

# **MCMTp**y

**version 0.1.0a1**

**Fu Yin**

■■ 28, 2021



# Contents

Welcome to MCMTpy-test's documentation!	1
Contents:	2
Installation	2
Dependencies	2
Quick installation	2
Source installation	3
Parallel installation (for future version)	3
1. Install pyfk	3
2. Install the HDF5 parallel version	3
3. Install the h5py parallel version	4
1. Install pyasdf parallel version	5
5. Install MCMTpy parallel version	5
Reference Installation Blog	5
Tutorial	5
About ASDF format	5
About pyfk	5
Short tutorial	6
Tutorial_Detailed	11
Calculate Green Function Database	11
Station and Source	12
MPI	13
Path	13
SourceModel	14
SeisModel	14
Config	14
Logging file	15
Example	15
Synthesize the Test Data	16
Path	16
Source	16
Station	16
Source Time Function	17
Filter	17
Add Noise	17
Output	17
Example	18
Prepare Data for MCMTpy	19
Format of MCMTpy input data:	19
Example	20
Inversion of Source	23

Path	23
Source Time Function	24
MPI and Chains_n	24
Misfit	24
Inversion parameters	26
Alpha	27
Logging file	28
Example Double-Couple Inv	28
Result Visualization	29
Path	29
Hist	30
Misfit	30
Waveform	30
Example	31
Conversion of Source Parameters	36
The installation	37
str_dip_rake to mt	37
The conversion between str_dip_rake and A/N vector	37
The conversion between A/N vector and P/T/N vector	38
mt to P/T/N vector	38
P/T/N vector to P/T/N vector's stirke and dip	39
Describe_fault_plane with two str_dip_rake	39
Plot Beachball with Station Projected	39
Moment Tensor Decompose	42
Huston Plot	47
References	49
ASDF	49
pyfk	49
NoisePy	49
MoPaD	49
BEAT	50
Other references adding...	50
<b>Indices and tables</b>	<b>50</b>

# Welcome to MCMTpy-test's documentation!



## Note

Version 0.1.0a1 is the currently release.

This is the documentation for the Python package of **MCMTpy**, which is python tool for seismic source study. For further information please see below website:

- Github repository of **MCMTpy**: <https://github.com/OUCyf>

If you use **MCMTpy** for your research and prepare publications, please citing **MCMTpy**:

- MCMTpy: A Python Package for Simultaneous Inversion of Source Location, Focal Mechanism, and Rupture Directivity. In prep for Seismological Research Letter.

# Contents:

## Installation

<b>Dependencies</b>	<b>2</b>
<b>Quick installation</b>	<b>2</b>
<b>Source installation</b>	<b>3</b>
<b>Parallel installation (for future version)</b>	<b>3</b>
1. Install pyfk	3
2. Install the HDF5 parallel version	3
3. Install the h5py parallel version	4
1. Install pyasdf parallel version	5
5. Install MCMTpy parallel version	5
<b>Reference Installation Blog</b>	<b>5</b>

## Note

### ■ This package is still undergoing development. ■

It is not recommended use **Parallel installation**, because it's somewhat complicated and prepared for the future version for parallel reading of data. Now use **Quick installation** or **Source installation**, you can also run MCMTpy in parallel. Please note that the test is performed on **macOS Big Sur (11.2.1)**, so it could be slightly different for other OS.

## Dependencies

The package MCMTpy runs on Unix-like systems including Mac and Linux. Package need **Python 3.6 or greater**, but python 3.9 is not supported for `pyfk`. It depends on the following Python modules:

- `numpy>=1.14`
- `tqdm>=4.19.4`
- `matplotlib<=3.1.1`
- `mpi4py`
- `obspy`
- `pyfk`
- `pyasdf`
- `json5`

For parallel:

- `h5py`

For data preprocessing and visualization:

- `pygmt`

We recommend to use [anaconda](#) as your python environment, and use [conda](#) and [pip](#) to install those libraries.

## Quick installation

Firstly, make sure **Anaconda** has been installed, then:

Contents:

```
$ conda create -n MCMTpy python=3.8 numpy=1.16 matplotlib=3.1.1 mpi4py obspy pyasdf json5 t  
$ conda activate MCMTpy  
$ pip install pyfk  
$ pip install MCMTpy
```

Some errors may occurred of that pyfk not support the new version of cysignals, please:

```
$ conda uninstall cysignals  
$ pip install cysignals==1.10.2  
$ pip install pyfk
```

Successful installation will generate an executable program MCMTpy under the anaconda environment path: /Users/user/opt/anaconda3/bin. And run following code to check the installation:

```
$ MCMTpy --help
```

## Source installation

```
$ git clone https://github.com/OUCyf/MCMTpy.git  
$ cd MCMTpy  
$ python setup.py install
```

## Parallel installation (for future version)

### 1. Install pyfk

Firstly, make sure **Anaconda** has been installed, then:

```
$ conda create -n MCMTpy python=3.8 numpy=1.16 obspy  
$ conda activate MCMTpy  
$ pip install pyfk
```

Some errors may occurred of that pyfk not support the new version of cysignals, please:

```
$ conda uninstall cysignals  
$ pip install cysignals==1.10.2  
$ pip install pyfk
```

■ Make sure the new environment does not include the following packages that we will install below: ■

### 2. Install the HDF5 parallel version

- First, make sure **openmpi** or other **MPI** program has been installed, preferably source code installed. You can use **which mpicc** to get the installation's path.
- Activate the MCMTpy environment, and uninstall the **mpi4py** package in this environment (if it have). Then use **pip install the mpi4py package not conda**.

```
$ conda uninstall mpi4py  
$ pip install mpi4py
```

## Note

When you use **pip install mpi4py**, it will use the **mpicc** in your system. But if you use **conda** to do the installation, it resolves the dependent environment itself and install **conda's mpicc** in your environment which will cause problems for subsequent installations due to the not source code compilation of **MPI**. After installing **mpi4py**, you can use **which mpicc** to check whether it is a system mpicc or a **conda mpicc**.

## Contents:

- Then you need to install the parallel version of **HDF5**. Download the source package, compile and install the parallel version to your path. Your system may already have HDF5 installed. Software such as GMT relies on the HDF5 package, so use **h5cc-showconfig** to see more information about installation, note that h5cc is usually not shown in parallel. We do not want to uninstall the existing version of HDF5 on our system. This does not conflict with our subsequent installation of the parallel version of HDF5, as long as we specify the installation path of the parallel version.

```
$ chmod 777 ./configure # enable read/write access
$ ./configure --enable-parallel --enable-shared --prefix=/your/path/of/hdf5_dir/
$ make
$ make check
$ make install
```

- After the installation, the executable program will be generated in **./bin** path of your **hdf5\_dir** directory. **h5pcc** will be generated due to the parallel installation, similar to the **h5cc** will be generated by the non-parallel installation. Check the installation version information:

```
$ h5pcc -showconfig
```

### 3. Install the **h5py** parallel version

- Before installing **h5py**, you need to uninstall both **hdf5** and **h5py** in your environment, and then download **h5py** <<https://github.com/h5py/h5py>> and compile the source code. You need to specify the path of **HDF5** and **CC**.

```
$ export CC=/path/to/mpicc # do not use conda's mpicc
$ echo %CC
$ export HDF5_MPI="ON"
$ export HDF5_DIR="/your/path/of/hdf5_dir/"
$ pip install .
```

- Check the installation version information (download the **demo2.py** on h5py):

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from mpi4py import MPI
import h5py

rank = MPI.COMM_WORLD.rank # The process ID (integer 0-3 for 4-process run)

f = h5py.File('parallel_test.hdf5', 'w', driver='mpio', comm=MPI.COMM_WORLD)

dset = f.create_dataset('test', (4,), dtype='i')
dset[rank] = rank

f.close()
```

Run the program:

```
$ mpiexec -n 4 python demo2.py
```

Looking at the file with **h5dump**:

```
$ h5dump parallel_test.hdf5
HDF5 "parallel_test.hdf5" {
GROUP "/" {
    DATASET "test" {
        DATATYPE H5T_STD_I32LE
        DATASPACE SIMPLE { ( 4 ) / ( 4 ) }
        DATA {
            (0): 0, 1, 2, 3
        }
    }
}
```

## Tutorial

```
}
```

### 1. Install pyasdf parallel version

- After the above steps are completed sucessful, you can directly use pip to install pyasdf

```
$ pip install pyasdf
$ python -c "import pyasdf; pyasdf.print_sys_info()" # (check for a parallel version)
```

### 5. Install MCMTpy parallel version

- Finally, you can install the parallel version of MCMTov.

```
$ pip install MCMTpy          # or python setup.py install
$ pip uninstall MCMTpy        # uninstall
```

## Reference Installation Blog

- <https://drtiresome.com/2016/08/23/build-and-install-mpi-parallel-hdf5-and-h5py-from-source-on-linux/>
- <https://gist.github.com/kentwait/280aa20e9d8f2c8737b2bec4a49b7c92>
- <https://github.com/h5py/h5py/issues/759>
- <https://seismicdata.github.io/pyasdf/installation.html>
- <https://noise-python.readthedocs.io/en/latest/installation.html>
- <https://docs.h5py.org/en/stable/mpi.html>
- <https://docs.h5py.org/en/stable/build.html>
- <https://github.com/conda-forge/hdf5-feedstock/issues/118>
- <https://distarray.readthedocs.io/en/latest/hdf5-notes.html>
- <https://forum.hdfgroup.org/t/installing-hdf5-ready-version-of-open-mpi/4998>

## Tutorial

In this tutorial, we will show some of the key steps in **MCMTpy** for **Focal Mechanism Inversion**, including **DC** (double couple) and **MT** (moment tensor) inversion (not in now).

### About ASDF format

The users who are interested in the details of **ASDF format** are referred to the following publication. And the pyasdf Github repository <https://github.com/SeismicData/pyasdf>.

- Krischer, L., Smith, J., Lei, W., Lefebvre, M., Ruan, Y., de Andrade, E.S., Podhorszki, N., Bozdal, E. and Tromp, J., 2016. An adaptable seismic data format. Geophysical Supplements to the Monthly Notices of the Royal Astronomical Society, 207(2), 1003-1011.

You can also find some useful examples about pyasdf in [NoisePy](#).

### About pyfk

The users who are interested in the details of **pyfk** are referred to the pyfk Github repository <https://github.com/ziyixi/pyfk>.

## Short tutorial

MCMTpy stores all the parameter information in four JSON files: **build\_GFs.json**, **syn.json**, **sample.json** and **plot.json**, and you can find those json-files in path `./MCMTpy-master/jsons/`. For parameters choosing, please refer to **Detailed tutorial**. The steps to do inversion process are:

### 1. Calculate green function database:

```
$ MCMTpy build_GFs pyfk -c ./build_GFs.json
$ mpirun -n 4 MCMTpy build_GFs pyfk -c ./build_GFs.json # parallel
```

### 2. Synthesize the test data:

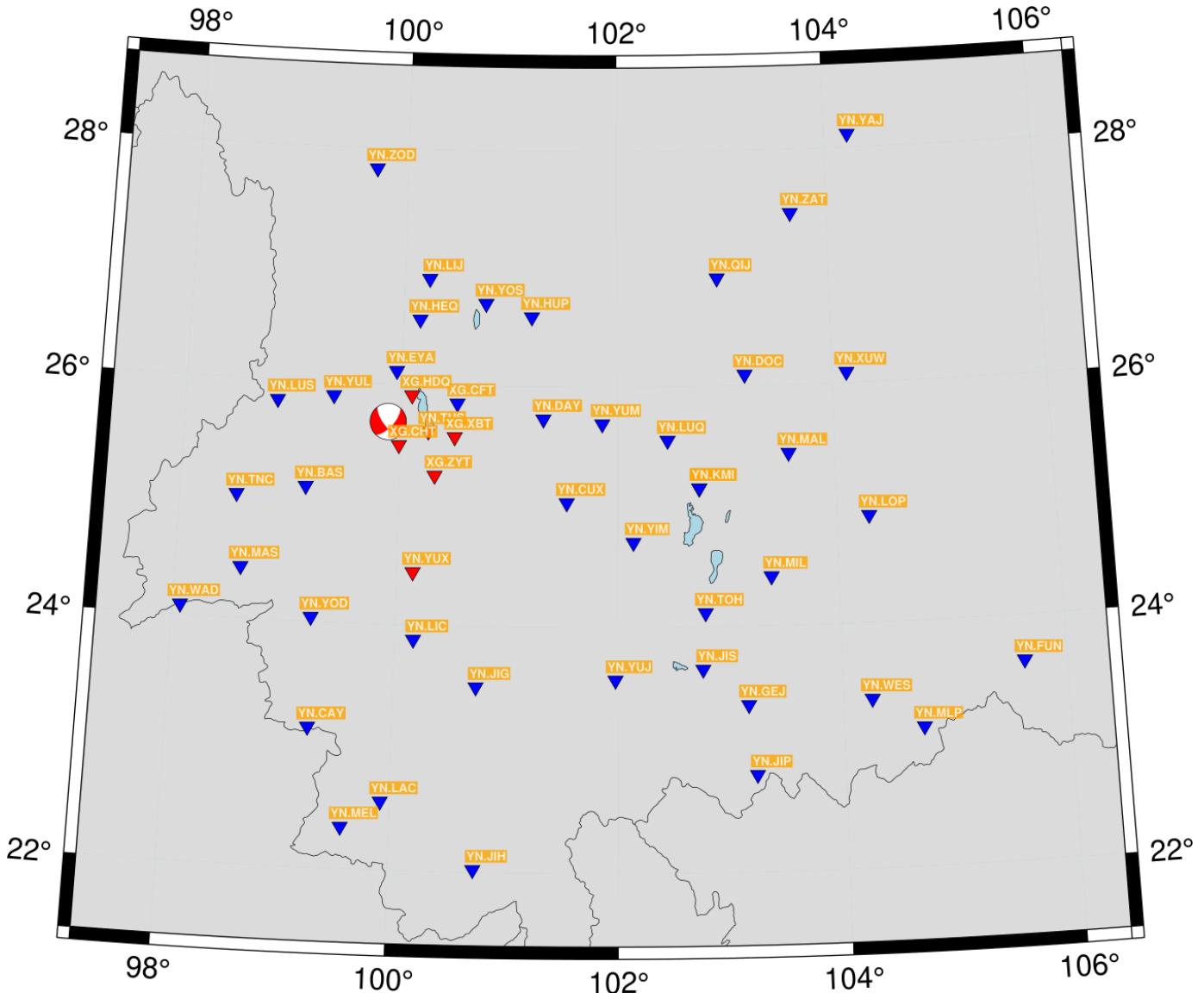
```
$ MCMTpy syn pyfk -c ./syn.json
```

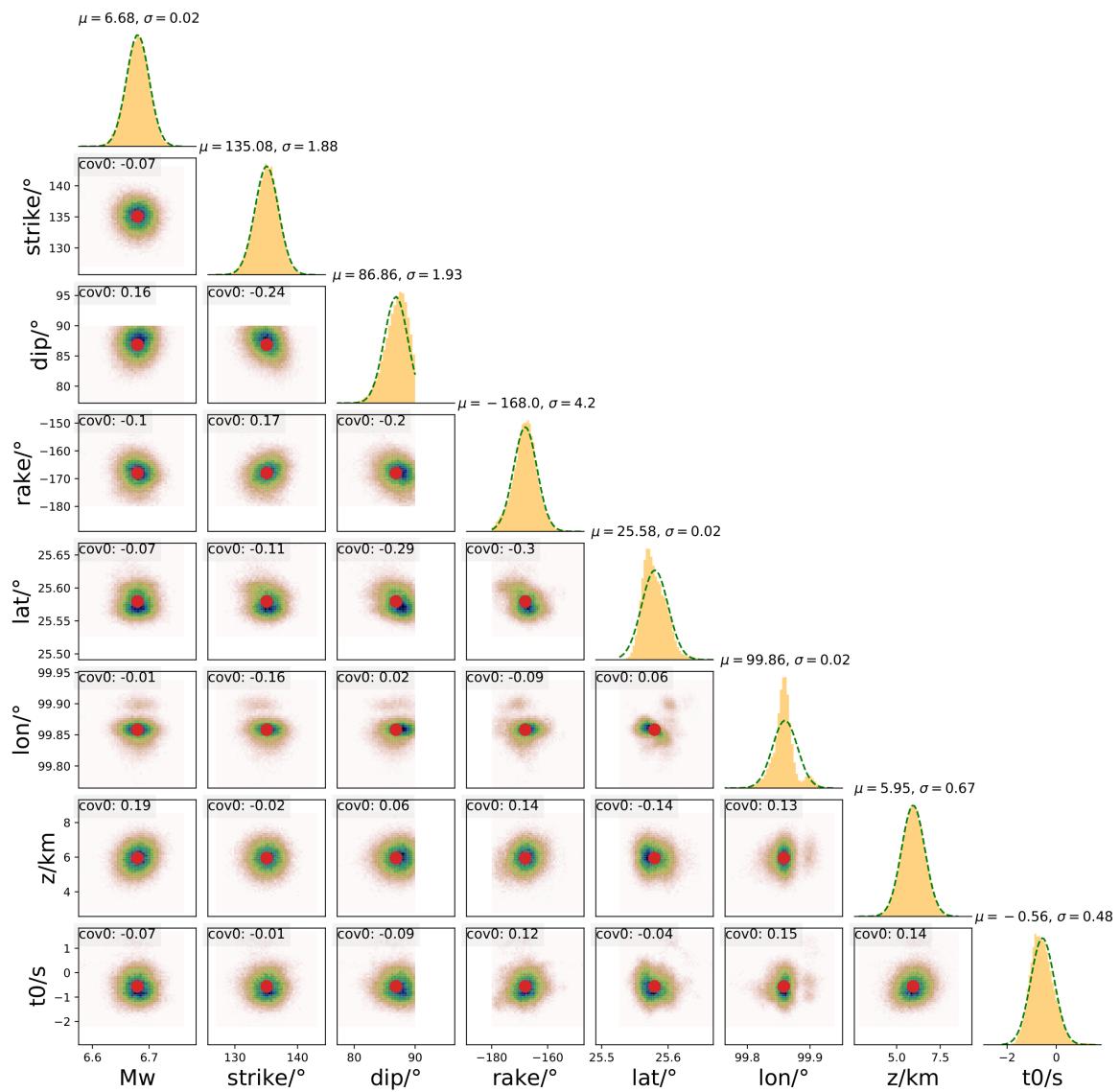
### 3. Inversion of focal mechanism:

```
$ MCMTpy sample MH -c ./sample.json
$ mpirun -n 4 MCMTpy sample MH -c ./sample.json # parallel
```

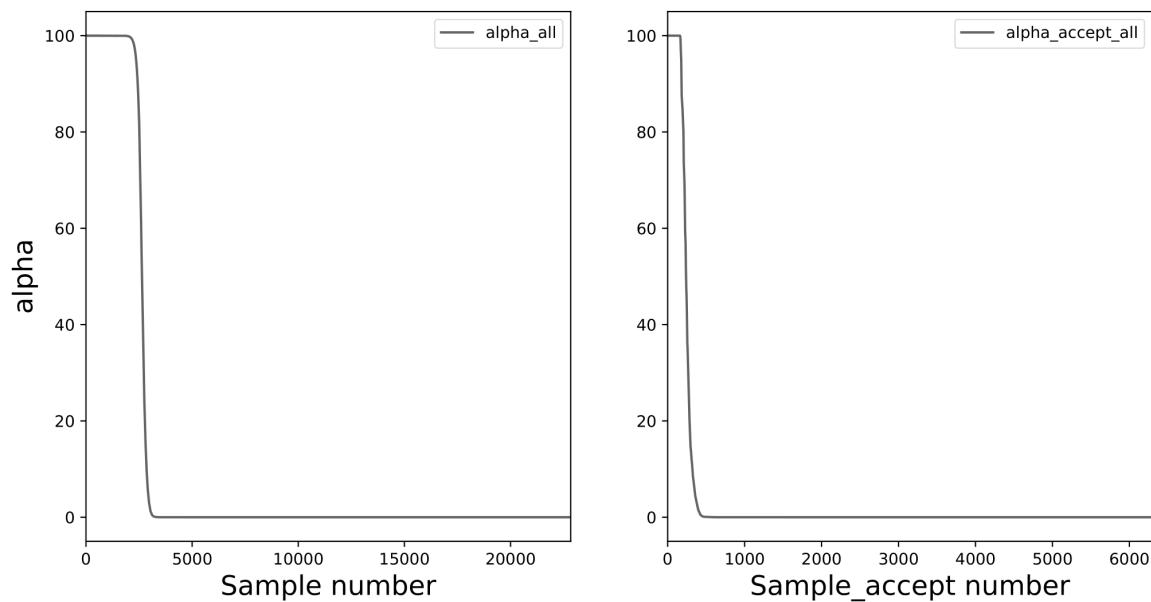
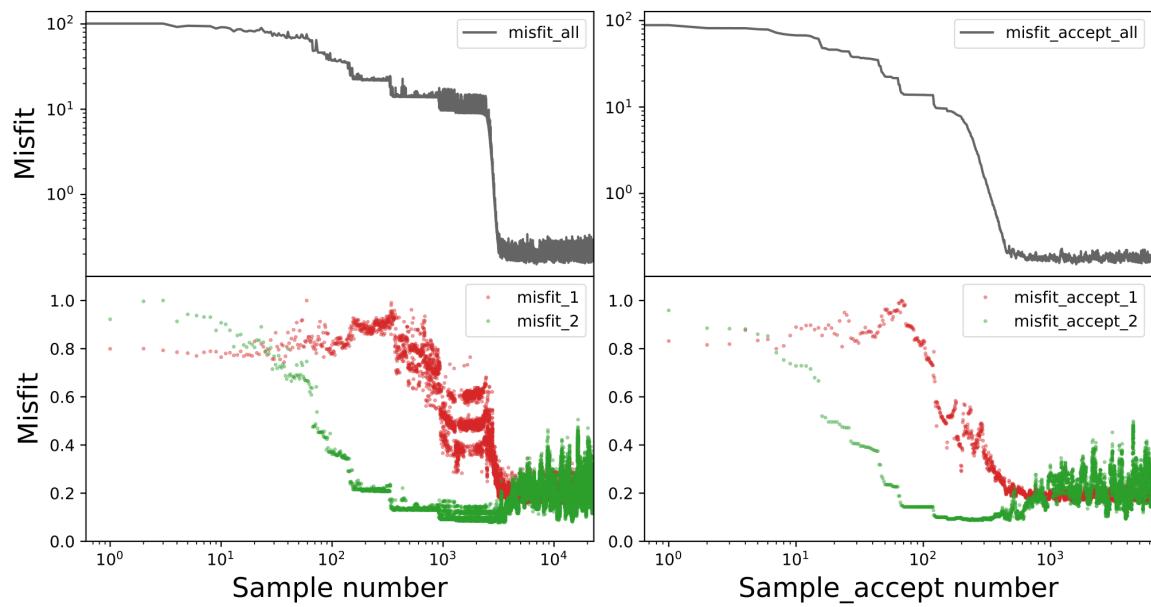
### 4. Result visualization:

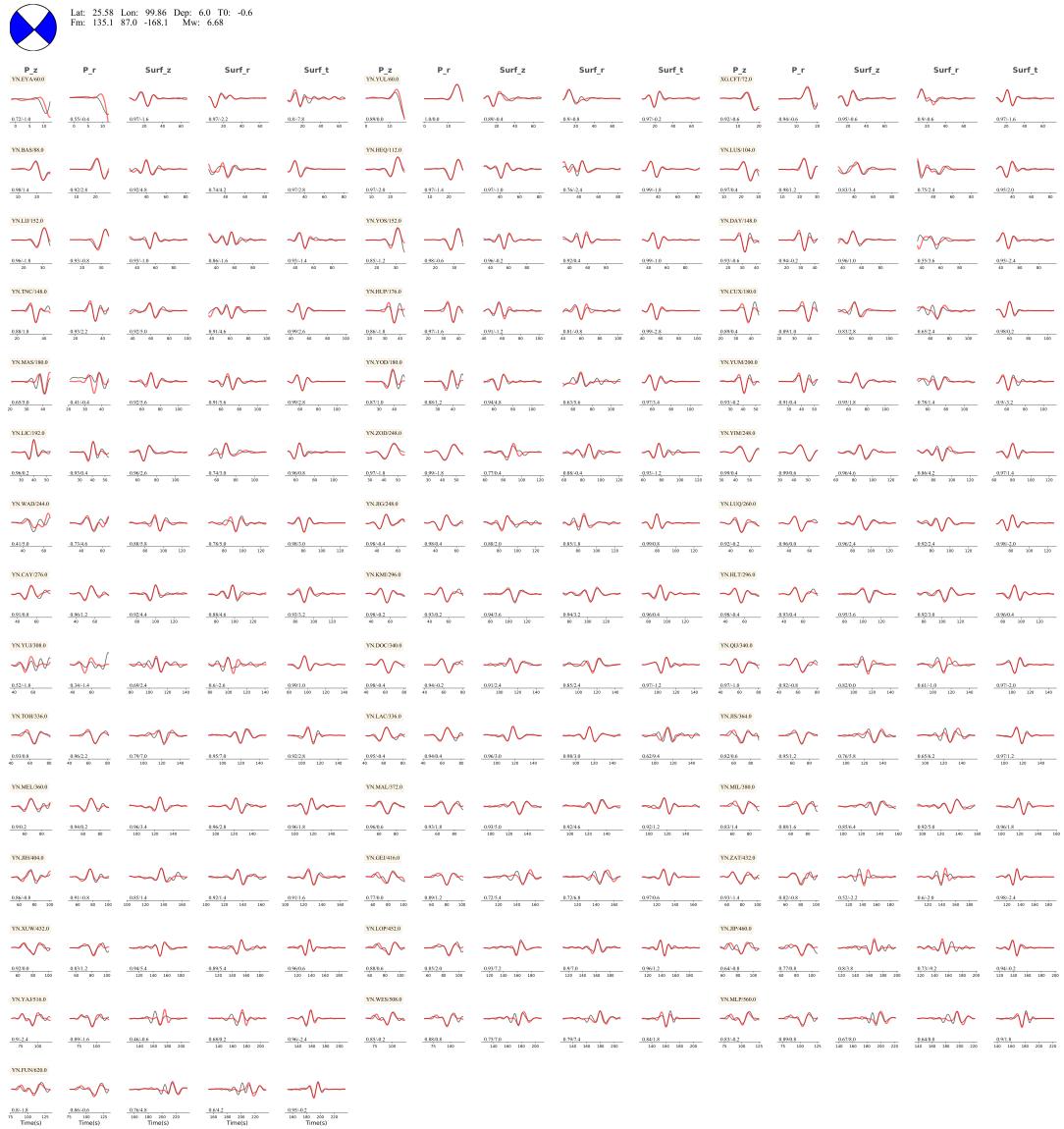
```
$ MCMTpy plot pyfk -c plot.json
```

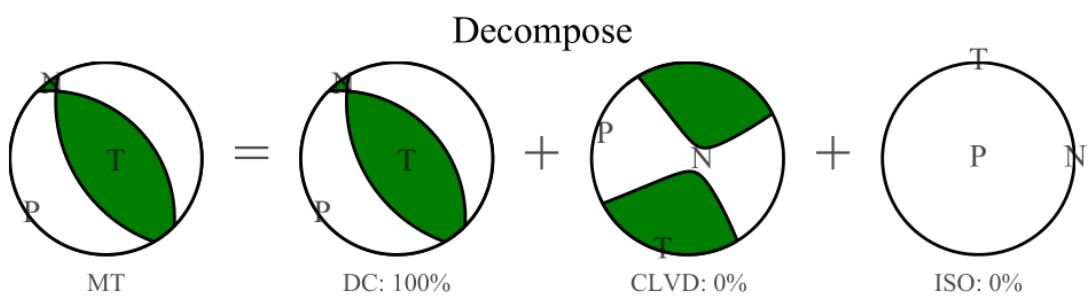
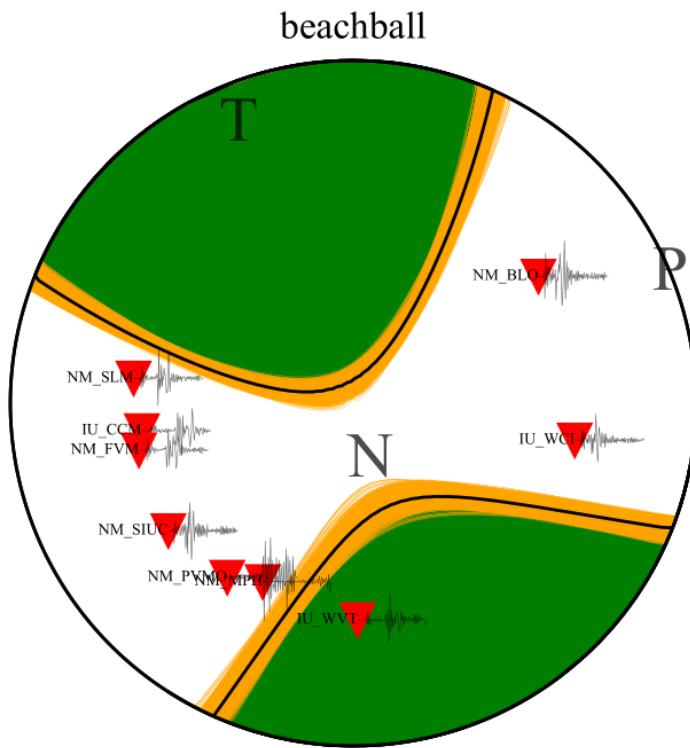




### Misfit with iteration







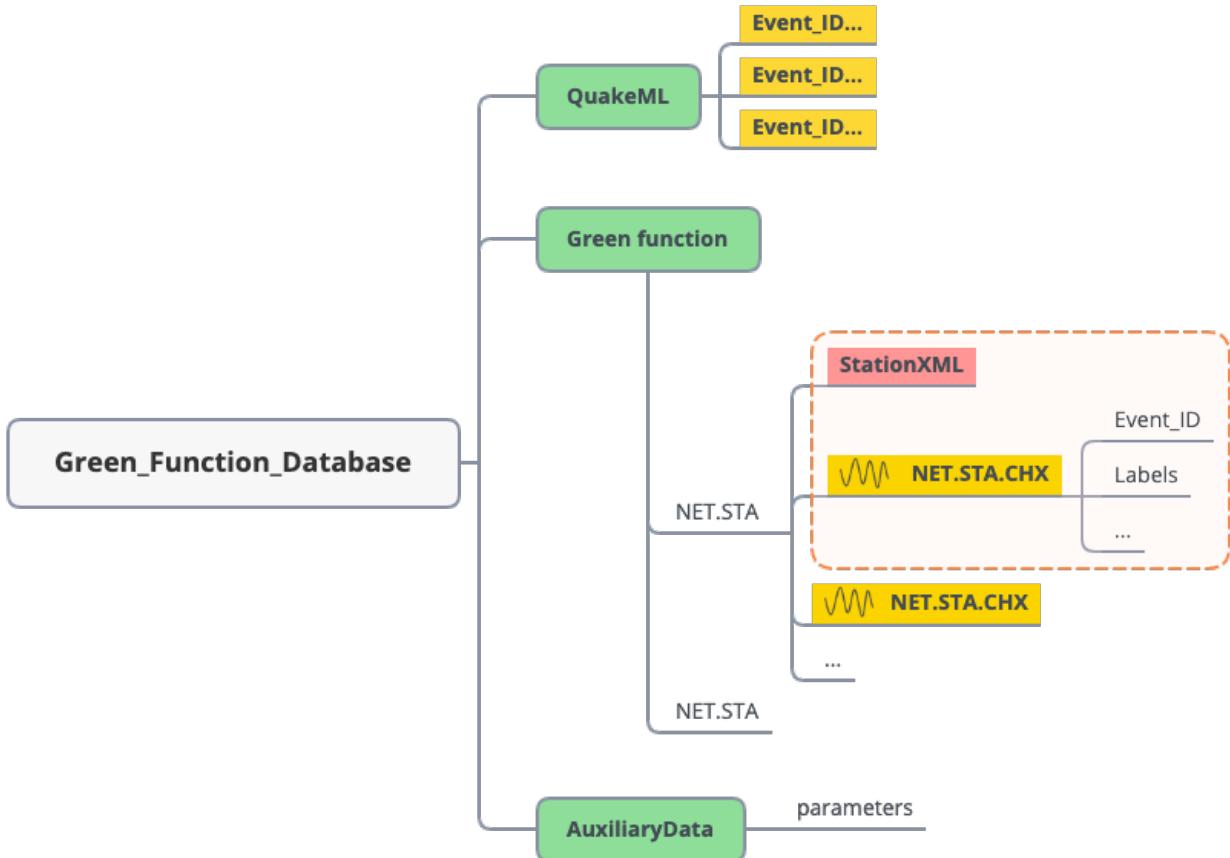
## Tutorial\_Detailed

More details on **MCMTpy** for data processing, parameters choosing can be found in following documentations, and the **jupyter notebook** can be find in `./MCMTpy-master/data/example_yunnan/Jupyter_notebook`.

### Calculate Green Function Database

<b>Station and Source</b>	<b>12</b>
<b>MPI</b>	<b>13</b>
<b>Path</b>	<b>13</b>
<b>SourceModel</b>	<b>14</b>
<b>SeisModel</b>	<b>14</b>
<b>Config</b>	<b>14</b>
<b>Logging file</b>	<b>15</b>
<b>Example</b>	<b>15</b>

- The first step of **MCMTpy** is to build Green's function database. All parameters are written into a JSON file named **build\_GFs.json**. Some brief descriptions of the parameters are included here following the definination.
- We do not recommend using relative paths because of the possibility of errors. **Absolute paths** are preferred.
- We used ASDF seismic data format to manage Green Function database, which is convenient to locally build up a database of pre-processed waveforms. ASDF file greatly reduces the number of green functions files, and a single ASDF file can replace thousands of green function files which may threw us into confusion in the management process. We store the Green's functions corresponding to **Event\_ID (QuakeML)** in “**Waveforms**” of ASDF data. In the “**Labels**” of “**Waveforms**” , we store the phase (P and S) travel times which can be computed in advance by a given velocity model.
- The general structure of MCMTpy stores Green's function with ASDF file format shown in the image below.



- You can also find some useful information about parameters in refers to [pyfk](#).

## Station and Source

Database\_mode

- Set to be **.true.** when you want to compute the Green's function library for some region. For real examples, the GF databases may require up to several GBs of free disc space. In addition you need to set the following parameters correctly:

```
Source_depth_min,           Source_depth_max,           Source_depth_spacing,
Station_distance_radius_min, Station_distance_radius_max, Station_distance_spacing,
Source_name, Network_name, Station_name, Station_depth_reference
```

- Set to be **.false.** when you want to calculate the Green's function for stations at which the source is given. You must provide an .txt file named **Source\_Station\_info.txt**, which including the longitude latitude and depth information of the source and station. Different sources should be separated by blank lines.

- **Source\_Station\_info.txt** format.

```
[source_name] [source_lat] [source_lon] [source_depth]
[network_name] [station_name] [station_lat] [source_lon] [station_depth]
```

```
1 source_1 25.58 99.86 1
2 YN EYA 26.108 99.947 0
3 YN YUL 25.885 99.371 0
4
5 source_2 25.58 99.86 2
6 YN EYA 26.108 99.947 0
7 YN YUL 25.885 99.371 0
8 XG CFT 25.848 100.517 0
```

```
9 YN BAS 25.118 99.146 0
10
11 source_3 25.58 99.86 3
12 YN WES 23.372 104.253 0
13 YN MLP 23.128 104.702 0
14 YN FUN 23.624 105.620 0
```

### Source\_depth\_min

- The minimum source depth in Green's function databases (km).

### Source\_depth\_max

- The maximum source depth in Green's function databases (km).

### Source\_depth\_spacing

- Interval of source depths in Green's function databases (km).

### Station\_distance\_radius\_min

- The minimum distance of the station in Green's function databases (km or degree, depend by degrees).

### Station\_distance\_radius\_max

- The maximum distance of the station in Green's function databases (km or degree, depend by degrees).

### Station\_distance\_spacing

- Horizontal spacing interval in Green's function databases (km or degree, depend by degrees).

### Source\_name

- Source name in Green's function databases, default value is *source*, please do not change. Only lowercase letters are supported.

### Network\_name

- Network name in Green's function databases, default value is *NET*, please do not change.

### Station\_name

- Station name in Green's function databases, default value is *STA*, please do not change.

### Station\_depth\_reference

- Depth of all virtual stations (km).

## Note

When `Database_mode = .false.`, depths can vary from station to station. But when set to be `.true.`, all stations must use the same depth `Station_depth_reference`.

## MPI

### `MPI_n`

- CPU number. When `Database_mode = .false.`, the number of call cores should be less than or equal to the number of source. But when set to be `.true.`, there are no limits.

## Path

### `DATADIR`

- The output path of the Green function databases.

### `DATADIR_splits`

- A subfolder under `DATADIR` for Green's function. Please keep the path relative to `DATADIR`.

## SourceModel

source\_mechanism

- Focal mechanism, not used. Please set it to `.null..`

srcType

- The type of Green's function, dc(double-couple) sf(single-couple) ep(explosion). Please set it to `.dc..`

## SeisModel

velocity\_model

- Path to the velocity model file. And the format:

`thick(km) vs(km/s) vp(km/s) density(g/cm^3) Qs Qn`

1	5.01000000	3.17601610	5.41925121	2.50000000	600.00000000	1200.00000000
2	5.01000000	3.36477536	5.67947907	2.55000000	600.00000000	1200.00000000
3	5.01000000	3.52147424	5.93255556	2.60000000	600.00000000	1200.00000000
4	5.01000000	3.55152174	6.00984300	2.65000000	600.00000000	1200.00000000
5	5.01000000	3.54084541	6.02592110	2.70000000	600.00000000	1250.00000000
6	5.01000000	3.58840419	6.14793478	2.75000000	650.00000000	1300.00000000
7	5.01000000	3.75369968	6.48572061	2.80000000	700.00000000	1350.00000000
8	5.01000000	3.94361514	6.86566667	2.90000000	750.00000000	1400.00000000
9	10.01000000	4.12710548	7.20229388	3.00000000	800.00000000	1500.00000000
10	10.01000000	4.30825765	7.55049678	3.10000000	850.00000000	1600.00000000
11	10.01000000	4.38069968	7.72566425	3.20000000	850.00000000	1700.00000000
12	90.00000000	4.50000000	7.80000000	3.27000000	900.00000000	1800.00000000

flattening

- Set the flatten status of the velocity model, `.true.` or `.false..`

## Config

The following parameters explanation refers to pyfk <<https://github.com/zixiyi/pyfk>>.

npt

- The sampling points is a multiple of 2.

dt

- Sampling interval in seconds

degrees

- Use degrees instead of km, defaults to `.true..` Only when Database\_mode = `.true..`, it can be changed to `.false..`

taper

- Taper applies a low-pass cosine filter at  $fc=(1-taper)*f\_Nquist$ , defaults to 0.3.

filter

- Apply a high-pass filter with a cosine transition zone between freq. f1 and f2 in Hz, defaults to (0, 0).

dk

- The non-dimensional sampling interval of wavenumber, defaults to 0.3.

smth

- Makes the final sampling interval to be dt/smth, defaults to 1.

pmin

- The min slownesses in term of 1/vs\_at\_the\_source, defaults to 0.

pmax

- The max slownesses in term of 1/vs\_at\_the\_source, defaults to 1.

## Tutorial\_Detailed

kmax

- The kmax at zero frequency in term of 1/hs, defaults to 15.

rdep

- Not used now. Please keep the default values of the parameters, defaults to 0 in km.

updn

- The “up” for up-going wave only, “down” for down-going wave only, “all” for both “up” and “down”, defaults to “all”.

samples\_before\_first\_arrival

- The number of points before the first arrival, defaults to 50.

### Logging file

- MCMTpy will generates 2 logging files when it builds GFs database, that is **prepro\_para\_info.txt** and **write\_Source\_Station\_info\_MPI**. They record all the parameter information of GFs database. You can find them in DATADIR path.

### Example

- The **example\_path** need to be changed to your path, and run this notebook to set **build\_GFs.json** file.

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import os
5 import sys
6 import json5
7
8
9 # The root directory of the project
10 example_path = '/Users/yf/3.Project/8.MCMTpy/MCMTpy-master/data/example_yunnan'
11
12
13 #-----#
14 # we expect no parameters need to be changed below
15 #-----#
16 # 1. get notebook path
17 notebook_path = sys.path[0]
18 os.chdir(notebook_path)
19
20 # 2. change path to Green_Funcs
21 os.chdir('../Green_Funcs')
22 print(os.listdir())
23
24 # 3. show build_GFs.json
25 filename = 'build_GFs.json'
26 with open(filename, 'r', encoding='utf-8') as f1:
27     gfs_json=json5.load(f1)
28 f1.close()
29
30 # 4. change parameters with absolute path
31 gfs_json['Source_Station_info'] = os.path.join(example_path, "Green_Funcs/Source_Station_info")
32 gfs_json["DATADIR"] = os.path.join(example_path, "Green_Funcs/GFs1")
33 gfs_json["DATADIR_splits"] = os.path.join(example_path, "Green_Funcs1/GFs1/GFs_splits")
34 gfs_json["Velocity_model"] = os.path.join(example_path, "v_model/v_model_yunnan.txt")
35 gfs_json_new = os.path.join(example_path, 'Green_Funcs/build_GFs_new.json')
36
```

```
37 with open(gfs_json_new, 'w') as f2:
38     json5.dump(gfs_json, f2, indent=2)
39 f2.close()
40
41 # !mpirun -n 4 MCMTpy build_GFs pyfk -c build_GFs_new.json > gfs.log
```

- Now you can run MCMTpy in bash:

```
$ mpirun -n 4 MCMTpy build_GFs pyfk -c build_GFs_new.json > gfs.log
```

### Note

Please follow the above parameter instructions and set all parameters correctly before running the program. Otherwise, it is easy to report an error!

## Synthesize the Test Data

<b>Path</b>	<b>16</b>
<b>Source</b>	<b>16</b>
<b>Station</b>	<b>16</b>
<b>Source Time Function</b>	<b>17</b>
<b>Filter</b>	<b>17</b>
<b>Add Noise</b>	<b>17</b>
<b>Output</b>	<b>17</b>
<b>Example</b>	<b>18</b>

- The synthesized test data can help us check whether the Green's function database is correctly calculated.
- We do not recommend using relative paths because of the possibility of errors. **Absolute paths** are preferred.

### Path

GFs\_json\_file

- The path to the Json-file used to calculate Green Function Database.

### Source

srcType

- The type of Source, mt(moment tensor) dc(double-couple) sf(single-couple) ep(explosion).

point

- The latitude longitude and depth of the source.

source\_mechanism

- Focal mechanism.

- When `srcType = dc`: `source_mechanism = [Mw, Strike, Dip, Rake]`

- When `srcType = mt`: `source_mechanism = [Mw, Mxx, Mxy, Mxz, Myy, Myz, Mzz]`

### Station

NET\_STA

- The information contained in the two-dimensional list: [network\_name] [station\_name] [station\_lat] [source\_lon] [station\_depth]

### Source Time Function

Trapezoid\_stf\_mode

- Whether to use pyfk provided trapezoid shaped source time function. Set to be **.true.**, you need to set the following parameters correctly:

dura rise delta

dura

- Duration time (s).

rise

- Rise time (s).

delta

- The time interval (s).

Stf\_file

- Stf\_file should be an **SAC** file as the source time function when set Trapezoid\_stf\_mode to be **.false..**

### Note

To read the data from the Stf\_file file, keep the sampling rate of SAC data consistent with the data and GFs.

### Filter

filter\_mode

- Whether to filter.

freqmin

- The minimum frequency of filtering.

freqmax

- The maximum frequency of filtering.

### Add Noise

Add\_noise

- Whether add noise to syn.

noise\_level\_waveform

- Noise level added to the waveform.

noise\_level\_arrive\_time

- Noise level added to the phase time as **tp** and **ts**.

### Output

Output\_mode

- Output data or not.

Output\_path

- Path of output data

source\_name

- Event name of output data, you can set it any way you want. Only lowercase letters are supported.

ASDF\_filename

- The name of the output ASDF file

### Example

- The **example\_path** need to be changed to your path, and run this notebook to set **svn.json** file.

```

1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import os
5 import sys
6 import json5
7 import shutil
8
9 # The root directory of the project
10 example_path = '/Users/yf/3.Project/8.MCMTpy/MCMTpy-master/data/example_yunnan'
11
12 #-----#
13 # we expect no parameters need to be changed below
14 #-----#
15 # change path to Green_Funcs
16 notebook_path = sys.path[0]
17 os.chdir(notebook_path)
18 os.chdir('../syn')
19 print(os.listdir())
20
21 # show build_GFs.json
22 filename = 'syn.json'
23 with open(filename, 'r', encoding='utf-8') as f1:
24     syn_json=json5.load(f1)
25
26 # change parameters with absolute path
27 syn_json["GFS_json_file"] = os.path.join(example_path, "Green_Funcs/build_GFs_new.json")
28 syn_json["Stf_file"] = os.path.join(example_path, "syn/Stf_file/Stf_file.sac")
29 syn_json["Output_path"] = os.path.join(example_path, "syn/Synthetic")
30 syn_json_new = os.path.join(example_path, 'syn/syn_new.json')
31
32 with open(syn_json_new, 'w') as f2:
33     json5.dump(syn_json, f2, indent=2)
34 f2.close()
35
36 shutil.rmtree('./Synthetic')
37 # !MCMTpy syn pyfk -c ./syn_new.json > syn.log

```

- Now you can run MCMTpy in bash:

```
$ MCMTpy syn pyfk -c ./syn.json
```

- View svn waveform:

```

1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import os
5 import pyasdf
6
7

```

```

8 # The root directory of the project
9 example_path = '/Users/yf/3.Project/8.MCMTpy/MCMTpy-master/data/example_yunnan'
10
11 #-----#
12 # we expect no parameters need to be changed below
13 #-----#
14 ## change the path
15 h5_path = os.path.join(example_path, 'syn/Synthetic/SYN_test.h5')
16 source_name = 'source_syn'
17
18 ### read h5 data --> Sta_data (dict)
19 ds_raw=pyasdf.ASDFDataSet(h5_path,mpi=False,mode='r')
20 Station_list = ds_raw.waveforms.list()
21 Station_num = len(Station_list)
22
23 Sta_data={}
24 for i in range(0,Station_num,1):
25     info={}
26     tags = ds_raw.waveforms[Station_list[i]].get_waveform_tags()
27     if source_name in tags:
28         raw_sac = ds_raw.waveforms[Station_list[i]][source_name]
29         tp = float(ds_raw.waveforms[Station_list[i]][source_name][0].stats.asdf['labels'])
30         ts = float(ds_raw.waveforms[Station_list[i]][source_name][0].stats.asdf['labels'])
31         info.update({ "tp": tp})
32         info.update({ "ts": ts})
33         info.update({ "data": raw_sac})
34         Sta_data.update({ Station_list[i]: info})
35
36 # plot data
37 ax = Sta_data['YN.YUM'][ 'data'].plot()
38 Sta_data['YN.YUM'][ 'data'][0].stats

```

**Note**

Please follow the above parameter instructions and set all parameters correctly before running the program. Otherwise, it is easy to report an error!

**Prepare Data for MCMTpy****Format of MCMTpy input data:**

19

**Example**

20

**Format of MCMTpy input data:**

- **ASDF** file format, and the sampling rate **dt** is consistent with that of GFs database.
- Must be **ENZ** three-component data and the name of the channel must be 'E'/N'/Z' order.

**Note**

Pyasdf files are automatically sorted by name. So you must make sure that the order of the Trace is still ENZ after the automatic sorting, generally the name of the channel 'BHE'/'BHN'/'BHZ' will be fine.

- Trace.Stats.sac.t1/Trace.Stats.sac.t2 must be included in the trace header file to represent the P/S phase time.
- The trace header file must include the **station** and **network** name.
- The trace header file must include **b/e/o/stla/stlo/stdp/evlo/evla/evdp/mag/dist/az/baz**, time information, event longitude and latitude information, azimuth information.
- The trace header file must include **starttime/endtime**.
- All traces need to retain a uniform number of sampling points before T1, such as **p\_n0 = 50**.

## Example

- We have provided some useful scripts for data preprocessing, and the **example\_path** need to be changed to your path, and run this notebook.

```

1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import os
5 import glob
6 import obspy
7 import shutil
8 from obspy import Stream
9 from obspy.taup import TauPyModel
10 from MCMTpy.utils.asdf_function import Add_waveforms,Add_stationxml,get_QuakeML,Add_quak
11 from MCMTpy.utils.asdf_function import get_StationXML
12
13 #-----
14 # 1.read raw data
15 def read_data(allfiles_path):
16
17     allfiles = sorted( glob.glob(allfiles_path) )
18     data_raw = Stream()
19     for i in range(0,len(allfiles),1):
20         try:
21             tr = obspy.read(allfiles[i])
22             data_raw += tr
23         except Exception:
24             print(allfiles[i],': no such file or obspy read error');continue
25     data = data_raw.copy()
26
27     return data
28
29 #-----
30 # 2.get_taup_tp_ts
31 def get_taup_tp_ts(model,depth,distance,degree=None):
32     if degree==False:
33         distance = distance/111.19
34
35     time_p = model.get_travel_times(source_depth_in_km=depth,
36                                     distance_in_degree=distance,
37                                     phase_list=[ "P", "P" ])
38
39     time_s = model.get_travel_times(source_depth_in_km=depth,
40                                     distance_in_degree=distance,
41                                     phase_list=[ "S", "S" ])
42
43     ray_p = time_p[0].ray_param
44     tp = time_p[0].time
45     angle_p = time_p[0].incident_angle

```

```

46
47     ray_s = time_s[0].ray_param
48     ts = time_s[0].time
49     angle_s = time_s[0].incident_angle
50
51     return ray_p, tp, angle_p, ray_s, ts, angle_s
52
53 #-----#
54 # 3. preprocess
55 def preprocess(data,freqmin,freqmax,samp_freq,p_n0,npts):
56
57     data.detrend(type='demean')
58     data.detrend(type='simple')
59     data.taper(max_percentage=0.05)
60
61     data.filter('bandpass', freqmin=freqmin, freqmax=freqmax, corners=4, zerophase=True)
62     data.resample(samp_freq,window='hanning',no_filter=True, strict_length=False)
63
64     for i in range(0,len(data),1):
65         b = data[i].stats.sac['b']
66         t1 = data[i].stats.sac['t1']
67         t_start = data[i].stats starttime + (t1-b) - p_n0/samp_freq
68         t_end_npts = t_start+npts/samp_freq
69         t_endtime = data[i].stats.endtime
70         if t_end_npts > t_endtime:
71             t_end = t_endtime
72         else:
73             t_end = t_end_npts
74
75         data[i].trim(t_start, t_end)
76
77     for i in range(0,round(len(data)/3)):
78         t_start_1 = data[3*i].stats.starttime
79         t_start_2 = data[3*i+1].stats.starttime
80         t_start_3 = data[3*i+2].stats.starttime
81         t_end_1 = data[3*i].stats.endtime
82         t_end_2 = data[3*i+1].stats.endtime
83         t_end_3 = data[3*i+2].stats.endtime
84         t_start = max(t_start_1,t_start_2,t_start_3)
85         t_end = min(t_end_1,t_end_2,t_end_3)
86         data[3*i:3*i+3].trim(t_start, t_end)
87
88     # velocity(nm/s) to velocity(cm/s)
89     for i in range(0,len(data),1):
90         data[i].data = 1e-6*data[i].data
91
92 #-----#
93 # 4.write_ASDF
94 def write_ASDF(data_enz,Output_path,source_name,ASDF_filename):
95     source_lat    = data_enz[0].stats.sac.evla
96     source_lon    = data_enz[0].stats.sac.evlo
97     source_depth  = data_enz[0].stats.sac.evdp
98     source_time   = data_enz[0].stats starttime + data_enz[0].stats.sac.b
99
100    Output_file_path = os.path.join(Output_path, ASDF_filename+'.h5')
101    catalog_create   = get_QuakeML(source_name, source_lat, source_lon, source_depth, sou
102    Add_quakeml(Output_file_path,catalog_create)
103
104    Station_num = round(len(data_enz)/3)
105    for i in range(0,Station_num,1):

```

```

106     net_sta_name      = data_enz[3*i].stats.network+'_'+data[3*i].stats.station
107     station_network   = data_enz[3*i].stats.network
108     station_name      = data_enz[3*i].stats.station
109     station_lat       = data_enz[3*i].stats.sac.stla
110     station_lon       = data_enz[3*i].stats.sac.stlo
111     station_depth     = 0
112     station_distance   = data_enz[3*i].stats.sac.dist
113     station_az        = data_enz[3*i].stats.sac.az
114     station_baz       = data[3*i].stats.sac.baz
115     tp                = data_enz[3*i].stats.sac.t1
116     ts                = data_enz[3*i].stats.sac.t2
117     stream            = data_enz[3*i:3*i+3].copy()
118     starttime         = data_enz[3*i].stats.starttime
119     time_before_first_arrival = p_n0/samp_freq
120
121     inv=get_StationXML(station_network,
122                           station_name,
123                           station_lat,
124                           station_lon,
125                           station_depth,
126                           'enz')
127
128     Add_waveforms([stream],
129                   catalog_create,
130                   Output_file_path,
131                   source_name,
132                   tp,
133                   ts,
134                   station_distance,
135                   station_az,
136                   station_baz)
137
138     Add_stationxml(inv,
139                      Output_file_path)
140
141     for j in range(0,3,1):
142         CHN = stream[j].stats.channel
143         sac_filename = os.path.join(Output_path,source_name+'.'+station_network+'.'
144                                     stream[j].write(sac_filename, format='SAC'))
145 #-----#
146
147
148
149
150 #-----#
151 # 0. main
152 # The root directory of the project
153 example_path = '/Users/yf/3.Project/8.MCMTpy/MCMTpy-master/data/example_yunnan'
154 freqmin      = 0.005                                # pre filtering frequency bandwidth
155 freqmax      = 2                                    # note this cannot exceed Nyquist fre
156 samp_freq    = 5                                    # targeted sampling rate (hz)
157 p_n0         = 50                                   # number of sampling points before P
158 npts         = 2048                                 # data length
159 Output_path   = os.path.join(example_path, "YN.202105212148_Inv/YN.202105212148_MCMTpy")
160 source_name   = "source_enz"                         # asdf's source_tag
161 ASDF_filename = "YN.202105212148"                  # asdf filename
162
163
164
165

```

```

166
167 #-----#
168 # we expect no parameters need to be changed below
169 #-----#
170 # 1. read raw data
171 allfiles_path = os.path.join(example_path, 'YN.202105212148_Inv/YN.202105212148_raw/*.SAC')
172 data = read_data(allfiles_path)
173
174 # 2. taup ray tracing
175 model_path = os.path.join(example_path, "v_model/v_model.npz")
176 model = TauPyModel(model=model_path)
177 for i in range(0,len(data),1):
178     depth = data[i].stats.sac['evdp']
179     distance = data[i].stats.sac['dist']
180     ray_p, tp, angle_p, ray_s, ts, angle_s = get_taup_tp_ts(model, depth, distance, degree=False)
181     data[i].stats.sac["t1"] = tp
182     data[i].stats.sac["t2"] = ts
183
184 # 3. preprocess
185 preprocess(data,freqmin,freqmax,samp_freq,p_n0,npts)
186
187 # 4. output ASDF and SAC data
188 shutil.rmtree(Output_path)
189 os.mkdir(Output_path)
190 write_ASDF(data,Output_path,source_name,ASDF_filename)

```

## Note

Please follow the above parameter instructions and set all parameters correctly before running the program. Otherwise, it is easy to report an error!

## Inversion of Source

<b>Path</b>	<b>23</b>
<b>Source Time Function</b>	<b>24</b>
<b>MPI and Chains_n</b>	<b>24</b>
<b>Misfit</b>	<b>24</b>
<b>Inversion parameters</b>	<b>26</b>
<b>Alpha</b>	<b>27</b>
<b>Logging file</b>	<b>28</b>
<b>Example Double-Couple Inv</b>	<b>28</b>

- This is the core of the MCMTpy, and all parameters are written into a JSON file named **sample\_dc.json**. Some brief descriptions of the parameters are included here following the definination.
- We do not recommend using relative paths because of the possibility of errors. **Absolute paths** are preferred.

## Path

GFs\_json\_file

- The path to the Json-file used to calculate Green Function Database.

GFs\_file

## Tutorial\_Detailed

- The path to the Green Function Database.

Raw\_data\_file

- The path to the Raw data ASDF file.

Source\_tag

- Raw data ASDF file's source name which setted in the data preprocess function.

Raw\_p\_n0

- The number of points before the first arrival. Be consistent with the GFs' samples\_before\_first\_arrival.

Dt

- Sampling rate of the raw data.

Output\_path

- Inversion result output file path.

## Source Time Function

Trapezoid\_stf\_mode

- Whether to use pyfk provided trapezoid shaped source time function. Set to be **.true.** you need to set the following parameters correctly:

dura rise delta

dura

- Duration time (s).

rise

- Rise time (s).

delta

- The time interval (s).

Stf\_file

- Stf\_file should be an **SAC** file as the source time function when set Trapezoid\_stf\_mode to be **.false..**

## Note

To read the data from the Stf\_file file, keep the sampling rate of SAC data consistent with the data and GFs.

## MPI and Chains\_n

MPI\_n

- CPU number

Chains\_n

- A total of n Markov Chains are sampled on each core (CPU).

N

- Each Markov Chains' sampling number.

## Misfit

Geometrical\_spreading\_mode

- Whether the geometric diffusion correction is turned on or not. Waveform normalization is used when it set to be **.false.**, and please set the magnitude **M0** not to be inverted in **Fixed\_FM**. Only one of **Geometrical\_spreading\_mode** and **Amplitude\_normalization\_mode** can be **.true.**.

Distance\_0

- The reference epicentral distance when **Geometrical\_spreading\_mode** set to be **.true.**.

Amplitude\_normalization\_mode

- Waveform normalization mode. Only one of **Geometrical\_spreading\_mode** and **Amplitude\_normalization\_mode** can be **.true.**.

Use\_file\_mode

- When set it to be **.false.**, don't need to provide **Raw\_data\_inv\_info** file. Setting following parameters (**Raw\_data\_inv\_info\_writed**, **NET\_STA**, **P\_inv\_comp**, **S\_inv\_comp**, **Surf\_inv\_comp**, **P\_win**, **S\_win**, **Surf\_win**, **Phase\_weight**, **Distance\_weight**, **P\_maxlag**, **S\_maxlag**, **Surf\_maxlag**, **P\_filter**, **S\_filter**, **Surf\_filter**) correctly, **MCMTpy** will generate this file automatically, so you can change individual parameter values and use this file for inversion. **Raw\_data\_inv\_info\_writed** is the filename that generate this file automatically. When set it to be **.true.**, **Raw\_data\_inv\_info** is needed which is the path to inversion file.

Raw\_data\_inv\_info

- The path to inversion parameters file. It includes the time window length, filtering range, weight and other parameters used for data. When set **Use\_file\_mode** to be **.true.**, it must be provided.
- **Raw\_data\_inv\_info.txt** format.

```

1 YN EYA
2 Lat_lon_depth      26.108801   99.947502   0.000000
3 P_inv_comp         yes        yes        no
4 S_inv_comp         no         no         no
5 Surf_inv_comp      yes        yes        yes
6 P_win              -10.00     3.40
7 S_win              -5.00      25.20
8 Surf_win           -12.00     50.20
9 Phase_weight       1.0        1.0        1.0
10 Distance_weight   1.0
11 P_maxlag          5.0000
12 S_maxlag          15.0000
13 Surf_maxlag       10.0000
14 P_filter          0.010     0.20
15 S_filter          0.010     0.100
16 Surf_filter        0.005     0.100
17
18 YN YUL
19 Lat_lon_depth      25.885401   99.371696   0.000000
20 P_inv_comp         yes        yes        no
21 S_inv_comp         no         no         no
22 Surf_inv_comp      yes        yes        yes
23 P_win              -10.00     6.20
24 S_win              -5.00      25.20
25 Surf_win           -12.00     50.20
26 Phase_weight       1.0        1.0        1.0
27 Distance_weight   1.0
28 P_maxlag          5.0000
29 S_maxlag          15.0000
30 Surf_maxlag       10.0000
31 P_filter          0.010     0.200
32 S_filter          0.010     0.100
33 Surf_filter        0.005     0.100

```

Raw\_data\_inv\_info\_writed

## Tutorial\_Detailed

- Raw\_data\_inv\_info\_writed is the filename that generate this file automatically When set Use\_file\_mode to be **true**..

NET\_STA

- The information contained in the two-dimensional list: [network\_name] [station\_name] [station\_lat] [source\_lon] [station\_depth]

P\_inv\_comp

- P wave Z/R/T component. ‘yes’: inversion. ‘no’: no inversion. Select the component you want to invert.

S\_inv\_comp

- S wave Z/R/T component. ‘yes’: inversion. ‘no’: no inversion. Select the component you want to invert.

Surf\_inv\_comp

- Surf wave Z/R/T component. ‘yes’: inversion. ‘no’: no inversion. Select the component you want to invert.

P\_win

- P wave P\_win=[-1,2] unit/s; The P wave fitting band, and ‘-1’ means 1 second before the P wave, ‘2’ means 2 second after the P wave

S\_win

- S wave S\_win...

Surf\_win

- Surf wave Surf\_win...

Phase\_weight

- P/S/Surface wave inversion weight

Distance\_weight

- The weight of each station in NET\_STA

P\_maxlag

- The maximum number of moveable sampling points to fitting waveform of P wave. It is generally half the period of the P phase waveform. unit/s.

S\_maxlag

- The maximum number of moveable sampling points to fitting waveform of S wave.

Surf\_maxlag

- The maximum number of moveable sampling points to fitting waveform of Surf wave.

P\_filter

- P wave filter range, [freqmin, freqmax].

S\_filter

- S wave filter range, [freqmin, freqmax].

Surf\_filter

- Surf wave filter range, [freqmin, freqmax].

### Inversion parameters

InvType

- The type of source, mt(moment tensor) dc(double-couple) sf(single-couple) ep(explosion).

FM0

- The initial value of the inversion, it's a 1-d list.

## Tutorial\_Detailed

- dc:[m0, str, dip rake, lat, lon, depth, t0] || mt:[m0, m\_xx, m\_xy, m\_xz, m\_yy, m\_yz, m\_zz, lat, lon, depth, t0] || ep:[m0]
- The default initial value of t0 is usually 0.

### Fixed\_FM

- The length of the list corresponds to FM0, and ‘constant’ means fixed to a constant and unchanged in the inversion process. ‘variable’ is represented as a variable and participates in the inversion.

### FM\_boundary

- A 2-d list, FM’s boundary.

### Noise\_level\_waveform

- Noise level estimation of raw data.

### MISFIT0

- The initial error of the objective function, usually set to a very large value.

### alpha\_max

- The initial value of **alpha**.

### alpha\_min

- The final value of **alpha**.

### a

- The alpha function’s parameter.

### b

- The alpha function’s parameter.

### sigma\_mw

- The standard deviation of the new solution of **Mw**.

### sigma\_mt

- The standard deviation of the new solution of **Moment Tensor**.

### sigma\_dc

- The standard deviation of the new solution of **Strike Dip Rake**.

### sigma\_sf

- The standard deviation of the new solution of **Single Couple** (not test now).

### sigma\_ep

- The standard deviation of the new solution of **Explosion** (not test now).

### sigma\_lat\_lon

- The standard deviation of the new solution of **Latitude and Longitude** (degree).

### sigma\_depth

- The standard deviation of the new solution of **Source Depth** (km).

### sigma\_t

- The standard deviation of the new solution of **T0** (s).

## Alpha

- The **alpha** function is shown in following script, you can test different parameters alpha\_max alpha\_min a b and plot the figure.

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
```

```

3 import math
4 import numpy as np
5 import matplotlib.pyplot as plt
6
7 # parameters
8 N=5e3
9 a=1000                      # 0.2*N
10 b=100                        # 0.04*N
11 alpha_max=100
12 alpha_min=0
13
14 #-----#
15 # we expect no parameters need to be changed below
16 #-----#
17 tt=np.linspace(0, round(N), num=round(N))
18 yy=np.zeros(len(tt))
19 for i in range(0,len(tt)):
20     yy[i]=(alpha_max-alpha_min)*(a+math.exp((tt[0]-a)/b))/(a+math.exp((tt[i]-a)/b))+alpha
21
22 fig, axs = plt.subplots(1,1)
23 axs.plot(tt,yy, linestyle='-', color='red', lw=3, alpha=0.6)
24 axs.set_xlabel("Sample Number", fontsize=17)
25 axs.set_ylabel("Alpha Value", fontsize=17)
26 fig.show()

```

## Logging file

- MCMTpy will generates 1 logging files when it do **Inversion**, that is **Inv\_para\_info.txt**. They record all the parameter information of inversion process. You can find them in **Output\_path** path.

## Example Double-Couple Inv

- The **example\_path** need to be changed to your path, and run this notebook to set **sample\_dc.json** file.

```

1#!/usr/bin/env python3
2# -*- coding: utf-8 -*-
3
4import os
5import sys
6import json5
7
8
9# The root directory of the project
10example_path = '/Users/yf/3.Project/8.MCMTpy/MCMTpy-master/data/example_yunnan'
11
12
13#-----#
14# we expect no parameters need to be changed below
15#-----#
16# 1. get notebook path
17notebook_path = sys.path[0]
18os.chdir(notebook_path)
19# change path to Green_Funcs
20os.chdir('../YN.202105212148_Inv/dc_inv')
21
22# 2.show build_GFs.json
23filename = 'sample_dc.json'
24with open(filename, 'r', encoding='utf-8') as f1:
25    sample_dc_json=json5.load(f1)

```

```

26
27 # 3.change parameters with absolute path
28 sample_dc_json[ "GFS_json_file" ] = os.path.join(example_path, "Green_Funcs/build_GFs_new.json")
29 sample_dc_json[ "GFS_file" ] = os.path.join(example_path, "Green_Funcs/GFs/GFs_splits" )
30 sample_dc_json[ "Raw_data_file" ] = os.path.join(example_path, "YN.202105212148_Inv/YN.202105212148_In")
31 sample_dc_json[ "Output_path" ] = os.path.join(example_path, "YN.202105212148_Inv/dc_inv/Out")
32 sample_dc_json[ "Raw_data_inv_info" ] = os.path.join(example_path, "YN.202105212148_Inv/dc_inv/Out")
33 sample_dc_json[ "Raw_data_inv_info_writed" ] = os.path.join(example_path, "YN.202105212148_Inv/dc_inv/Out")
34 sample_dc_json_new = os.path.join(example_path, 'YN.202105212148_Inv/dc_inv/sample_dc_new.json')
35
36 sample_dc_json[ "MPI_n" ] = 4                                     # CPU num
37 sample_dc_json[ "Chains_n" ] = 4                                  # Each MK-chains num
38 sample_dc_json[ "N" ] = 1e1                                      # each chain's search number
39
40 sample_dc_json[ "alpha_max" ] = 100                                # The initial value of alpha
41 sample_dc_json[ "alpha_min" ] = 0                                   # The final value of alpha
42 sample_dc_json[ "a" ] = 10
43 sample_dc_json[ "b" ] = 10
44
45 with open(sample_dc_json_new, 'w') as f2:
46     json.dump(sample_dc_json, f2, indent=2)
47 f2.close()
48
49 # 4.run MCMTpy in shell
50 os.chdir(notebook_path)
51 os.chdir('../YN.202105212148_Inv/dc_inv')
52 # !MCMTpy sample MH -c ./sample_dc_new.json > sample.log

```

- Now you can run MCMTpy in bash:

```
$ MCMTpy sample MH -c ./sample_dc_new.json > sample.log
```

## Note

Please follow the above parameter instructions and set all parameters correctly before running the program. Otherwise, it is easy to report an error!

## Result Visualization

<b>Path</b>	<b>29</b>
<b>Hist</b>	<b>30</b>
<b>Misfit</b>	<b>30</b>
<b>Waveform</b>	<b>30</b>
<b>Example</b>	<b>31</b>

- This is the plotting script of the MCMTpy, and all parameters are written into a JSON file named **plot.json**. Some brief descriptions of the parameters are included here following the definition.
- We do not recommend using relative paths because of the possibility of errors. **Absolute paths** are preferred.

## Path

plot\_output\_path,

- Plot result output file path.

Inv\_json\_file,

- The path to the Json-file used in inversion.

fig\_format,

- Output figure format.

### Hist

plot\_hist

- Whether to plot hist-figure.

N\_start

- The sequence number of all-FM Markov chain to plot in hist-figure.

N\_start\_accept

- The sequence number of all-accept-FM Markov chain to plot in hist-figure.

num\_bins

- Number of grids to draw in hist-figure.

num\_std

- Range of axes in each subgraph (mean +- several times standard deviation). num\_std means  $n$  times standard deviation.

labels\_name

- Labels' name, it's depend on the number parameters that you want to inverse.

• eg: dc-> ['mw', 'strike/o', 'dip/o', 'rake/o', 'x/km', 'y/km', 'z/km', 't0/s'].

• eg: mt-> ['Mw', 'Mxx', 'Myy', 'Mzz', 'Mxy', 'Mxz', 'Myz', 'x/km', 'y/km', 'z/km', 't0/s'].

### Misfit

plot\_misfit

- Whether to plot misfit-figure.

MPI\_n\_st

- Which MPI's misfit do you want to draw in figure.

Chains\_n\_st

- Which Chains's misfit do you want to draw in figure.

### Note

We can only specify a chain of an MPI to draw misfit-figure.

### Waveform

plot\_waveform

- Whether to plot waveform-figure.

NET\_STA

- The information contained in the two-dimensional list: [network\_name] [station\_name] [station\_lat] [source\_lon] [station\_depth]

FM\_best

- The best focal mechanism, it's a 1-d list.

• dc:[m0, str, dip rake, lat, lon, depth, t0] || mt:[m0, m\_xx, m\_xy, m\_xz, m\_yy, m\_yz, m\_zz, lat, lon, depth, t0] || ep:[m0]

```
line_n_sta
```

- Draw several stations per row in waveform-figure.

```
max_p_ylim
```

- The maximum amplitude of P wave drawing in waveform-figure.

```
max_s_ylim
```

- The maximum amplitude of S wave drawing in waveform-figure.

```
max_surf_ylim
```

- The maximum amplitude of Surf wave drawing in waveform-figure.

```
plot_comp
```

- Which components of the waveform that you want to plot in waveform-figure.

- The 2-d list represent the three components of P, S and Surf, respectively ZRT.

```
plot_comp_name
```

- Labels' name, it's depend on the number parameters that you want to plot with `plot_comp`.

## Example

- The `example_path` need to be changed to your path, and run this notebook to set `sample_dc.json` file.

```

1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import os
5 import sys
6 import json5
7
8 # The root directory of the project
9 example_path = '/Users/yf/3.Project/8.MCMTpy/MCMTpy-master/data/example_yunnan'
10
11 #-----#
12 # we expect no parameters need to be changed below
13 #-----#
14 # 1. get notebook path
15 notebook_path = sys.path[0]
16 os.chdir(notebook_path)
17
18 # 2.change path to Green_Funcs
19 os.chdir(notebook_path)
20 os.chdir('../YN.202105212148_Inv/dc_inv')
21 print(os.listdir())
22
23 # 3.read plot_dc.json
24 filename = 'plot_dc.json'
25 with open(filename, 'r', encoding='utf-8') as f1:
26     plot_dc_json=json5.load(f1)
27
28 # 4.change parameters with absolute path
29 plot_dc_json['plot_output_path'] = os.path.join(example_path, "YN.202105212148_Inv/dc_inv")
30 plot_dc_json['Inv_json_file'] = os.path.join(example_path, "YN.202105212148_Inv/dc_inv/sam")
31 plot_dc_json_new = os.path.join(example_path, 'YN.202105212148_Inv/dc_inv/plot_dc_new.json')
32
33 # 5.hist
34 plot_dc_json['plot_hist'] =True
35 plot_dc_json['N_start'] = 0
36 plot_dc_json['N_start_accept'] = 1

```

```

37 plot_dc_json[ "num_bins" ] = 50
38 plot_dc_json[ "num_std" ] = 5
39 plot_dc_json[ "labels_name" ] = [ 'Mw', 'strike/°', 'dip/°', 'rake/°', 'z/km' ]
40
41 # 6. misfit
42 plot_dc_json[ "plot_misfit" ] = True
43 plot_dc_json[ "MPI_n_st" ] = 0
44 plot_dc_json[ "Chains_n_st" ] = 0
45
46 # 7. waveform
47 plot_dc_json[ "plot_waveform" ] = True
48 plot_dc_json[ "FM_best" ] = [ 6.68, 135.1, 87.0, -168.1, 25.58, 99.86, 5.95, -0.57
49 plot_dc_json[ "line_n_sta" ] = 3 # Draw the data of several stations
50 plot_dc_json[ "max_p_ylim" ] = 1 # max amp y-axis of p
51 plot_dc_json[ "max_s_ylim" ] = 1.0
52 plot_dc_json[ "max_surf_ylim" ] = 1
53 plot_dc_json[ "plot_comp" ] = [[1,1,0],[0,0,0],[1,1,1]] # What components do you want to plot?
54
55
56 with open(plot_dc_json_new, 'w') as f2:
57     json5.dump(plot_dc_json, f2, indent=2)
58 f2.close()
59
60 # 8.run MCMTpy in shell
61 os.chdir(notebook_path)
62 os.chdir('../YN.202105212148_Inv/dc_inv')
63 # !MCMTpy plot pyfk -c plot_dc_new.json > plot.log

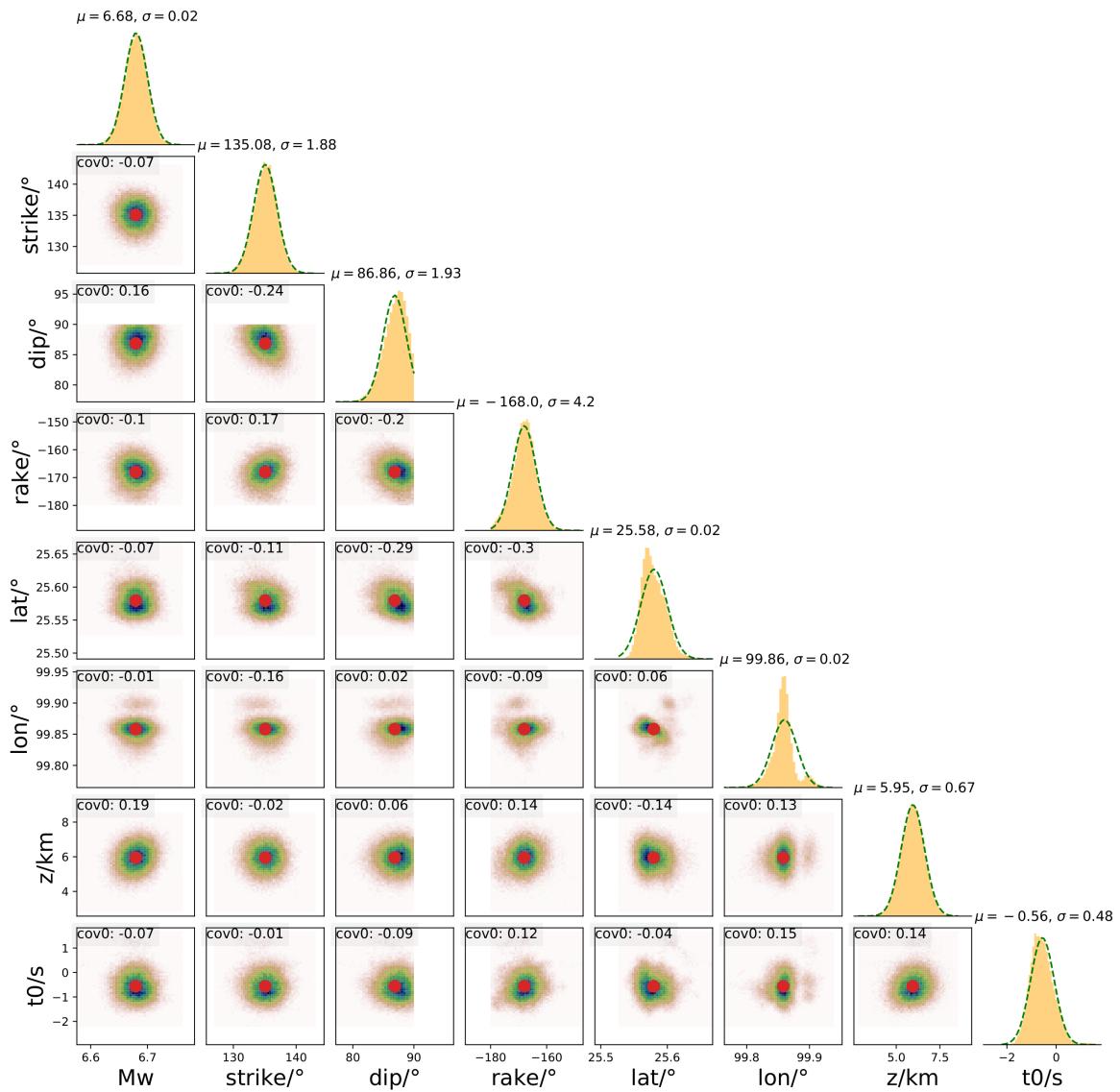
```

- Now you can run MCMTpy in bash:

```
$ MCMTpy plot pyfk -c plot_dc_new.json
```

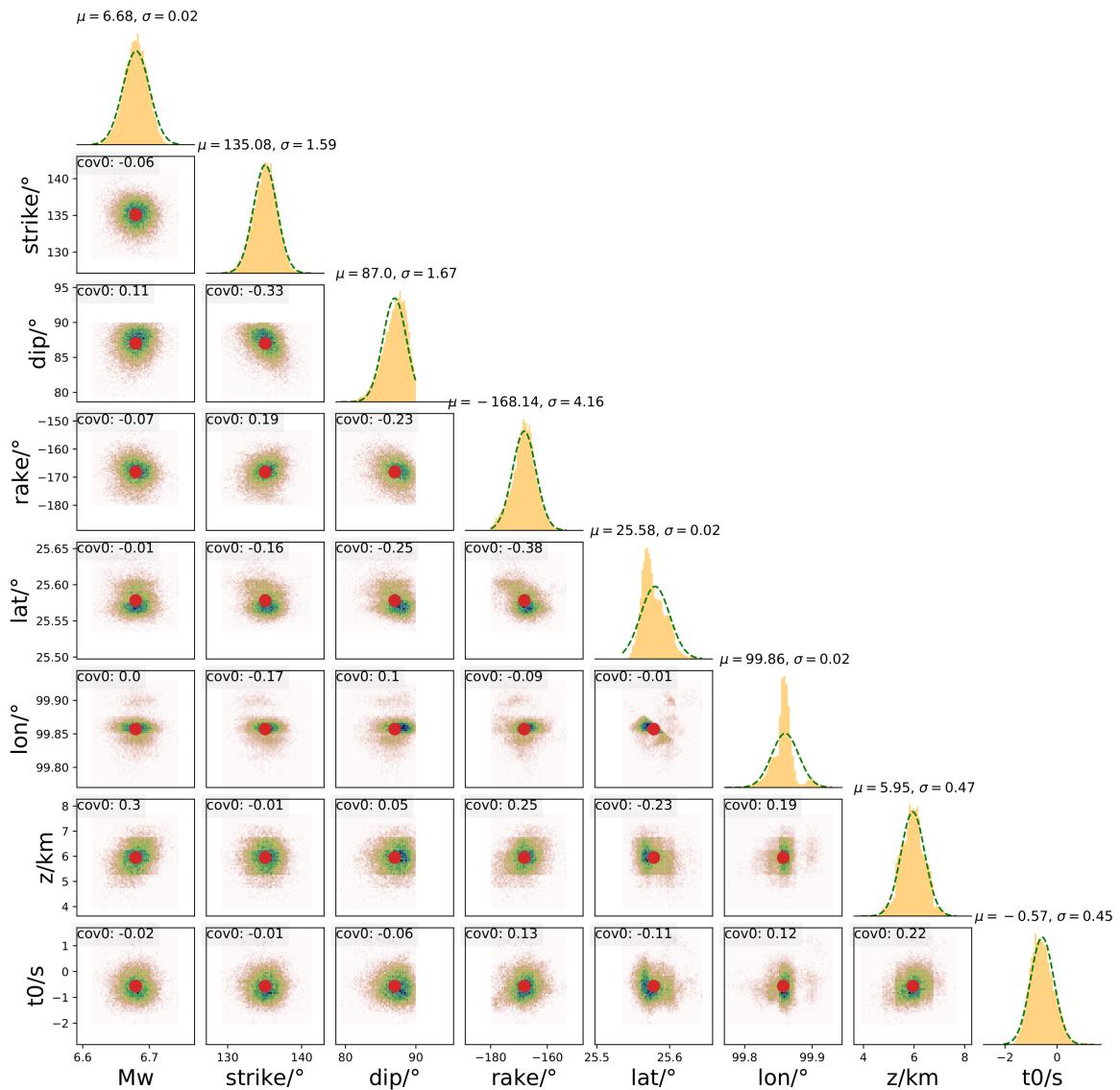
- Show hist in jupyter notebook:

```
fig_hist = os.path.join(plot_dc_json[ "plot_output_path" ], 'hist.jpg')
Image(filename = fig_hist, width=600)
```



- Show hist\_accept in jupyter notebook:

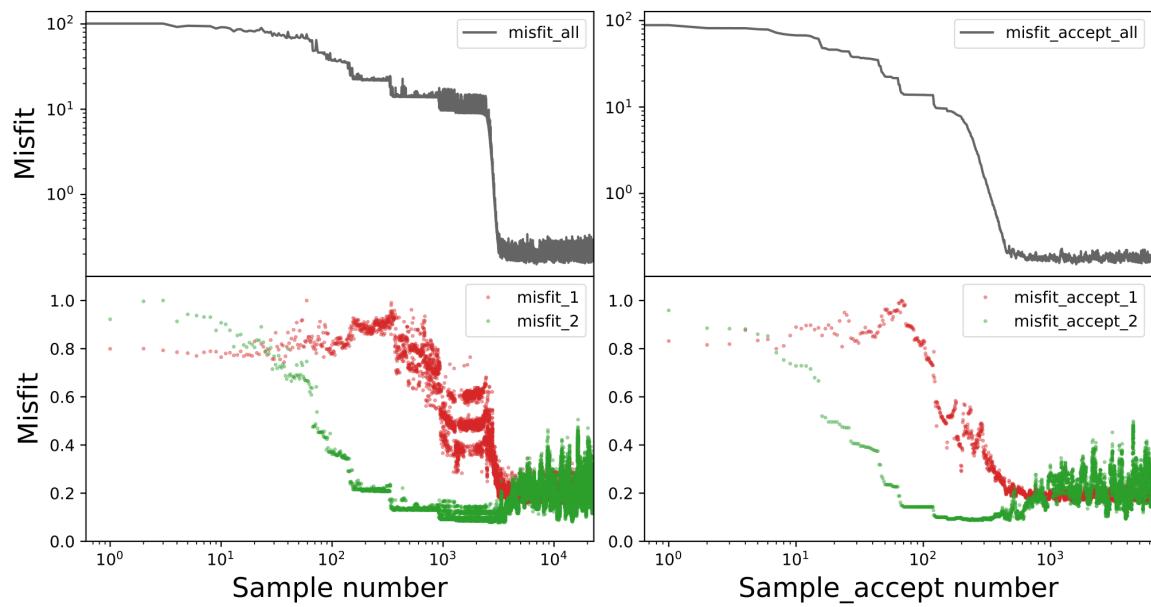
```
fig_hist_accept = os.path.join(plot_dc_json["plot_output_path"], 'hist_accept.jpg')
Image(filename = fig_hist_accept, width=600)
```



- Show misfit in jupyter notebook:

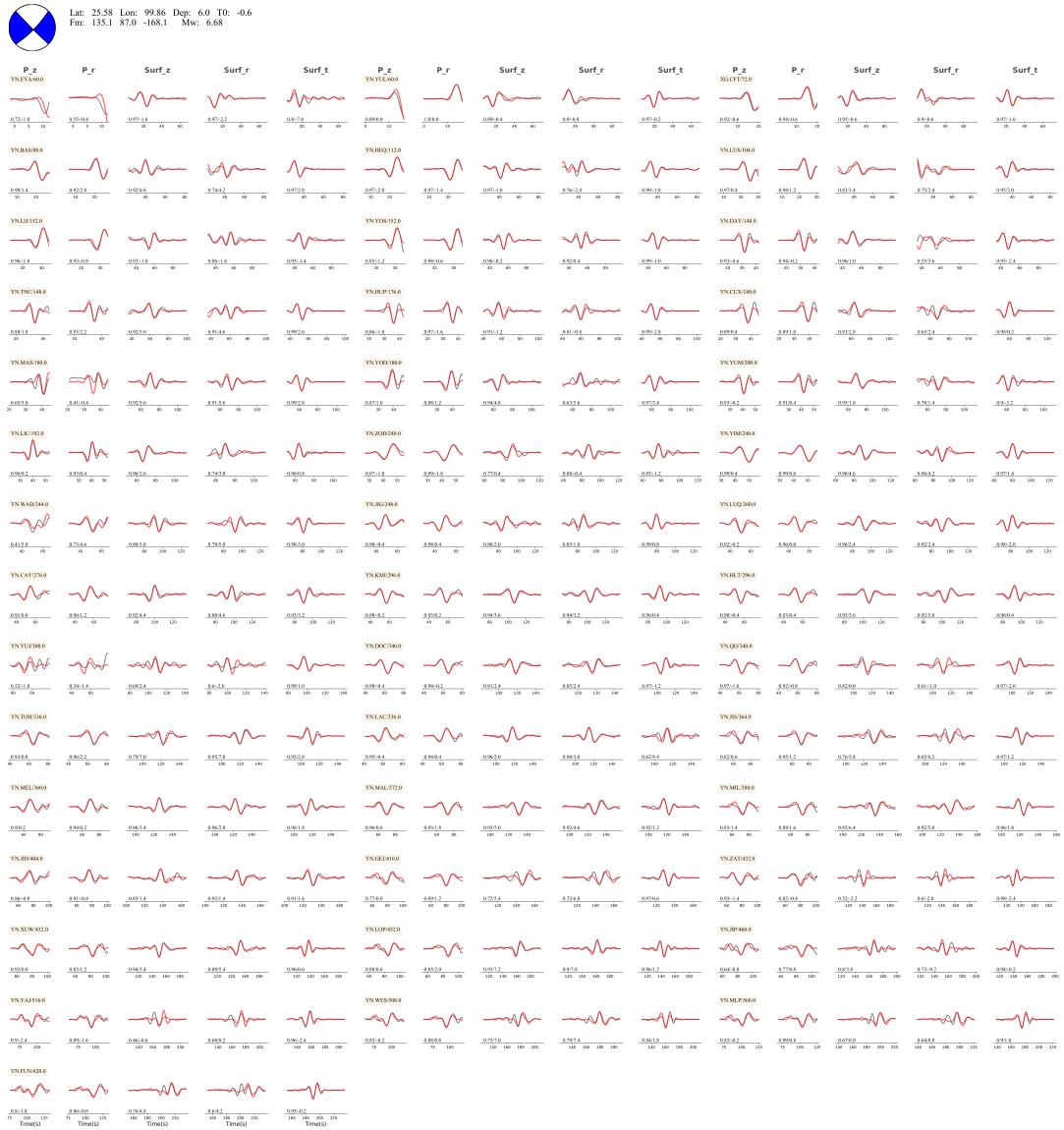
```
fig_misfit = os.path.join(plot_dc_json["plot_output_path"], 'misfit.jpg')
Image(filename = fig_misfit, width=600)
```

### Misfit with iteration



- Show waveform in jupyter notebook:

```
fig_waveform = os.path.join(plot_dc_json["plot_output_path"], 'waveform.jpg')
Image(filename = fig_waveform, width=1000)
```



### Note

Please follow the above parameter instructions and set all parameters correctly before running the program. Otherwise, it is easy to report an error!

### Conversion of Source Parameters

- The MCMTpy provides a series of scripts for source parameter calculation and conversion. The MomentTensor script refers to some theories and codes of [Obspy MoPaD](#) and [Introduction to Seismology](#) (Yongge Wan), mainly including:

<b>The installation</b>	37
<b>str_dip_rake to mt</b>	37
<b>The conversion between str_dip_rake and A/N vector</b>	37
<b>The conversion between A/N vector and P/T/N vector</b>	38
<b>mt to P/T/N vector</b>	38
<b>P/T/N vector to P/T/N vector's stirke and dip</b>	39
<b>Describe_fault_plane with two str_dip_rake</b>	39

- We do not recommend using relative paths because of the possibility of errors. Absolute paths are preferred.

## ***The installation***

- jupyter notebook:

```
# import MomentTensor
from MCMTpy import MomentTensor as MTpy

# get __doc__
MTpy.__doc__.strip().split("\n")
```

## ***str\_dip\_rake to mt***

- jupyter notebook:

```
# function: str_dip_rake2MT
MTpy.str_dip_rake2MT.__doc__.strip().split("\n")

# input
strike = 50
dip = 50
rake = 100

A = MTpy.str_dip_rake2MT(strike,dip,rake)
A.mt
```

## ***The conversion between str\_dip\_rake and A/N vector***

- jupyter notebook:

```
-----#
# function: str_dip_rake2AN
MTpy.str_dip_rake2AN.__doc__.strip().split("\n")

# str_dip_rake to A/N vector
# input
strike = 50
dip = 50
rake = 100

A,N = MTpy.str_dip_rake2AN(strike,dip,rake)
print('A = ', A)
print('N = ', N)

-----#
# function AN2str_dip_rake
MTpy.AN2str_dip_rake.__doc__.strip().split("\n")
```

```
# A/N vector to str_dip_rake
# input
A = np.array([0.37330426, -0.53992106, -0.75440651])
N = np.array([-0.58682409, 0.49240388, -0.64278761])

DD = MTpy.AN2str_dip_rake(A,N)
print('strike = ', DD.strike)
print('dip = ', DD.dip)
print('rake = ', DD.rake)
```

### The conversion between A/N vector and P/T/N vector

er notebook:

```
-----#
unction AN2TPN
y.AN2TPN.__doc__.strip().split("\n")

alculate the T-axis, P-axis and N-axis according to the slip vector (A) and fault plane direction ve
nput
np.array([0.37330426, -0.53992106, -0.75440651])
np.array([-0.58682409, 0.49240388, -0.64278761])

,Null = MTpy.AN2TPN(A,N)
nt('T = ', T)
nt('P = ', P)
nt('Null = ', Null)

-----#
unction TP2AN
y.TP2AN.__doc__.strip().split("\n")

alculate the slip vector (A) and fault plane direction vector (N) according to the T-axis and P-axis
nput
np.array([-0.15098132, -0.03359972, -0.98796544])
np.array([0.67891327, -0.72996397, -0.07892648])

= MTpy.TP2AN(T,P)
nt('A = ', A)
nt('N = ', N)
```

### *mt to P/T/N vector*

- jupyter notebook:

```
# function MT2TPN
MTpy.MT2TPN.__doc__.strip().split("\n")

# input
strike = 50
dip = 50
rake = 100

A = MTpy.str_dip_rake2MT(strike,dip,rake)
T, P, Null = MTpy.MT2TPN(A)
print('T = ', T)
print('P = ', P)
print('Null = ', Null)
```

## P/T/N vector to P/T/N vector's strike and dip

- jupyter notebook:

```
# function vector2str_dip
MTpy.vector2str_dip.__doc__.strip().split("\n")

# input
A = np.array([0.37330426, -0.53992106, -0.75440651])

CC = MTpy.vector2str_dip(A)
print('A.strike = ', CC.strike)
print('A.dip = ', CC.dip)
```

## Describe\_fault\_plane with two str\_dip\_rake

- jupyter notebook:

```
# function describe_fault_plane
MTpy.describe_fault_plane.__doc__.strip().split("\n")

# input
strike = 50
dip = 50
rake = 100

A = MTpy.str_dip_rake2MT(strike,dip,rake)
CC = MTpy.describe_fault_plane(A.mt)
print('FM_1 = ', CC[0,:])
print('FM_2 = ', CC[1,:])
```

## Plot Beachball with Station Projected

- The **example\_path** need to be changed to your path, and run this notebook.

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 import os
4 import glob
5 import obspy
6 import numpy as np
7 import matplotlib.pyplot as plt
8 from obspy import Stream
9 from obspy.taup import TauPyModel
10 from obspy.imaging.mopad_wrapper import beach
11 from MCMTpy import MomentTensor as MTpy
12
13 # Helper function
14 #-----#
15 %% 1.read raw data
16 def read_data(allfiles_path):
17
18     allfiles = sorted( glob.glob(allfiles_path) )
19     data_raw = Stream()
20     for i in range(0,len(allfiles),1):
21         try:
22             tr = obspy.read(allfiles[i])
23             data_raw += tr
24         except Exception:
25             print(allfiles[i],': no such file or obspy read error');continue
```

```

26     data = data_raw.copy()
27
28     return data
29
30 #-----#
31 #%% 2.taup ray trace
32 def get_taup_tp_ts(model,depth,distance,degree=None):
33     if degree==False:
34         distance = distance/111.19
35
36     time_p = model.get_travel_times(source_depth_in_km=depth,
37                                     distance_in_degree=distance,
38                                     phase_list=[ "P", "P"])
39
40     time_s = model.get_travel_times(source_depth_in_km=depth,
41                                     distance_in_degree=distance,
42                                     phase_list=[ "S", "S"])
43
44     ray_p = time_p[0].ray_param
45     tp = time_p[0].time
46     angle_p = time_p[0].incident_angle
47
48     ray_s = time_s[0].ray_param
49     ts = time_s[0].time
50     angle_s = time_s[0].incident_angle
51
52     return ray_p,tp,angle_p,ray_s,ts,angle_s
53
54
55 #%%-----#
56 # 0. set path
57 example_path = '/Users/yf/3.Project/8.MCMTpy/MCMTpy-master/data/example_yunnan'
58
59
60 #-----#
61 # we expect no parameters need to be changed below
62 #-----#
63 FM_path=os.path.join(example_path, "YN.202105212148_Inv/dc_inv/Output_YN.202105212148_dc/")
64 allfiles_path = os.path.join(example_path, 'YN.202105212148_Inv/YN.202105212148_raw/*.SAC')
65
66 ## 1. read FM
67 N=3 # Three parameters are required to describe the focal mechanism
68 FM_all = np.loadtxt(FM_path)
69 FM_raw = FM_all[0:1:N+1] # Define the number of solutions you want to plot
70 strike_np = FM_raw[:,0]
71 dip_np = FM_raw[:,1]
72 rake_np = FM_raw[:,2]
73 FM = np.vstack((strike_np, dip_np, rake_np)).T
74 FM_mean=np.zeros(shape=(N))
75 for i in range(0,N,1):
76     FM_mean[i]=np.mean(FM[0: ,i])
77
78 ## 2.read raw data
79 data = read_data(allfiles_path)
80 data.filter('bandpass', freqmin=0.005, freqmax=0.5, corners=4, zerophase=True)
81
82 ## 3.ray trace with taup
83 model_path = os.path.join(example_path, "v_model/v_model.npz")
84 model = TauPyModel(model=model_path)      # "iasp91" "prem"
85 for i in range(0,len(data),1):

```

```

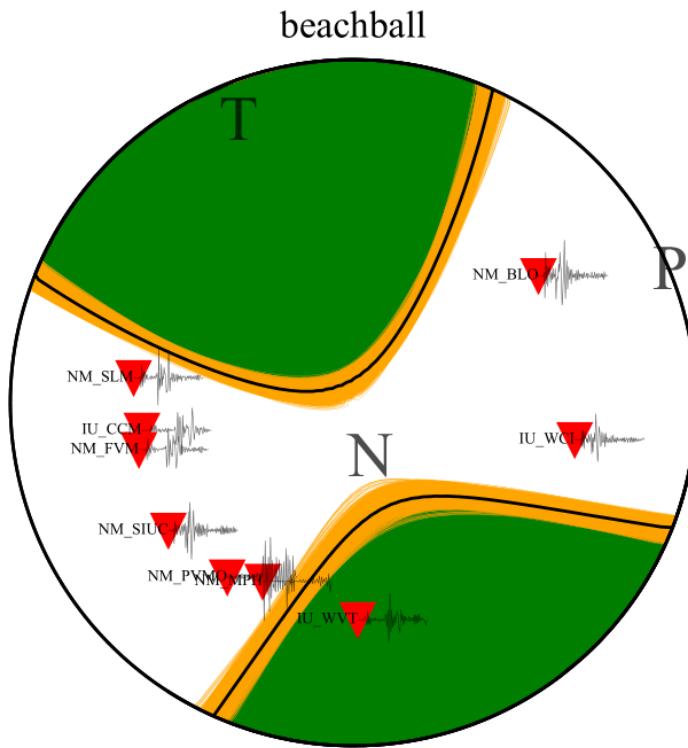
86     depth = data[i].stats.sac['evdp']
87     distance = data[i].stats.sac['dist']
88     ray_p, tp, angle_p, ray_s, ts, angle_s = get_taup_tp_ts(model, depth, distance, degree=False)
89     data[i].stats.sac["user1"] = angle_p
90     data[i].stats.sac["user2"] = angle_s
91
92 ## 4.1 plot FM_mean
93 ax0 = plt.gca()
94 Length_Ball = 100
95 beach1 = beach(FM_mean, xy=(50, 50), linewidth=1, width=Length_Ball-1, alpha=1, \
96                  facecolor='g', bgcolor='w', edgecolor='k', mopad_basis='NED', nofill=False, zorder=1)
97 ax0.add_collection(beach1)
98 ax0.set_aspect("equal")
99
100 ## 4.2 plot FM_all
101 for i in range(0, FM.shape[0], 1):
102     beach1 = beach(FM[i, :], xy=(50, 50), linewidth=1, width=Length_Ball-1, alpha=1, \
103                  facecolor='b', bgcolor='w', edgecolor='orange', mopad_basis='NED', nofill=True, zorder=1)
104     ax0.add_collection(beach1)
105     ax0.set_aspect("equal")
106
107 ## 4.3 plot background line
108 beach1 = beach(FM_mean, xy=(50, 50), linewidth=1, width=Length_Ball-1, alpha=1, \
109                  facecolor='w', bgcolor='w', edgecolor='k', mopad_basis='NED', nofill=True, zorder=1)
110 ax0.add_collection(beach1)
111 ax0.set_aspect("equal")
112
113 ## 5.plot station and waveform
114 method='schmidt' # 'schmidt' 'wulff'
115 for i in range(0, len(data), 3):
116     AZM = data[i].stats.sac['az']
117     TKO = data[i].stats.sac['user1']
118     net_sta_name = data[i].stats.network + '_' + data[i].stats.station
119     X, Y = MTpy.project_beachball(AZM, TKO, R=Length_Ball/2, method=method)
120     tt = np.linspace(X, X+10, num=len(data[i].data))
121     ax0.plot(X, Y, "rv", ms=10, zorder=1)
122     ax0.plot(tt, 5*data[i].data/2000000 + Y, color='black', lw=0.2, alpha=0.6, zorder=1)
123     ax0.text(X, Y, net_sta_name, horizontalalignment='right', verticalalignment='center', \
124               fontsize=5, color='black', bbox=dict(facecolor='r', alpha=0.0), zorder=1)
125
126 ## 6. plot P/T/N axis
127 MT = MTpy.str_dip_rake2MT(strike=FM_mean[0], dip=FM_mean[1], rake=FM_mean[2])
128 T_axis, P_axis, N_axis = MTpy.MT2TPN(MT)
129 T = MTpy.vector2str_dip(T_axis)
130 P = MTpy.vector2str_dip(P_axis)
131 N = MTpy.vector2str_dip(N_axis)
132
133 Tx, Ty = MTpy.project_beachball(AZM=T.strike, TKO=(90-T.dip), R=Length_Ball/2, method=method)
134 ax0.text(Tx, Ty, 'T', horizontalalignment='center', verticalalignment='center', \
135           fontsize=20, color='k', alpha=0.7, zorder=1)
136
137 Px, Py = MTpy.project_beachball(AZM=P.strike, TKO=(90-P.dip), R=Length_Ball/2, method=method)
138 ax0.text(Px, Py, 'P', horizontalalignment='center', verticalalignment='center', \
139           fontsize=20, color='k', alpha=0.7, zorder=1)
140
141 Nx, Ny = MTpy.project_beachball(AZM=N.strike, TKO=(90-N.dip), R=Length_Ball/2, method=method)
142 ax0.text(Nx, Ny, 'N', horizontalalignment='center', verticalalignment='center', \
143           fontsize=20, color='k', alpha=0.7, zorder=1)
144
145

```

```

146 ## 7. save figure
147 ax0.set_xlim(0,100)
148 ax0.set_ylim(0,100)
149 ax0.set_axis_off()
150 figurename=os.path.join('./S2_figure/beachball.pdf')
151 plt.savefig(figurename,dpi=800,format="pdf")

```



## Moment Tensor Decompose

- The **example** **path** need to be changed to your path. and run this notebook.

```

1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import os
5 import glob
6 import obspy
7 import numpy as np
8 import matplotlib.pyplot as plt
9 from obspy import Stream
10 from obspy.taup import TauPyModel
11 from obspy.imaging.mopad_wrapper import beach
12 from MCMTpy import MomentTensor as MTpy
13
14 # Helper function
15 #-----#
16 %% 1.plot decompose mt
17 def plot_decompose(FM,FM_DC,FM_CLVD,FM_iso):
18     MT = MTpy.MTensor(FM)
19     Dec = MTpy.Decompose(MT)

```

```

20     Dec.decomposition_iso_DC_CLVD()
21
22     fig2, ax2 = plt.subplots(1 ,1, dpi=800)
23     Length_Ball = 100
24
25     ##### MT
26     beach1 = beach(FM, xy=(50, 50), linewidth=1, width=Length_Ball-1, alpha=1,\ 
27                     facecolor='g', bgcolor='w', edgecolor='k', mepad_basis='NED', nofill=False)
28     ax2.add_collection(beach1)
29     ax2.set_aspect("equal")
30
31     menthod='schmidt' # 'schmidt' # 'wulff'
32     T_axis, P_axis, N_axis = MTpy.MT2TPN(MTpy.MTensor(FM))
33     T = MTpy.vector2str_dip(T_axis)
34     P = MTpy.vector2str_dip(P_axis)
35     N = MTpy.vector2str_dip(N_axis)
36
37     Tx, Ty = MTpy.project_beachball(AZM=T.strike, TKO=(90-T.dip), R=Length_Ball/2, menthod)
38     ax2.text(Tx,Ty,'T',horizontalalignment='center', verticalalignment='center',\ 
39               fontsize=10, color='k',alpha=0.7,zorder=1)
40
41     Px, Py = MTpy.project_beachball(AZM=P.strike, TKO=(90-P.dip), R=Length_Ball/2, menthod)
42     ax2.text(Px,Py,'P',horizontalalignment='center', verticalalignment='center',\ 
43               fontsize=10, color='k',alpha=0.7,zorder=1)
44
45     Nx, Ny = MTpy.project_beachball(AZM=N.strike, TKO=(90-N.dip), R=Length_Ball/2, menthod)
46     ax2.text(Nx,Ny,'N',horizontalalignment='center', verticalalignment='center',\ 
47               fontsize=10, color='k',alpha=0.7,zorder=1)
48
49     ##### DC
50     beach2 = beach(FM_DC, xy=(200, 50), linewidth=1, width=Length_Ball-1, alpha=1,\ 
51                     facecolor='g', bgcolor='w', edgecolor='k', mepad_basis='NED', nofill=False)
52     ax2.add_collection(beach2)
53     ax2.set_aspect("equal")
54
55     menthod='schmidt' # 'schmidt' # 'wulff'
56     T_axis, P_axis, N_axis = MTpy.MT2TPN(MTpy.MTensor(FM_DC))
57     T = MTpy.vector2str_dip(T_axis)
58     P = MTpy.vector2str_dip(P_axis)
59     N = MTpy.vector2str_dip(N_axis)
60
61     Tx, Ty = MTpy.project_beachball(AZM=T.strike, TKO=(90-T.dip), R=Length_Ball/2, menthod)
62     ax2.text(150*1+Tx,Ty,'T',horizontalalignment='center', verticalalignment='center',\ 
63               fontsize=10, color='k',alpha=0.7,zorder=1)
64
65     Px, Py = MTpy.project_beachball(AZM=P.strike, TKO=(90-P.dip), R=Length_Ball/2, menthod)
66     ax2.text(150*1+Px,Py,'P',horizontalalignment='center', verticalalignment='center',\ 
67               fontsize=10, color='k',alpha=0.7,zorder=1)
68
69     Nx, Ny = MTpy.project_beachball(AZM=N.strike, TKO=(90-N.dip), R=Length_Ball/2, menthod)
70     ax2.text(150*1+Nx,Ny,'N',horizontalalignment='center', verticalalignment='center',\ 
71               fontsize=10, color='k',alpha=0.7,zorder=1)
72
73     ##### CLVD
74     beach3 = beach(FM_CLVD, xy=(350, 50), linewidth=1, width=Length_Ball-1, alpha=1,\ 
75                     facecolor='g', bgcolor='w', edgecolor='k', mepad_basis='NED', nofill=False)
76     ax2.add_collection(beach3)
77     ax2.set_aspect("equal")
78
79     menthod='schmidt' # 'schmidt' # 'wulff'

```

```

80     T_axis, P_axis, N_axis = MTpy.MT2TPN(MTpy.MTensor(FM_CLVD))
81     T = MTpy.vector2str_dip(T_axis)
82     P = MTpy.vector2str_dip(P_axis)
83     N = MTpy.vector2str_dip(N_axis)
84
85     Tx, Ty = MTpy.project_beachball(AZM=T.strike, TKO=(90-T.dip), R=Length_Ball/2, menth
86     ax2.text(150*2+Tx,Ty,'T',horizontalalignment='center', verticalalignment='center',\
87             fontsize=10, color='k',alpha=0.7,zorder=1)
88
89     Px, Py = MTpy.project_beachball(AZM=P.strike, TKO=(90-P.dip), R=Length_Ball/2, menth
90     ax2.text(150*2+Px,Py,'P',horizontalalignment='center', verticalalignment='center',\
91             fontsize=10, color='k',alpha=0.7,zorder=1)
92
93     Nx, Ny = MTpy.project_beachball(AZM=N.strike, TKO=(90-N.dip), R=Length_Ball/2, menth
94     ax2.text(150*2+Nx,Ny,'N',horizontalalignment='center', verticalalignment='center',\
95             fontsize=10, color='k',alpha=0.7,zorder=1)
96
97 ##### iso
98 beach4 = beach(FM_iso, xy=(500, 50), linewidth=1, width=Length_Ball-1, alpha=1,\n
99                 facecolor='g',bgcolor='w', edgecolor='k',mopad_basis='NED',nofill=False)
100    ax2.add_collection(beach4)
101    ax2.set_aspect("equal")
102
103    menthod='schmidt' # 'schmidt' # 'wulff'
104    T_axis, P_axis, N_axis = MTpy.MT2TPN(MTpy.MTensor(FM_iso))
105    T = MTpy.vector2str_dip(T_axis)
106    P = MTpy.vector2str_dip(P_axis)
107    N = MTpy.vector2str_dip(N_axis)
108
109    Tx, Ty = MTpy.project_beachball(AZM=T.strike, TKO=(90-T.dip), R=Length_Ball/2, menth
110    ax2.text(150*3+Tx,Ty,'T',horizontalalignment='center', verticalalignment='center',\
111             fontsize=10, color='k',alpha=0.7,zorder=1)
112
113    Px, Py = MTpy.project_beachball(AZM=P.strike, TKO=(90-P.dip), R=Length_Ball/2, menth
114    ax2.text(150*3+Px,Py,'P',horizontalalignment='center', verticalalignment='center',\
115             fontsize=10, color='k',alpha=0.7,zorder=1)
116
117    Nx, Ny = MTpy.project_beachball(AZM=N.strike, TKO=(90-N.dip), R=Length_Ball/2, menth
118    ax2.text(150*3+Nx,Ny,'N',horizontalalignment='center', verticalalignment='center',\
119             fontsize=10, color='k',alpha=0.7,zorder=1)
120
121 ##### plot '+' and '='
122    ax2.text(125,50,'=',horizontalalignment='center', verticalalignment='center',\
123             fontsize=20, color='k',alpha=0.7,zorder=1)
124    ax2.text(275,50,'+',horizontalalignment='center', verticalalignment='center',\
125             fontsize=20, color='k',alpha=0.7,zorder=1)
126    ax2.text(425,50,'+',horizontalalignment='center', verticalalignment='center',\
127             fontsize=20, color='k',alpha=0.7,zorder=1)
128
129 ##### plot percentage
130    MT_text = 'MT'
131    ax2.text(50,-15,MT_text,horizontalalignment='center', verticalalignment='center',\
132             fontsize=8, color='k',alpha=0.7,zorder=1)
133
134    iso_text = 'ISO: '+str(round(Dec.M_iso_percentage,2))+'%'
135    ax2.text(500,-15,iso_text,horizontalalignment='center', verticalalignment='center',\
136             fontsize=8, color='k',alpha=0.7,zorder=1)
137
138    DC_text = 'DC: '+str(round(Dec.M_DC_percentage,2))+'%'
139    ax2.text(200,-15,DC_text,horizontalalignment='center', verticalalignment='center',\

```

```

140         fontsize=8, color='k',alpha=0.7,zorder=1)
141
142     CLVD_text = 'CLVD: '+str(round(Dec.M_CLVD_percentage,2))+'%'
143     ax2.text(350,-15,CLVD_text,horizontalalignment='center', verticalalignment='center',
144               fontsize=8, color='k',alpha=0.7,zorder=1)
145
146 ##### set figure
147 plt.title("Decompose")
148 ax2.set_xlim(0,550)
149 ax2.set_ylim(-30,100)
150 ax2.set_axis_off()
151
152 return fig2
153
154 #####-----#
155 # 1.set FM
156 # [Mxx_np, Myy_np, Mzz_np, Mxy_np, Mxz_np, Myz_np] or [strike,dip,rake]
157 FM=[150,50,100]
158
159
160 #####-----#
161 # we expect no parameters need to be changed below
162 #####-----#
163 MT = MTpy.MTensor(FM)
164 Dec = MTpy.Decompose(MT)
165 Dec.decomposition_iso_DC_CLVD()    # Dec.help()
166 print(Dec.help())
167 print('\n*****\n')
168 Dec.print_self()
169
170 Mxx = Dec.M_iso[0,0]
171 Mxy = Dec.M_iso[0,1]
172 Mxz = Dec.M_iso[0,2]
173 Myy = Dec.M_iso[1,1]
174 Myz = Dec.M_iso[1,2]
175 Mzz = Dec.M_iso[2,2]
176 FM_iso = np.array((Mxx, Myy, Mzz, Mxy, Mxz, Myz))
177
178 Mxx = Dec.M_DC[0,0]
179 Mxy = Dec.M_DC[0,1]
180 Mxz = Dec.M_DC[0,2]
181 Myy = Dec.M_DC[1,1]
182 Myz = Dec.M_DC[1,2]
183 Mzz = Dec.M_DC[2,2]
184 FM_DC = np.array((Mxx, Myy, Mzz, Mxy, Mxz, Myz))
185
186 Mxx = Dec.M_CLVD[0,0]
187 Mxy = Dec.M_CLVD[0,1]
188 Mxz = Dec.M_CLVD[0,2]
189 Myy = Dec.M_CLVD[1,1]
190 Myz = Dec.M_CLVD[1,2]
191 Mzz = Dec.M_CLVD[2,2]
192 FM_CLVD = np.array((Mxx, Myy, Mzz, Mxy, Mxz, Myz))
193
194 fig = plot_decompose(FM, FM_DC, FM_CLVD, FM_iso)
195 figurename=os.path.join('./S2_figure/Decompose.pdf')
196 plt.savefig(figurename,dpi=800, format="pdf")

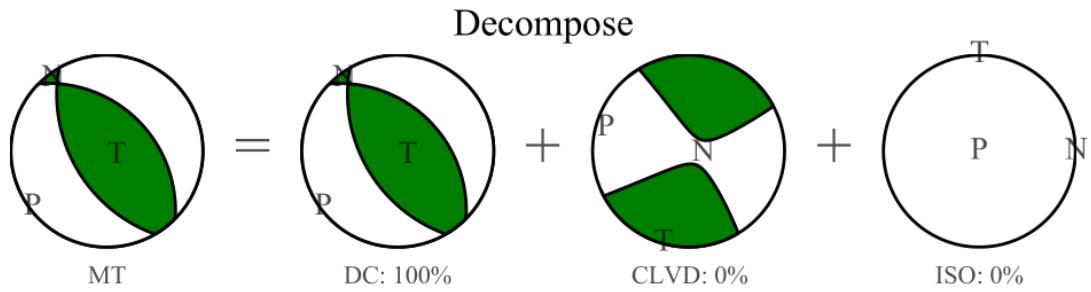
```

- **Decompose result.**

```

1 Including function:
2 self.M
3 self.M_iso
4 self.M_devi
5 self.M_DC
6 self.M_CLVD
7 self.M0
8 self.Mw
9 self.M_iso_percentage
10 self.M_DC_percentage
11 self.M_CLVD_percentage
12 self.eigen_val
13 self.eigen_vec
14 self.F
15 None
16
17
18 ****
19
20
21 self.M: [[-0.3576622 -0.48646688 -0.01115976]
22 [-0.48646688 -0.61218411  0.20390851]
23 [-0.01115976  0.20390851  0.96984631]]
24
25 self.M_iso: [[ -3.70074342e-17    0.00000000e+00    0.00000000e+00]
26 [  0.00000000e+00   -3.70074342e-17    0.00000000e+00]
27 [  0.00000000e+00    0.00000000e+00   -3.70074342e-17]]
28
29 self.M_devi: [[-0.3576622 -0.48646688 -0.01115976]
30 [-0.48646688 -0.61218411  0.20390851]
31 [-0.01115976  0.20390851  0.96984631]]
32
33 self.M_DC: [[-0.3576622 -0.48646688 -0.01115976]
34 [-0.48646688 -0.61218411  0.20390851]
35 [-0.01115976  0.20390851  0.96984631]]
36
37 self.M_CLVD: [[  3.33066907e-16   1.66533454e-16  -2.77555756e-17]
38 [  1.66533454e-16  -2.22044605e-16   5.55111512e-17]
39 [ -2.77555756e-17   5.55111512e-17   0.00000000e+00]]
40
41 self.M0: 1.0
42
43 self.Mw: -6.0666666667
44
45 self.M_iso_percentage: 0
46
47 self.M_DC_percentage: 100
48
49 self.M_CLVD_percentage: 0
50
51 self.eigen_val: [ -1.00000000e+00  -5.46437895e-17   1.00000000e+00]
52
53 self.eigen_vec: [[-0.60098212 -0.79705908 -0.0593069 ]
54 [-0.79535596  0.58906868  0.14285304]
55 [ 0.07892648 -0.13302222  0.98796543]]
56
57 self.F: -1.76363553391e-17

```



## Huston Plot

- The **example\_path** need to be changed to your path, and run this notebook.

```

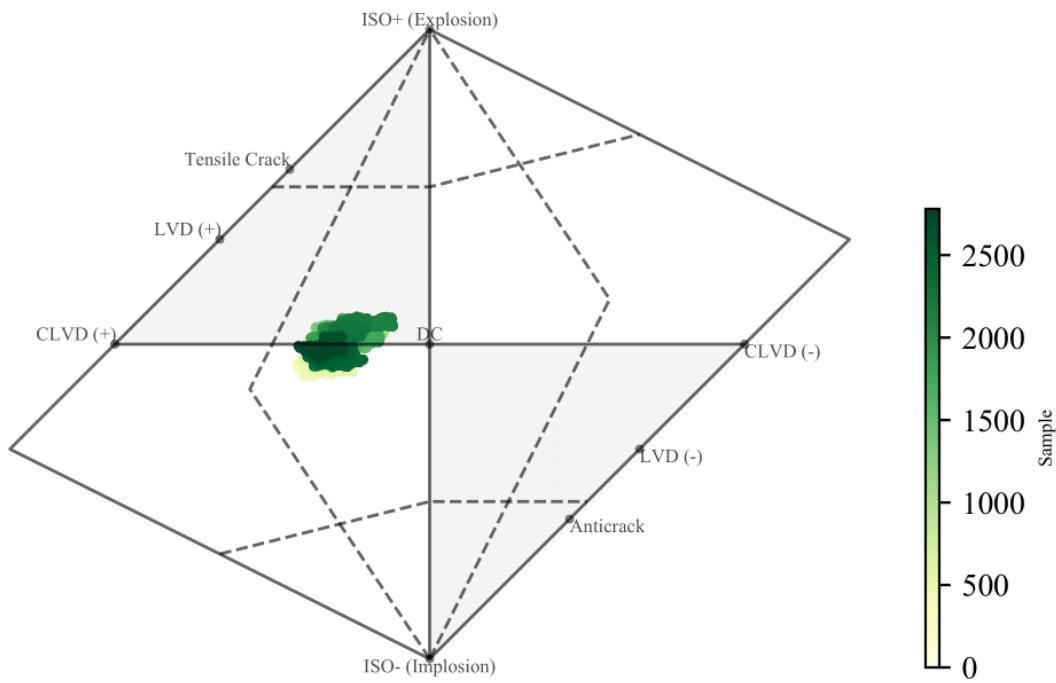
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 import os
4 import numpy as np
5 import matplotlib.pyplot as plt
6 from matplotlib import cm
7 from MCMTpy import MomentTensor as MTpy
8
9 # Helper function
10 #-----#
11 %% 4. plot_Hudson_points
12 def plot_Hudson_points(FM):
13     fig1, ax1 = plt.subplots(1,1, dpi=800)
14     plt.rc('font',family='Times New Roman')
15     MTpy.Hudson_plot(ax=ax1)
16     for i in range(0,len(FM)):
17         MT = MTpy.MTensor(FM[i,:])
18         M=MT.mt
19
20         k,T = MTpy.M2kT_space(M)
21         U,V = MTpy.kT2UV_space(k,T)
22
23         map_vir = cm.get_cmap(name='YlGn')
24         colors = map_vir(i/len(FM))
25
26         ax1.scatter(U,V, color=colors,marker='o', s=5)

```

```

28
29     position=fig1.add_axes([0.85, 0.15, 0.01, 0.5])
30     font_colorbar = {'family' : 'Times New Roman',
31                     'color'   : 'black',
32                     'weight'  : 'normal',
33                     'size'    : 6,
34                     }
35     sm = cm.ScalarMappable(cmap=map_vir)
36     sm.set_array(np.arange(0,len(FM)+1))
37     cb=plt.colorbar(sm,cax=position,orientation='vertical')
38     cb.set_label('Sample',fontdict=font_colorbar)
39
40     ax1.set_xlim(-4/3-0.1, 4/3+0.3)
41     ax1.set_ylim(-1-0.1, 1+0.1)
42
43     return fig1
44
45
46 #####-----#
47 ## 0. set path
48 example_path = '/Users/yf/3.Project/8.MCMTpy/MCMTpy-master/data/example_yunnan'
49
50
51 #####-----#
52 # we expect no parameters need to be changed below
53 #####-----#
54 FM_path=os.path.join(example_path,"YN.202105212148_Inv/mt_inv/Output_YN.202105212148_mt/r"
55 allfiles_path = os.path.join(example_path,'YN.202105212148_Inv/YN.202105212148_raw/*.SAC')
56 ## 1. read FM
57 N=6
58 FM_all = np.loadtxt(FM_path)
59 FM_raw = FM_all[0:,1:N+1]
60 Mxx_np = FM_raw[:,0]
61 Mxy_np = FM_raw[:,1]
62 Mxz_np = FM_raw[:,2]
63 Myy_np = FM_raw[:,3]
64 Myz_np = FM_raw[:,4]
65 Mzz_np = FM_raw[:,5]
66 FM = np.vstack((Mxx_np, Myy_np, Mzz_np, Mxy_np, Mxz_np, Myz_np)).T
67
68 ## 2. plot Hudson
69 fig = plot_Hudson_points(FM)
70 figurename=os.path.join('./S2_figure/Hudson.pdf')
71 plt.savefig(figurename,dpi=800, format="pdf")

```



## References

### **ASDF**

The `pyasdf` Github repository <https://github.com/SeismicData/pyasdf>.

- Krischer, L., Smith, J., Lei, W., Lefebvre, M., Ruan, Y., de Andrade, E.S., Podhorszki, N., Bozda, E. and Tromp, J., 2016. An adaptable seismic data format. Geophysical Supplements to the Monthly Notices of the Royal Astronomical Society, 207(2), 1003-1011.

### **pyfk**

The `pyfk` Github repository <https://github.com/ziyixi/pyfk>.

### **NoisePy**

The `NoisePy` Github repository <https://noise-python.readthedocs.io/en/latest/examples.html>.

- Jiang, C., Yuan, C., and Denolle, M. NoisePy: a new high-performance python tool for seismic ambient noise seismology.

### **MoPaD**

The `MoPaD` website <http://www.larskrieger.de/mopad/>

- Lars Krieger and Sebastian Heimann. MoPad —Moment Tensor Plotting and Decomposition: A Tool for Graphical and Numerical Analysis of Seismic Moment Tensors.

## BEAT

The BEAT website <https://hvasbath.github.io/beat/index.html>

- Vasyura-Bathke, H., J. Dettmer, A. Steinberg, S. Heimann, M. P. Isken, O. Zielke, P. M. Mai, H. Sudhaus, and S. Jónsson (2020). The Bayesian Earthquake Analysis Tool, *Seismol. Res. Lett.* 91, 1003–1018, doi: 10.1785/0220190075.

## Other references adding...

Working on that...

## Indices and tables

- [genindex](#)
- [search](#)