linux systems

3.2 Input data

User input data section on `main.py` looks like Listing 2, and should be the only part of the script with user interaction.

LISTING 2: All input parameters on `main.py`

```
'''
############################################################
##                      USER INPUT DATA                   ##
############################################################
'''

### GENERAL ###

# Boolean variable for setting everything for the first execution
first_execution = True

# Boolean variable for generating plots
show_plots = False

# Boolean variable for pre-cross-correlated data
pre_xc = False

# Boolean variable for pre-converted to top cloud energy
pre_tce = True

# Boolean variable for pre-detected peaks
pre_detected_peaks = True

# Boolean variable for pre-studied peaks
pre_studied = False

# Boolean variable for just outputting results
just_results = False

# Boolean variable for pre-oredered events in directories
pre_event_directories = True

# Boolean variable for deleting non-important directories at the end of
    execution
delete_non_important_directories = False

# Path to Hard Disk (with all MMIA files and where to store all files)
ssd_path = '/home/lrg/Desktop/tgfResults'


# Path where MMIA's .cdf files are located
MMIA_files_path = '/home/lrg/Desktop/tgfResults/tgfCases'

# Path to MATLAB executable
matlab_path = '/usr/local/MATLAB/R2021b/bin/matlab'


### GLM ###

# Time in seconds to analyze GLM before and after MMIA's time snippet
cropping_margin = 0.1
```

```
51
52 # GLM pixel size [km^2]
53 glm_pix_size = 8*8  # Colombia
54 #glm_pix_size = 78   # USA
55
56 # Plus of angle in latitude and longitude to snip GLM data
57 GLM_radius = 400 # [km]
58 angle_margin = GLM_radius / 111.11 # or a given value in degrees
59
60 # Minumum number of peaks to consider a valid event for GLM
61 glm_min_peak_num = 3
62
63 # Boolean variable for downloading GLM .nc files from Google Cloud Storage
64 pre_downloaded_GLM = True
65
66 # Boolean variable for pre-extracted files
67 pre_extracted_GLM = True
68
69 # Boolean variable for integrating GLM signals if not pre-done
70 pre_conditioned_GLM = True
71
72 # Boolean variable for defining TGF events (outputs GLM .txt's that could
      be cross-correlated). Uses a TGF list to evaluate the cross-correlation
       algorithm
73 tgf = True
74
75
76 ### MMIA ###
77
78 # Boolean variable for pre-extracted files
79 pre_extracted_MMIA = True
80
81 # Boolean variable for conditioning MMIA data if not done before
82 pre_conditioned_MMIA = True
83
84 # Maximum length in seconds of each event
85 event_length = 2 # [s]
86
87 # Minimum time to consider as two sepparate events
88 split_window = 2000*0.00001
89
90 # Threshold for MMIA signal
91 mmia_threshold = 1.75   # [micro W / m^2]
92
93 # Minumum number of peaks to consider a valid event for MMIA
94 mmia_min_peak_num = 3
95
96 '''
97 ############################################################
98 ##                 END OF USER INPUT DATA                 ##
99 ############################################################
100 '''
```

The main variables for setting an execution are, first and foremost:

- `ssd_path`: String defining the path to the directory where all results will be outputted into
- `MMIA_files_path`: String defining the path to the directory containing all MMIA *.cdf* files
- `matlab_path`: String defining the path to MATLAB binary

While there are some other specific variables both for GLM and MMIA or for the functioning of the program being:

- `cropping_margin`: Time in seconds to crop GLM signal before and after MMIA signal's starting and ending time (for better match and cross-correlation).
- `glm_pix_size`: Pixel size for GLM, important for TCE conversion [km^2]
- `GLM_radius`: Radius in [km] to search for GLM signals with respect to MMIA coordinates (`angle_margin` is calculated after this value or can be directly given)
- `(glm/mmia)_min_peak_num`: Minimum number of peaks to find for GLM or MMIA, respectively. If found less (but not equal to) in any of them, the event is discarded.
- `event_length`: Maximum time for a MMIA event in order to classify its *.cdf* files in events [s]
- `split_window`: Minimum time with no signal inside a MMIA event to be considered two different events (because of the `event_length` definition)
- `mmia_threshold`: Maximum value of MMIA signal to be considered noise $\left[\frac{\mu W}{m^2}\right]$
- `show_plots`: Boolean variable for outputting plots throughout the whole program
- `delete_non_important_directories`: Boolean variable for deleting all non-necessary directories used in the execution of the program at its conclusion if not desired (saves storage)
- `tgf`: Boolean variable for stating that all MMIA *.cdf* files represent TGF's, outputting all generated GLM *.txt* files that could cross-correlate with MMIA

The other boolean variables are explained due to the contruction of the `main.py` scipt. The whole document consists of a series of steps as stated previously, which store their results just after finishing that step (storing MMIA conditioned data, then GLM conditioned data, then cross-correlated data, ...) so *if the previous steps were made correctly*, some parts of the code can be deactivated and necessary data will be uploaded from binaries, making the processing of data way faster if one step is in testing stage or a single little change was made. Those main sections and their corresponding boolean variable are:

1. Order MMIA *.cdf* files in daily events (`pre_event_directories`)
2. Extract MMIA data from *.cdf* files to *.mat* files (`pre_extracted_MMIA`)

And then, for every day with existing MMIA data:

3. MMIA data conditioning (`pre_conditioned_MMIA`)
4. GLM *.nc* files download (`pre_downloaded_GLM`)
5. GLM data conditioning (`pre_conditioned_GLM`)
6. Top Cloud Energy Conversion (`pre_tce`)
7. GLM-MMIA Cross Correlation (`pre_xc`)
8. GLM-MMIA peak detection and peak matching (`pre_detected_peaks`)
9. Output results (`pre_studied`)

And some extra special variables:

10. `first_execution`: Bool variable for executing the whole program (no need to set all previous boolean variables to False)
11. `just_results`: Bool variable for jumping all processing and just computing results. Important to note that all previous steps should be correctly done, being handful if all processing was pre-done and more variables want to be extracted

3.3 Output data

At the conclusion of the execution, some useful information remains storaged in the outputting directory. Among the most useful, and depending on outputting boolean variables:

- GLM downloaded *.nc* files for their corresponding ordered MMIA *.cdf* files
- GLM and MMIA extracted data (in *.txt* and *.mat* file formats, respectively)
- GLM and MMIA conditioned, TCE-converted signals, as *pickle* binaries
- GLM and MMIA cross-correlated signals, as *pickle* binaries, and their representations pre- and post- cross correlation
- MMIA delays for every event
- All GLM and MMIA peaks as well as their matching peaks and their representation in plots, per event
- Some basic statistics (including the most important variables for further statistical studies in a *.mat* format)

Giving special attention to the output *.mat* file called `vars_for_stats.mat`, where all important results can be found, its fields represent:

- `all_delays_t`: A vector containing all MMIA delays in seconds for every correlated event (MMIA was not noise-only and MMIA and GLM files contained data). *Positive values show GLM coming after MMIA, and negative values show GLM coming before MMIA signal.*

- `glm_mmia_matching_time_signal`: Four vectors (1st and 2nd for GLM, 3rd and 4th for MMIA) containing time (in seconds) and the signal peak value (in joules, after TCE conversion) of all matching peaks. Every row represents a different matching peak with time and signal values for GLM and the same for MMIA.
- `matching_time_distribution`: Twelve position vector counting the number of matching peaks for every 2-hour time segment of the day.
- `peak_relations`: Two vectors, where every row represents a correlated event with a minimum of `glm_min_peak_num` peaks for GLM and `mmia_min_peak_num` peaks for MMIA. First column represents the relation between matching peaks over all GLM peaks for that row-event, and the second column represents the same over all MMIA peaks for that event.

3.4 First Setup Checklist

As a matter of summary for making the script work properly for the first time:

1. Clone GitHub Repository (will be main directory)
2. Download MATLAB CDF Patch and move it to main directory (with `main.py`, ...)
3. Enter *MMIA_extraction.m* and add path to the patch
4. Install all Python packages
5. Download GLM JSON key and add it as an environment variable
6. Create a directory with all desirable MMIA *.cdf* files
7. Create a directory where all results will be outputted
8. Enter `main.py` and define paths to directories and to MATLAB binary, as well as all input variables
9. Execute `main.py`

4 Flow Diagram for `main.py`

Shown in next page, Fig.(1).

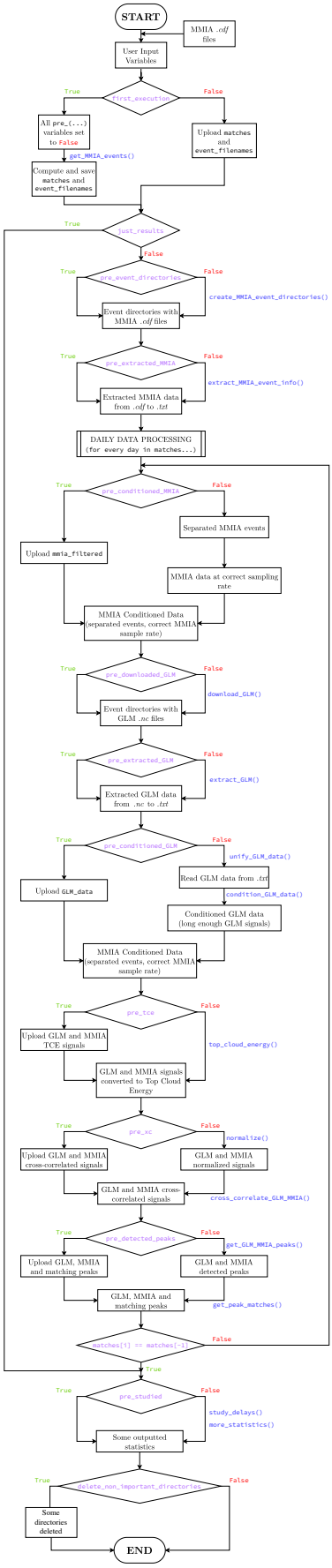

Figure 1: Basic flow diagram of the `main.py` script

5 Example

This section will serve as a practical example for using the script `main.py`, assuming all programs are already installed as well as their dependencies (*i.e.* MATLAB, CDF patch for MATLAB, Python and necessary `pip` packages and the JSON key for downloading GLM data). It also assumes there already exists a directory with all pre-downloaded MMIA *.cdf* files to analyze, as this script does not download MMIA data directly from the server.

5.1 Setting the environment

First of all, the code has to be cloned or downloaded from https://github.com/jaimefranm/GLM-ASIM. To clone it directly to the Desktop simply run from the command line:

```
>> cd; cd Desktop; git clone https://github.com/jaimefranm/GLM-ASIM.git
```

This will create a directory in the Desktop called *GLM-ASIM*, containing scripts `main.py`, `library.py` and `MMIA_extraction.m`, along with other files.

IMPORTANT: Move the downloaded JSON key to this folder, and moving the downloaded MATLAB CDF patch directory (macOS or Linux) is recommended for simplicity.

Access MATLAB file `MMIA_extraction.m` and add path to the CDF patch at the beginning of the file (it is clearly indicated). Finally, Fig.(2) shows how this directory should look like.

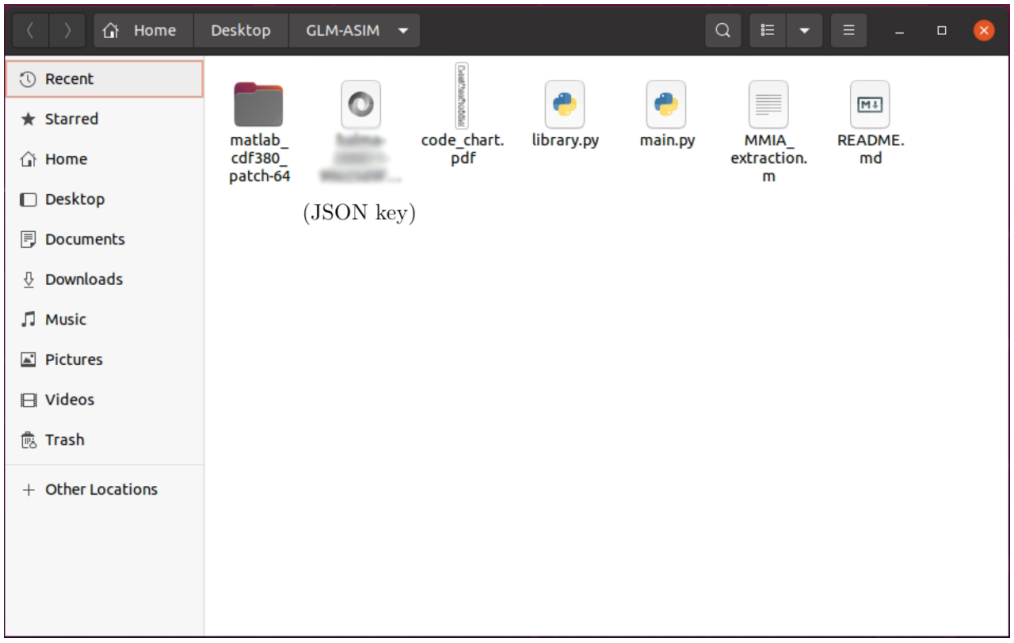

FIGURE 2: Cloned or downloaded directory containing all important files.

5.2 Creation of the necessary directories

As mentioned before, a directory with all provided MMIA *.cdf* files should exist as a starting point. This directory contains all files in no specific order of date or event, and represents the `MMIA_files_path` input variable in Listing 2, line *41*. Fig.(3) shows the interior of this directory for an example containing all lightning data over Colombia in 2020.

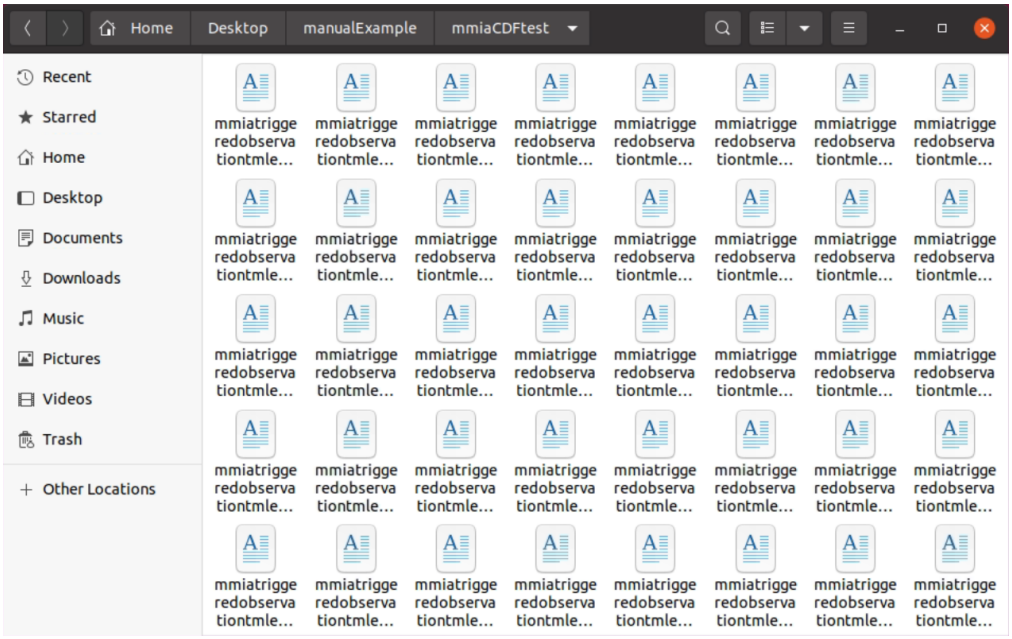

FIGURE 3: Directory with all MMIA *.cdf* files

A new directory for storing all processed data, results and necessary middle results has to be created as well. This will be the `ssd_path` input variable in Listing 2, line *37*.

The MMIA CDF files directory, as well as the outputting directory, can be located anywhere in the filesystem. For the sake of this example and for simplicity, all directories are located inside a test directory called *manualExample* in the desktop, so the path to all MMIA input files and outputted data would be `~/Desktop/manualExample/`. Fig.(4) shows an schematic of the directory tree up to this point.

```
lrg@lrg-pc1:~$ tree -d Desktop/manualExample/
Desktop/manualExample/
├── mmiaCDFtest
└── testOutput

2 directories
```

FIGURE 4: Initial example tree of necessary directories

WARNING: Directory names mustn't contain the character strings `_s`, `_e` or `_c` in order to scape GLM *.nc* file naming for *start*, *end* and *creation* times. `lowerUpperCaseNaming` is preferred to `underscore_naming` in order to scape from writing those strings unintencionally and getting undesired errors or warnings.

5.3 Input variables assignation

Once the necessary directories have been created, one can access the script `main.py` and start defining the input variables. As seen in §3.2 and following with the example presented in the previous subsection (*manualExample/* in Desktop with MMIA data for Colombia in 2020), Listing 3 shows the values of the input data for this example ***for a first execution***. The most important variables to see are:

- `MMIA_files_path`: Set to the path to the directory containing all MMIA *.cdf* files explained in the previous subsection (the one in Fig.(3)).
- `ssd_path`: Set to the path to the directory where all output will be stored. In this case, the void directory *testOutput/* inside the general *manualExample/* directory.
- `first_execution`: Set to `True` in order to bypass all other boolean variables and set them to `False`. If this was not the first execution and some previous **correct** data was generated, `first_execution = False` and the corresponding boolean variable should be set to `True` in order to make the execution faster by using that existent data. IMPORTANT: Make sure the code or input MMIA *.cdf* files were not changed from the previous execution in order to make sure those results are indeed correct before assuming they are.
- `glm_pix_size`: Set to 64 km^2 as the input data was over Colombia, where GLM pixel projection covers that area. In case of analyzing higher or lower altitudes (far from the equator) this value should be revised.

IMPORTANT: Make sure path variables `MMIA_files_path`, `ssd_path` and `matlab_path` do NOT end with the character '/', as it will be added when needed by the script.

All the other parameters are tuned for an overall good performance of the program, but can be changed (*i.e.* outputting plots, maximum considered length for an event, radius around MMIA signal to look for GLM signals, etc.). Note that the variable `matlab_path` is set to '`/usr/local/MATLAB/R2021b/bin/matlab`' as this is the usual path for MATLAB installation in Ubuntu (and in this case, version 2021b).

Listing 3: Example input parameters on `main.py` for a first execution

```
'''
############################################################
##                      USER INPUT DATA                   ##
############################################################
'''

### GENERAL ###

# Boolean variable for setting everything for the first execution
first_execution = True

# Boolean variable for generating plots
show_plots = False

# Boolean variable for pre-cross-correlated data
```

```
pre_xc = False

# Boolean variable for pre-converted to top cloud energy
pre_tce = False

# Boolean variable for pre-detected peaks
pre_detected_peaks = False

# Boolean variable for pre-studied peaks
pre_studied = False

# Boolean variable for just outputting results
just_results = False

# Boolean variable for pre-oredered events in directories
pre_event_directories = False

# Boolean variable for deleting non-important directories at the end of
    execution
delete_non_important_directories = False

# Path to Hard Disk (with all MMIA files and where to store all files)
ssd_path = '/home/lrg/Desktop/manualExample/testOutput'

# Path where MMIA's .cdf files are located
MMIA_files_path = '/home/lrg/Desktop/manualExample/mmiaCDFtest'

# Path to MATLAB executable
matlab_path = '/usr/local/MATLAB/R2021b/bin/matlab'


### GLM ###

# Time in seconds to analyze GLM before and after MMIA's time snippet
cropping_margin = 0.1

# GLM pixel size [km^2]
glm_pix_size = 8*8  # Colombia
#glm_pix_size = 78   # USA

# Plus of angle in latitude and longitude to snip GLM data
GLM_radius = 400 # [km]
angle_margin = GLM_radius / 111.11 # or a given value in degrees

# Minumum number of peaks to consider a valid event for GLM
glm_min_peak_num = 3

# Boolean variable for downloading GLM .nc files from Google Cloud Storage
pre_downloaded_GLM = False

# Boolean variable for pre-extracted files
pre_extracted_GLM = False

# Boolean variable for integrating GLM signals if not pre-done
pre_conditioned_GLM = False

# Boolean variable for defining TGF events (outputs GLM .txt's that could
```

```
    be cross-correlated). Uses a TGF list to evaluate the cross-correlation
     algorithm
72 tgf = False
73
74
75 ### MMIA ###
76
77 # Boolean variable for pre-extracted files
78 pre_extracted_MMIA = False
79
80 # Boolean variable for conditioning MMIA data if not done before
81 pre_conditioned_MMIA = False
82
83 # Maximum length in seconds of each event
84 event_length = 2 # [s]
85
86 # Minimum time to consider as two sepparate events
87 split_window = 2000*0.00001
88
89 # Threshold for MMIA signal
90 mmia_threshold = 1.75   # [micro W / m^2]
91
92 # Minumum number of peaks to consider a valid event for MMIA
93 mmia_min_peak_num = 3
94
95 '''
96 ############################################################
97 ##                 END OF USER INPUT DATA                 ##
98 ############################################################
99 '''
```

5.4 Execution

Once the input variables have been correctly stipulated in `main.py`, the execution can be done. For doing so, simply run from the command line:

```
>> python main.py
```

5.5 Results

After waiting for the script to end its execution, some new directories will appear in *testOutput/* (or the folder specified as `ssd_path` variable). Fig.(5) shows how this directory looks like at the end of the execution.[3]

Several directories represent middle results stored as Python '`pickle`' binaries, others represent extracted GLM and MMIA data (from *.nc* and *.cdf* files to *.txt* and *.mat*, respectively), while others represent different figures (resulting from cross-correlation, peak identification or peak matching, for example) or ordered GLM and MMIA files per event.

File `RESULTS.txt` presents a simplified view of the general results (number of events evaluated, average delay, average number of peaks, etc), while it is more recommended to

[3]If variable `delete_non_important_directories` was set to `True`, directories `general_variables_path`, `GLM_ordered_dir`, `GLM_ordered_outputs`, `GLM_conditioned_bin`, `MMIA_mats_path`, `MMIA_filtered_bin` and `path_to_mmia_dirs` will not appear as they will be deleted.

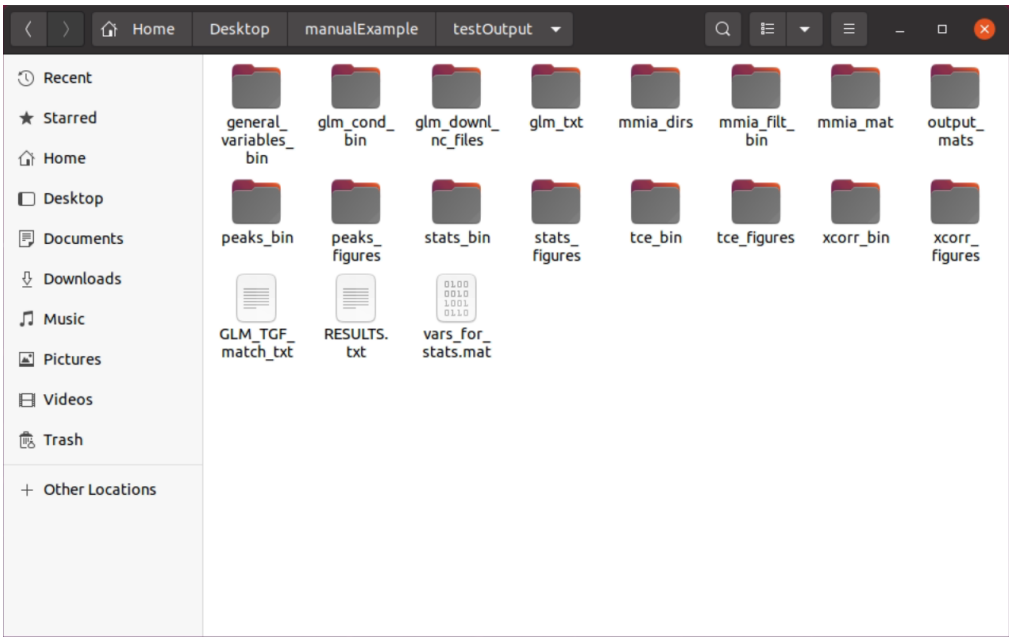

Figure 5: Output directory `ssd_path` after the execution of the script

grab results from the file `vars_for_stats.mat`. Its structure and a little description of the different variables it contains was given at the end of §3.3.