
mttime

Release 1.1.dev0

Andrea Chiang

Aug 17, 2021

CONTENTS

1	Getting started	3
1.1	Requirements	3
1.2	Installation	3
1.3	Usage	4
2	User Guide	5
2.1	Basics	5
2.2	Dataset	5
2.3	Output Files	9
2.4	Figure Options	11
3	API Reference	15
3.1	Core Modules	15
3.2	Plotting Modules	26
3.3	Scripts	29
4	Indices and tables	31
	Python Module Index	33
	Index	35

mttime is a python package developed for time domain inversion of complete seismic waveform data to obtain the seismic moment tensor. It supports deviatoric and full moment tensor inversions, and 1-D and 3-D basis Green's functions.

GETTING STARTED

1.1 Requirements

The package was developed on python 3.7 and 3.8, and is running and tested on Mac OSX.

- ObsPy 1.0.* and its dependencies
- pandas 1.0.*

1.2 Installation

- Create a Python environment
- Install ObsPy and pandas
- Make sure you have [cloned the repository](#)
- Install mtime

I recommend installing Python via [Miniconda](#) or [Anaconda](#). Choose Miniconda for a lower footprint. Then follow the instructions on their sites to install [ObsPy](#) and [pandas](#) for your given platform.

Download mtime and install it from source. If you installed Python via conda make sure you activate the environment where ObsPy and pandas are installed.

```
# Activate environment
conda activate your_environment

# Build and install mtime
git clone https://github.com/LLNL/mtime
cd mtime
python setup.py install
```

Finally, if you want to run the tutorials you will need to install [Jupyter Notebook](#).

1.3 Usage

You can execute the package from the command line or run it interactively in the Python console. In either case you will need to provide an input file that contains information about the file structure and parameters to set up the inverse routine. If no file is given the code will look for a file named **mtinv.in** in the current working directory.

Executing the package from command line will launch the inversion, save and plot the result to file:

```
mttime-run mtinv.in
```

The equivalent in the Python console:

```
>>> import mttime
>>> config = mttime.Configure(path_to_file="mtinv.in")
>>> mt = mttime.Inversion(config=config)
>>> mt.invert()
>>> mt.write()
```

Refer to the *User Guide* for more details.

USER GUIDE

The user guide covers the inverse routine and file structure. Users brand new to moment tensor inversion should check out the Jupyter Notebook [tutorials](#) for an complete example that includes data processing and synthetics calculation.

Further information on any specific method can be obtained in the [API Reference](#).

2.1 Basics

The inverse routine is divided into three main parts:

1. Parsing an input file to get a set of parameters required to run the inversion.
2. Perform the inversion based on the parameters provided.
3. Write the results to file.

```
>>> # import the package
>>> import mtttime

>>> # 1. read input file and set up the inversion
>>> config = mtttime.Configure(path_to_file="mtinv.in")
>>> mt = mtttime.Inversion(config=config)

>>> # 2. Run inversion and plot the result (if plotting function is turned on)
>>> mt.invert()

>>> # 3. Save result to file
>>> mt.write()
```

We will go over the details in the following sections.

2.2 Dataset

In this section we will introduce the file structure and input parameters using an example. The earthquake in this example occurred near Byron, California on July 16, 2019. The raw data and instrument response were obtained from the [IRIS DMC](#) using ObsPy's [mass_downloader](#) functionality.

2.2.1 Directory and file structures

In this example all of the files (input, data and synthetics) are assumed to be located under a single root directory called **project**:

```
project
|   mtinv.in
|
|___40191336
|   |   BK.QRDG.00.z.dat
|   |   BK.RUSS.00.z.dat
|   |   ...
|
|___40191336/gil7
|   |   BK.QRDG.00.12.0000.ZDD
|   |   BK.QRDG.00.12.0000.ZDS
|   |   ...
```

Under project there is the input parameter file **mtinv.in** which contains headers and a station table:

```
origin      2019-07-16T20:11:01.470000Z
longitude   -121.7568
latitude     37.8187
depth       10,12,20
path_to_data 40191336
path_to_green 40191336/gil7
green        herrmann
components   ZRT
degree       6
weight       distance
plot         1
correlate    0
```

station	distance	azimuth	ts	npts	dt	used	longitude	latitude
BK.QRDG.00	80.99	335.29	32	150	1.00	1	-122.14	38.48
BK.RUSS.00	81.16	353.18	32	150	1.00	1	-121.87	38.54
BK.CVS.00	84.88	313.73	31	150	1.00	1	-122.46	38.35
BK.OAKV.00	88.89	320.02	31	150	1.00	1	-122.41	38.43
BK.FARB.00	110.46	263.41	31	150	1.00	1	-123.00	37.70
BK.SAO.00	120.23	166.71	31	150	1.00	1	-121.45	36.76
BK.CMB.00	122.83	78.33	31	150	1.00	1	-120.39	38.03
BK.MNRC.00	132.06	333.21	32	150	1.00	1	-122.44	38.88

Configure object will parse the input text file.

```
>>> config = mttime.Configure(path_to_file="mtinv.in")
```

The headers have two columns: a parameter name and its corresponding value. If the values are left blank, the default values will be used instead. A description of the parameters are shown here:

parameters	description
path_to_file	path to input file containing headers and station information. Directory will become the project root directory. Default is <code>"./mtinv.in"</code> .
datetime	event origin time, optional.
longitude	event longitude, optional.
latitude	event latitude, optional.
depth	source depths to invert.
path_to_data	path to data files, relative to root directory. Defaults is <code>"./"</code> .
path_to_green	path to Green's function files, relative to root directory. Defaults is <code>"./"</code> .
green	Green's function format, options are <code>"herrmann"</code> or <code>"tensor"</code> . Defaults to <code>"herrmann"</code> .
components	waveform components, options are <code>"Z"</code> for vertical component, or <code>"ZRT"</code> for three-component data in vertical, radial and transverse components. Defaults to <code>"ZRT"</code> .
degree	degrees of freedom allowed in the inversion, options are 5 for deviatoric or 6 for full. Defaults to 5.
weight	data weights, options are <code>"none"</code> , <code>"distance"</code> or <code>"variance"</code> for no weights, inverse distance, or inverse variance, respectively. Defaults to <code>"none"</code> .
plot	If <code>True</code> will plot the solution and waveform fits. Default is <code>False</code> .
correlate	Flag to cross-correlate data and Green's functions for best time shift in time points. Default is <code>False</code> .

Station table

Lines 13 and onward in **mtinv.in** contain the station information, line 13 is the station header and should not be modified. A description of the headers is shown here:

headers	description
station	file names of data and synthetics
distance	source-receiver distance
azimuth	source-receiver azimuth
ts	shift data by the number of time points specified
npts	number of samples to invert
dt	sampling interval in seconds
used	components to invert, set 1 to invert and 0 for prediction only. For three component data you can set flags for individual components, ordered by ZRT. e.g. 110 will invert ZR components only
longitude	station longitude
latitude	station latitude

2.2.2 Waveform Data

mttime expects both observed and synthetic Green's functions to be fully processed. This means they are corrected for instrument response, filtered, and decimated, and saved as SAC binary files.

The data file names have the following format:

`[station].[component].dat`

- station: from the station column
- component: Z, R, or T (from the components parameter)

With the example above the data file names for station BK.CMB.00 are:

- BK.CMB.00.Z.dat
- BK.CMB.00.R.dat
- BK.CMB.00.T.dat

2.2.3 Synthetic Seismograms

The basis Green's functions can be combined to create three component time histories for an arbitrarily oriented point source. As mentioned previously two types of synthetic basis Green's functions are accepted: `herrmann` and `tensor`.

The tensor format is pretty straight forward, it consists of the six elementary tensor elements in cartesian space. north, east and down directions (x, y and z) in Aki and Richards (2002). The herrmann format is based on the formulation of Herrmann and Wang (1985), which consists of ten fundamental source types.

The synthetic file names have the following format:

`[station].[depth].[green_function_name]`

- station: from the station column
- depth: source depth with four significant digits
- component: Z, R, or T (from the components parameter)
- green_function_name: this depends on format of the Green's functions
 - `herrmann`: TSS, TDS, RSS, RDS, RDD, ZSS, ZDS, ZDD, REX, and ZEX (total of 10)
 - `tensor`: ZXX, ZYY, ZZZ, ZXY, ZXZ, ZYZ, RXX, etc. (total of 18)

With the example above the synthetic file names for station BK.CMB.00 are:

- BK.CMB.00.12.0000.TSS
- BK.CMB.00.12.0000.TDS
- and so on.

If we change the format to tensor (such that `green="tensor"`), the file names become:

- BK.CMB.00.12.0000.TXX
- BK.CMB.00.12.0000.TXY
- and so on.

2.3 Output Files

Running the inversion with the example above will generate the following outputs.

```
>>> mt = mttime.Inversion(config=config)
>>> mt.invert()
>>> mt.write()
```

- Text files:
 - d10.0000.mtinvs.out - moment tensor solution at 10 km depth
 - d12.0000.mtinvs.out
 - d20.0000.mtinvs.out

```
Full Moment Tensor Inversion
Depth = 10.0000 (km)
Mo = 3.833e+22 (dyne-cm)
Mw = 4.36
Percent DC    = 90
Percent CLVD  = 8
Percent ISO   = 3
Fault Plane 1: Strike=233 Dip=66 Rake=-6
Fault Plane 2: Strike=326 Dip=84 Rake=-156
Percent Variance Reduction = 74.47

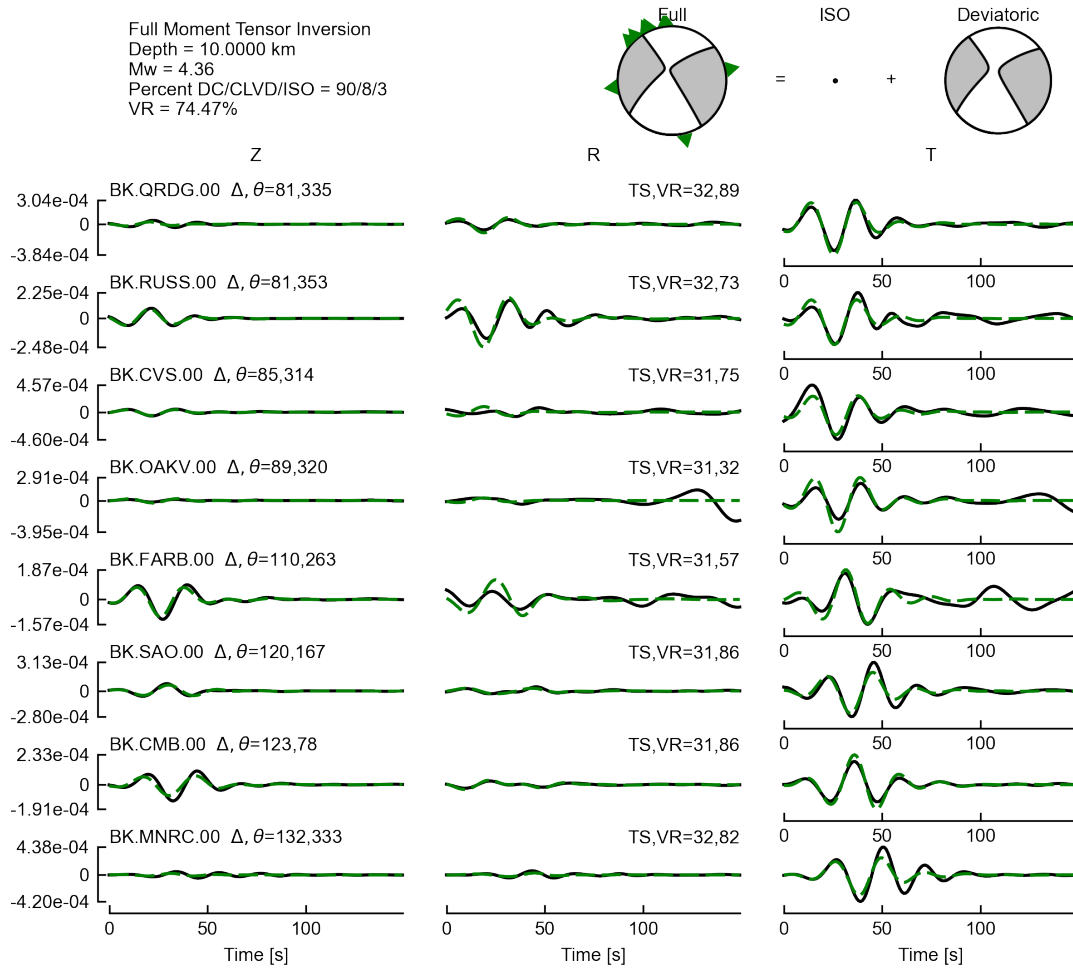
Moment Tensor Elements: Aki and Richards Cartesian Coordinates
Mxx      Myy      Mzz      Mxy      Mxz      Myz
-2.836e+22 3.458e+22 -3.037e+21 -1.067e+22 1.033e+22 1.066e+22

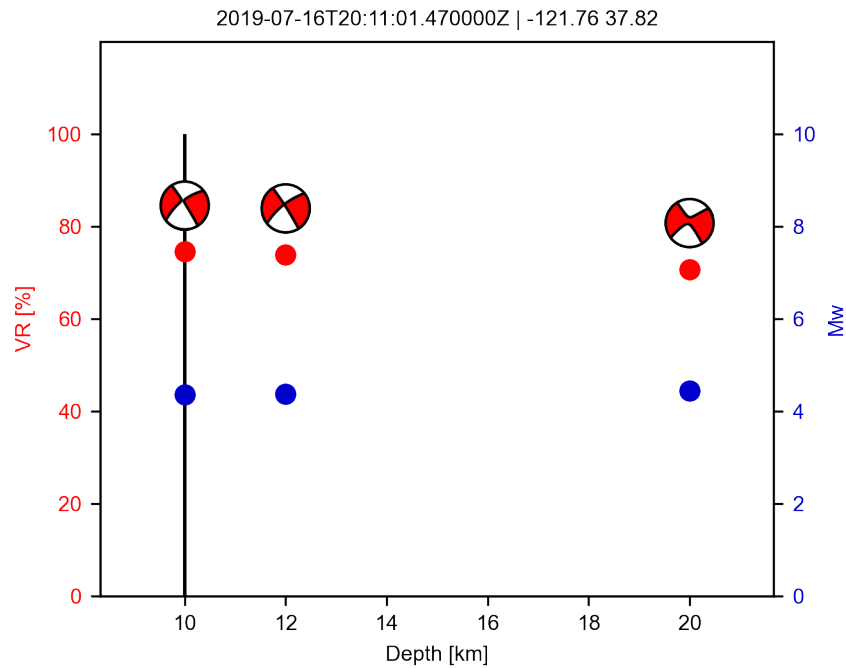
Harvard/CMT convention
Mrr      Mtt      Mpp      Mrt      Mrp      Mtp
-3.037e+21 -2.836e+22 3.458e+22 1.033e+22 -1.066e+22 1.067e+22

Eigenvalues: 3.833e+22 -3.790e+20 -3.477e+22
Lune Coordinates: -1.95 2.03
Station Information
  station distance azimuth ts npts dt Z R T weights VR longitude latitude
BK.QRDG.00 80.99 335.29 32 150 1.00 1 1 1 1.00 89.26 -122.140 38.480
BK.RUSS.00 81.16 353.18 32 150 1.00 1 1 1 1.00 73.30 -121.870 38.540
BK.CVS.00 84.88 313.73 31 150 1.00 1 1 1 1.05 74.81 -122.460 38.350
BK.OAKV.00 88.89 320.02 31 150 1.00 1 1 1 1.10 32.40 -122.410 38.430
BK.FARB.00 110.46 263.41 31 150 1.00 1 1 1 1.36 56.80 -123.000 37.700
BK.SAO.00 120.23 166.71 31 150 1.00 1 1 1 1.48 86.32 -121.450 36.760
BK.CMB.00 122.83 78.33 31 150 1.00 1 1 1 1.52 85.93 -120.390 38.030
BK.MNRC.00 132.06 333.21 32 150 1.00 1 1 1 1.63 82.31 -122.440 38.880
```

- Figures:
 - bbwaves.d10.0000.00.eps - focal mechanism and waveform fits at 10 km depth
 - bbwaves.d12.0000.00.eps
 - bbwaves.d20.0000.00.eps
 - depth.bbmw.eps

Since we performed the inversion at multiple source depths, in addition to the standard focal mechanism and waveform figures, the plotting function will also generate a figure that shows the solution as function of source depth.





2.4 Figure Options

There are more display options available in `mttime.core.inversion.Inversion.plot`.

- `view="waveform"` - MT solution and waveform fits.
- `view="depth"` - MT solution as a function of source depth.
- `view="map"` - map view.
- `view="lune"` - full MT source-type on the lune.

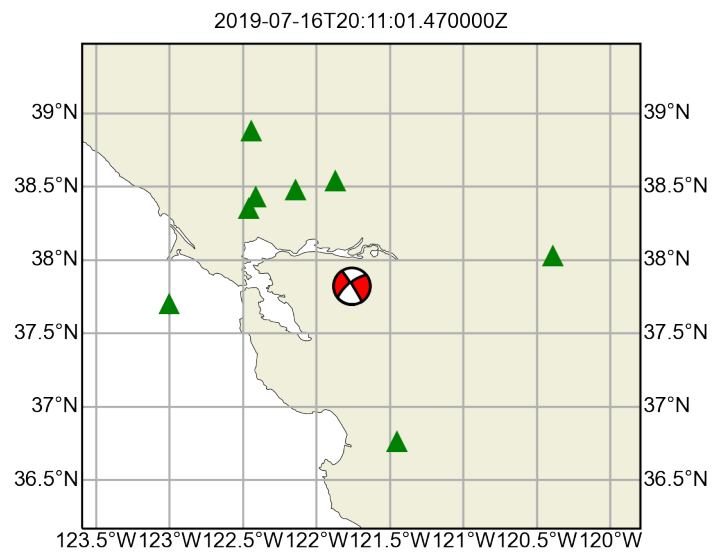
In the example above we set `mt.config.plot=True`, which is equivalent to:

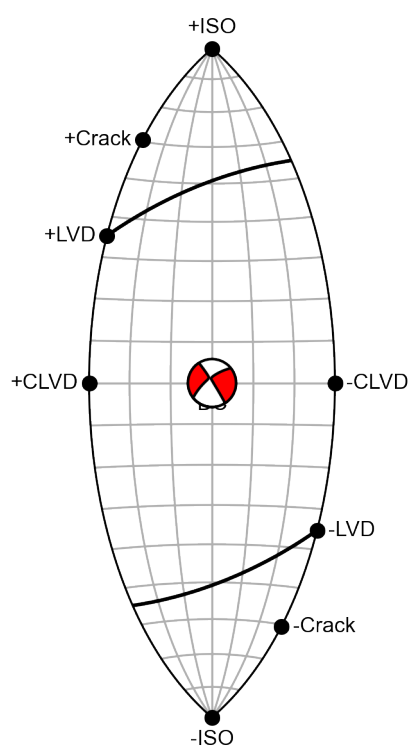
```
>>> mt.plot(view="waveform")
>>> mt.plot(view="depth") # if depth is not fixed
```

To plot the solution on a map or lune:

```
>>> mt.plot(view="map", show=True)
```

```
>>> mt.plot(view="lune", show=True)
```





API REFERENCE

This page gives an overview of all public `mttime` objects, methods and functions.

3.1 Core Modules

<code>mttime.core.configure</code>	Set global <code>mttime</code> configuration
<code>mttime.core.inversion</code>	Time domain moment tensor inverse routines
<code>mttime.core.tensor</code>	Routines for handling inversion results
<code>mttime.core.utils</code>	Utility functionality for <code>mttime</code>

3.1.1 `mttime.core.configure`

Set global `mttime` configuration

Example

```
>>> import mttime
>>> # read input file mtinv.in
>>> config = mttime.Configure(path_to_file="mtinv.in")
```

Classes

<code>Configure</code>	Configure object for <code>Inversion</code>
------------------------	---

`mttime.core.configure.Configure`

class `Configure` (`path_to_file=None`, `df=None`, `**kwargs`)

Bases: `object`

Configure object for `Inversion`

Sets up the moment tensor inverse routine. `**kwargs` can be provided either in `path_to_file` or during class instantiation. `df` and `depth` are required if not reading from a file. `**kwargs` will override the values in `path_to_file` and any missing keyword arguments will be set to their default values.

Parameters

- **path_to_file** (*str*) – path to input file containing headers and station information. Input file should be located in the project root directory.
- **df** (*DataFrame*) – station table, required if `path_to_file=None`.
- **datetime** (*str*, *optional*) – event origin time.
- **longitude** (*float*, *optional*) – event longitude.
- **latitude** (*float*, *optional*) – event latitude.
- **depth** (*float*, *int*, *list*) – source depths to invert, required if `path_to_file=None`
- **path_to_data** (*str*) – path to data files, relative to root directory. Defaults is `"."`.
- **path_to_green** (*str*) – path to Green's function files, relative to root directory. Default is `"."`.
- **green** (*str*) – Green's function format, options are `"herrmann"` or `"tensor"`. Default is `"herrmann"`.
- **components** (*str*) – waveform components, options are `"Z"` for vertical component, or `"ZRT"` for three-component data in vertical, radial and transverse components, and `"ZNE"` for vertical, north and east. Default is `"ZRT"`.
- **degree** (*int*) – degrees of freedom allowed in the inversion, options are 5 for deviatoric or 6 for full. Default is 5.
- **weight** (*str*) – data weights, options are `"none"`, `"distance"` or `"variance"` for no weights, inverse distance, or inverse variance, respectively. Default is `"none"`.
- **plot** (*int*, *bool*) – If True will plot the solution and waveform fits. Default is False.
- **correlate** (*int*, *bool*) – Flag to cross-correlate data and Green's functions for best time shift (in time points). Default is False.

Methods

<code>write</code>	Function to write inversion configuration to a file
--------------------	---

`mttime.core.configure.Configure.write`

`Configure.write` (*fileout*='config.out')

Function to write inversion configuration to a file

Write inverse routine parameters and station table to a file.

Parameters `fileout` (*str*) – output file name. Default is `"config.out"`.

Example

```
>>> config = Configure()
>>> config.write(fileout="configure.out")
```

3.1.2 mttime.core.inversion

Time domain moment tensor inverse routines

Example

```
>>> config = mttime.Configure(path_to_file="examples/synthetic/mtinv.in")
>>> tdmt = mttime.Inversion(config=config)
>>> print(tdmt)
  event: {'datetime': '2019-07-16T20:10:31.473', 'longitude': -121.757, 'latitude'
↳ ': 37.8187}
  depth: [10.0, 20.0]
  green: herrmann
components: ['Z', 'R', 'T']
  degree: 5
  weight: distance
  plot: False
  correlate: False
| STATION TABLE |
  station distance azimuth ts npts dt Z R T longitude latitude filter
↳ nc np lcrn hcrn model
BK.FARB.00 110.00 263.00 30 100 1.00 0 0 0 -123.00 37.70 bp
↳ 2 2 0.05 0.10 gil7
BK.SAO.00 120.00 167.00 30 150 1.00 0 0 1 -121.45 36.76 bp
↳ 2 2 0.05 0.10 gil7
BK.CMB.00 123.00 78.00 30 150 1.00 0 0 0 -120.39 38.03 bp
↳ 2 2 0.05 0.10 gil7
BK.MNRC.00 132.00 333.00 30 150 1.00 1 1 1 -122.44 38.88 bp
↳ 2 2 0.05 0.10 gil7
| PREFERRED SOLUTION |
None
>>> tdmt.invert()
Deviatoric Moment Tensor Inversion
Depth = 10.0000 km
Mw = 3.97
Percent DC/CLVD/ISO = 100/0/0
VR = 100.00%%

Deviatoric Moment Tensor Inversion
Depth = 20.0000 km
Mw = 4.25
Percent DC/CLVD/ISO = 62/38/0
VR = 87.05%%
```

Classes

<i>Inversion</i>	A container for a single inverse routine
------------------	--

mttime.core.inversion.Inversion

class Inversion (*config=None*)

Bases: *object*

A container for a single inverse routine

Object contains the inversion parameters, data and synthetic files, and moment tensor solutions.

Parameters **config** (*Configure*) – object containing the necessary parameters for setting up the inverse routine

Variables

- **streams** (a list of *Stream*) – processed waveform data.
- **moment_tensors** (a list of *Tensor*) – moment tensor solutions.
- **preferred_tensor_id** (*int*) – index to the preferred moment tensor solution (maximum variance reduction).

Methods

<i>get_preferred_tensor</i>	Returns the preferred moment tensor solution
<i>invert</i>	Launch the inversion
<i>plot</i>	Plot inversion results
<i>write</i>	Write inversion results to file

mttime.core.inversion.Inversion.get_preferred_tensor

Inversion.**get_preferred_tensor**()

Returns the preferred moment tensor solution

A function that returns the solution with the highest variance reduction (goodness-of-fit between data and synthetic waveforms).

Returns the preferred moment tensor solution.

Return type a *Tensor* object.

mttime.core.inversion.Inversion.invert`Inversion.invert (show=False)`

Launch the inversion

Will perform multiple inversions if more than one source depth provided, and plot the results if attribute `config.plot=True`

mttime.core.inversion.Inversion.plot`Inversion.plot (**kwargs)`

Plot inversion results

Various options available to display the results.

Parameters

- **view** (*str*) – type of figure to produce. Default `waveform` creates the standard figure with focal mechanisms and waveform fits. `depth` shows the focal mechanism and moment magnitude as a function of depth. `map` plots stations and focal mechanisms in map view. `lune` plots the moment tensor source-type on a lune.
- **show** (*bool*) – If `True` will display figure after plotting and not save image to file, default is `False`.
- **format** (*str*) – figure file format, default is `"eps"`.
- **option** (*str*, *optional*) – Optional parameter if view is set to `waveform`. The default plots all solutions. Set to `preferred` to plot only the preferred solution.

mttime.core.inversion.Inversion.write`Inversion.write (option=None)`

Write inversion results to file

Save all solutions or only the preferred solution to file. The output file name format is `d[depth].mtinv.out`.

Parameters **option** (*str*) – option to write all solutions or only the preferred solution. Default is `None`. If set to `preferred` only the solution with the highest VR will be saved.

3.1.3 mttime.core.tensor

Routines for handling inversion results

Functions

<code>eigen2lune</code>	Calculate source-type parameters on a lune
<code>find_fault_planes</code>	Compute direction of slip and orientation of fault planes
<code>find_strike_rake_dip</code>	Compute strike,rake and dip
<code>get_m_in_basis</code>	Convert moment tensor between coordinate systems

mttime.core.tensor.eigen2lune

eigen2lune (*lam*)

Calculate source-type parameters on a lune

A function that calculates the source-type parameters gamma and delta based on the formulation of Tape and Tape, (2012).

Parameters **lam** (`ndarray`) – eigenvalues in descending order.

Returns lune coordinates gamma and delta.

Return type (`float`, `float`)

mttime.core.tensor.find_fault_planes

find_fault_planes (*M*, *M_dc*)

Compute direction of slip and orientation of fault planes

Function that returns slip direction and orientation of fault plane and auxiliary plane.

Parameters

- **M** (`ndarray`) – moment tensor in matrix form.
- **M_dc** (`ndarray`) – double-couple moment tensor in matrix form.

Returns direction of slip and orientation of the fault plane and auxiliary plane.

Return type `list((float, float, float), (float, float, float))`

mttime.core.tensor.find_strike_rake_dip

find_strike_rake_dip (*u*, *n*)

Compute strike,rake and dip

A function that returns the strike, rake and dip of given slip and fault normal vectors.

Parameters

- **u** (`ndarray`) – slip vector.
- **n** (`ndarray`) – fault normal vector.

Returns strike, rake and dip of a fault plane.

Return type (`float`, `float`, `float`)

mttime.core.tensor.get_m_in_basis

get_m_in_basis (*m*, *in_basis*, *out_basis*)

Convert moment tensor between coordinate systems

A function that returns the moment tensor in a new coordinate system. See supported coordinate systems below.

Parameters

- **m** (`ndarray` or list) – moment tensor elements (M11, M22, M33, M12, M13, M23) for a given coordinate system (1, 2, 3).
- **in_basis** (`str`) – input moment tensor coordinate system.
- **out_basis** (`str`) – output moment tensor coordinate system.

Returns moment tensor in a new coordinate system.

Return type `ndarray`

Supported coordinate systems:

basis	vectors	reference
"NED"	north, east, down	Jost and Herrmann, 1989
"XYZ"	north, east, down	Aki and Richard, 1980
"USE"	up, south, east	Larson et al., 2010
"RTP"	r, theta, phi	Harvard/Global CMT convention

Classes

Tensor

Object containing a single six-element moment tensor

mttime.core.tensor.Tensor

class Tensor (*m*, *basis*='XYZ', ***kwargs*)

Bases: `object`

Object containing a single six-element moment tensor

A container for all things related to the moment tensor *m*. Optional ***kwargs* are used to write and plot inversion results. Moment tensor elements will be stored in XYZ coordinates. Use `get_tensor_elements` to convert between coordinate systems.

Parameters

- **m** (`ndarray`) – six independent moment tensor elements (M11, M22, M33, M12, M13, M23) in dyne-cm.
- **basis** (`str`) – moment tensor coordinate system. Default is "XYZ".
- **depth** (`float`) – source depth.
- **inversion_type** (`str`) – Deviatoric or Full.
- **component** (`list(str)`) – waveform components.

- **station_table** (`DataFrame`) – station table. Required header names are: station, distance, azimuth, ts, dt, weights, VR, [ZRTNE] (one column for each component in component), longitude, and latitude.
- **total_VR** (`float`) – total variance reduction.
- **dd** (`ndarray`) – observed data stored as a single vector, required for plotting.
- **ss** (`ndarray`) – synthetic seismograms stored as a single vector, required for plotting.

Methods

<code>decompose</code>	Moment tensor decomposition
<code>get_decomposition</code>	Get the decomposition of a moment tensor
<code>get_tensor_elements</code>	Retrieve the moment tensor elements
<code>write</code>	Write detailed solution to file

mttime.core.tensor.Tensor.decompose

`Tensor.decompose()`

Moment tensor decomposition

A function that decomposes the moment tensor from attribute `m`. Moment is in dyne-cm.

Attributes

iso: `ndarray` isotropic moment tensor elements.

dev: `ndarray` deviatoric moment tensor elements.

dc: `ndarray` double couple moment tensor elements.

clvd: `ndarray` CLVD moment tensor elements.

eigenvalues: `ndarray` eigenvalues.

mo: `float` total scalar seismic moment in Bowers and Hudson (1999) convention.

mw: `float` moment magnitude.

mw_dev: `float` deviatoric moment magnitude.

mis: `float` isotropic moment.

mdc: `float` double-couple moment.

mclvd: `float` CLVD moment.

pd: `float` percentage of double-couple component.

pclvd: `float` percentage of CLVD component.

pis: `float` percentage of isotropic component.

fps: list of (`float`, `float`, `float`) strike, dip, and rake of the two fault planes.

lune: (`float`, `float`) gamma and delta in lune source-type space.

mttime.core.tensor.Tensor.get_decomposition**Tensor.get_decomposition()**

Get the decomposition of a moment tensor

A function that returns the decomposed moment tensor in a dictionary. Refer to [decompose](#) for the complete list of decomposition attributes.

Returns moment tensor decomposition.

Return type `dict`

mttime.core.tensor.Tensor.get_tensor_elements**Tensor.get_tensor_elements** (*basis*='XYZ')

Retrieve the moment tensor elements

Function that returns the moment tensor in the specified system coordinates.

Parameters **basis** (*str*) – system coordinates, default is XYZ. Refer to [get_m_in_basis](#) for supported systems.

Returns a dictionary of moment tensor elements.

Return type `dict`

mttime.core.tensor.Tensor.write**Tensor.write()**

Write detailed solution to file

A function that saves the detailed moment tensor solution to a file named `d[depth].mtinv.out`.

Attributes

m

Moment tensor elements

mttime.core.tensor.Tensor.m**property** `Tensor.m`

Moment tensor elements

MXX, MYX, MZZ, MXY, MXZ, MYZ in dyne-cm.

3.1.4 mtttime.core.utils

Utility functionality for mtttime

Functions

<i>RGF_from_SW4</i>	Function to convert reciprocal Green's functions from SW4 to tensor format
<i>az2baz</i>	Azimuth to back-azimuth conversion
<i>gaussj</i>	Solves the linear matrix equation $Ax = b$
<i>get_dependency_version</i>	Get version information of a dependency package.
<i>rotate_rt2ne</i>	Rotate to the great circle path.
<i>to_int_or_zero</i>	Converts given value to an integer or returns 0 if it fails.
<i>xcorr</i>	Compute cross-correlation coefficient

mttime.core.utils.RGF_from_SW4

RGF_from_SW4 (*path_to_green*='', *t0*=0, *file_name*=None, *origin_time*=None, *event_lat*=None, *event_lon*=None, *depth*=None, *station_name*=None, *station_lat*=None, *station_lon*=None, *output_directory*='sw4out')

Function to convert reciprocal Green's functions from SW4 to tensor format

Reads the reciprocal Green's functions (displacement/unit force) from SW4 and performs the summation to get the Green's function tensor. RGFs from SW4 are oriented north, east and positive down by setting *az*=0.

Assumes the following file structure: *f*[*x,y,z*]/*station_name*/*event_name*.*[x,y,z]*

Parameters

- **path_to_green** (*str*) – path to RGFs.
- **t0** (*float*) – offset in time (≥ 0).
- **file_name** (*str*) – name of RGF sac files from SW4 (event name).
- **origin_time** – event origin time.
- **event_lat** (*float*) – event latitude.
- **event_lon** (*float*) – event longitude
- **depth** (*float*) – source depth
- **station_name** (*list of strings*.) – station names.
- **station_lat** (*list of floats*) – station latitudes.
- **station_lon** (*list of floats*) – station longitudes.
- **output_directory** (*str*) – output direcotry.

mttime.core.utils.az2baz**az2baz** (*angle*)

Azimuth to back-azimuth conversion

Parameters **angle** (*float* or *int*) – azimuth value in degrees between 0 and 360.

Returns corresponding back-azimuth value in degrees.

Return type *float*

mttime.core.utils.gaussj**gaussj** (*A*, *b*)

Solves the linear matrix equation $Ax = b$

Computes the solution of a well-determined linear matrix equation by performing Gaussian-Jordan elimination for a given matrix and transforming it to a reduced echelon form.

Parameters

- **A** (*numpy.ndarray*) – coefficient matrix.
- **b** (*numpy.ndarray*) – vector of dependent variables.

Returns solution vector *x* so that $Ax = b$.

Return type *numpy.ndarray*

mttime.core.utils.get_dependency_version**get_dependency_version** (*package_name*, *raw_string=False*)

Get version information of a dependency package.

Parameters **package_name** (*str*) – Name of package to return version info for

Returns Package version as a list of three integers or *None* if import fails. With option *raw_string=True* returns raw version string instead (or *None* if import fails). The last version number can indicate different things like it being a version from the old svn trunk, the latest git repo, some release candidate version, ... If the last number cannot be converted to an integer it will be set to 0.

mttime.core.utils.rotate_rt2ne**rotate_rt2ne** (*r*, *t*, *az*)

Rotate to the great circle path.

Rotates a pair of horizontal component data from north and east direction to radial and tangential direction.

Parameters

- **r** (*ndarray*) – radial component data.
- **t** (*ndarray*) – tangential component data.
- **az** (*float*) – The direction from a seismic source to the seismic station measured clockwise from north.

Returns rotated horizontal component data oriented in north and east direction.

Return type Tuple of *ndarray*

mttime.core.utils.to_int_or_zero**to_int_or_zero** (*value*)

Converts given value to an integer or returns 0 if it fails.

Function taken from `misc`**Parameters** *value* – arbitrary data type.**Returns****Return type** `int`**mttime.core.utils.xcorr****xcorr** (*data*, *template*, *normalize=True*)

Compute cross-correlation coefficient

Parameters

- **data** (`numpy.ndarray`) – first time series.
- **template** (`numpy.ndarray`) – second time series.
- **normalize** (`bool`) – If `True` normalizes correlations.

Returns cross-correlation function.**Return type** `numpy.ndarray`

3.2 Plotting Modules

<code>mttime.imaging.source</code>	Plotting routines for mttime
<code>mttime.imaging.beachball</code>	Draws the beach ball diagram of a moment tensor

3.2.1 mttime.imaging.source

Plotting routines for mttime

Warning: This module should NOT be used directly instead use the class method `plot`.**Functions**

<code>beach_map</code>	Plot beach ball on a map
<code>beach_mw_depth</code>	Plot depth search result
<code>plot_lune</code>	Plot source-type on a lune
<code>plot_waveform_fits</code>	Plot waveform fits and focal mechanisms

mttime.imaging.source.beach_map

beach_map (*m, event, longitude, latitude, distance, used, show, format*)

Plot beach ball on a map

Function to plot stations and focal mechanisms on a map.

Parameters

- **m** (*list*) – focal mechanism.
- **event** (*dict*) – event origin time, longitude and latitude.
- **longitude** (list or *ndarray*) – station longitudes.
- **latitude** (list or *ndarray*) – station latitudes.
- **distance** (list or *ndarray*) – source-receiver distance.
- **used** (list or *ndarray*) – inverted stations. Sets the station marker colors, `True` for green and `False` for gray.
- **show** (*bool*) – display image after plotting instead of saving it to file.
- **format** (*str*) – figure file format.

mttime.imaging.source.beach_mw_depth

beach_mw_depth (*tensors, event, show, format*)

Plot depth search result

A function that plots focal mechanism, variance reduction, and moment magnitude as a function of source depth.

Parameters

- **tensors** (a list of *Tensor*) – moment tensor solutions at various depths.
- **event** (*dict*) – event origin time, longitude and latitude.
- **show** (*bool*) – display image after plotting instead of saving it to file.
- **format** (*str*) – figure file format.

mttime.imaging.source.plot_lune

plot_lune (*m, gamma, delta, show, format*)

Plot source-type on a lune

Function to plot moment tensor source type on a lune based on the formulation of Tape and Tape (2012). Theoretical source types and source type arcs are also plotted.

Parameters

- **m** – moment tensor.
- **gamma** (*float*) – lune longitude.
- **delta** (*float*) – lune latitude.
- **show** (*bool*) – display image after plotting instead of saving it to file.
- **format** (*str*) – figure file format.

mttime.imaging.source.plot_waveform_fits

plot_waveform_fits (*tensor*, *show*, *format*, *nrows=10*)

Plot waveform fits and focal mechanisms

Parameters

- **tensor** (*Tensor*) – moment tensor solution.
- **show** (*bool*) – display image after plotting instead of saving it to file.
- **format** (*str*) – figure file format.
- **nrows** (*int*) – maximum stations per page. Default is 10.

3.2.2 mttime.imaging.beachball

Draws the beach ball diagram of a moment tensor

Source codes are adapted from `mopad_wrapper`.

Functions

beach

Plot beach ball based on `MoPaD`

mttime.imaging.beachball.beach

beach (*fm*, *linewidth=1*, *facecolor='0.75'*, *bgcolor='w'*, *edgecolor='k'*, *alpha=1.0*, *xy=(0, 0)*, *width=200*, *size=100*, *nofill=False*, *zorder=100*, *mopad_basis='NED'*, *axes=None*, *show_iso=False*)

Plot beach ball based on `MoPaD`

Function that returns a beach ball as a collection and can be added to an existing `Axes`.

Parameters

- **fm** (*list*) – focal mechanism in (strike, dip, rake) or (M11,M22,M33,M12,M13,M23). The moment tensor elements are given for a coordinate system with axes pointing in three directions.
- **linewidth** (*float*) – width of nodal and border lines. Default is 0.8.
- **facecolor** (*str*) – color or shade of the compressive quadrants. Default is 0.75 (gray).
- **bgcolor** (*str*) – background color, default is w (white).
- **edgecolor** (*str*) – color of nodal and border lines, default is b (black).
- **alpha** (*float*) – beach ball transparency, default is 1.0 (opaque).
- **xy** (*tuple*) – original position of the beach ball. Default is (0, 0)
- **width** (*float*) – width of the beach ball (aka symbol size). Default is 200.
- **size** (*int*) – number of points interpolated to draw the curve. Default is 100.
- **nofill** (*bool*) – no shading of the beach ball. Default is False.
- **zorder** (*float*) – set the zorder for the artist. Artists with lower zorder values are drawn first. Default is 100.
- **mopad_basis** (*str*) – moment tensor coordinate system. Default is "NED".

- **axes** (*Axes*) – figure axis for beach ball, this is used to ensure the aspect ratio is adjusted so that the beach ball is circular on non-scaled axes. When this option is used figure cannot be saved in vector format. Default is `None`.
- **show_iso** (*bool*) – flag to display the isotropic component, default is `False`.

Returns a collection of lines and polygons.

Return type `PatchCollection`

Warning: Set `axes=None` if you want to save the beach ball image in vector file formats.

3.3 Scripts

`mttime.scripts.run`

mttime command line utility

3.3.1 mttime.scripts.run

Note: This script automatically installs during the setup procedure with the name `$ tdmtpy-run`. For more info on the command line options, please run `$ tdmtpy-run --help`. Alternatively you can also execute `$ python -m mttime.scripts.run`.

mttime command line utility

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

Note: The code is still under development, your feedback is appreciated.

PYTHON MODULE INDEX

C

`mttime.core.configure`, [15](#)
`mttime.core.inversion`, [17](#)
`mttime.core.tensor`, [19](#)
`mttime.core.utils`, [24](#)

i

`mttime.imaging.beachball`, [28](#)
`mttime.imaging.source`, [26](#)

S

`mttime.scripts.run`, [29](#)

A

az2baz() (in module *mttime.core.utils*), 25

B

beach() (in module *mttime.imaging.beachball*), 28

beach_map() (in module *mttime.imaging.source*), 27

beach_mw_depth() (in module *mttime.imaging.source*), 27

C

Configure (class in *mttime.core.configure*), 15

D

decompose() (Tensor method), 22

E

eigen2lune() (in module *mttime.core.tensor*), 20

F

find_fault_planes() (in module *mttime.core.tensor*), 20

find_strike_rake_dip() (in module *mttime.core.tensor*), 20

G

gaussj() (in module *mttime.core.utils*), 25

get_decomposition() (Tensor method), 23

get_dependency_version() (in module *mttime.core.utils*), 25

get_m_in_basis() (in module *mttime.core.tensor*), 21

get_preferred_tensor() (Inversion method), 18

get_tensor_elements() (Tensor method), 23

I

Inversion (class in *mttime.core.inversion*), 18

invert() (Inversion method), 19

M

m() (Tensor property), 23

module

mttime.core.configure, 15

mttime.core.inversion, 17

mttime.core.tensor, 19

mttime.core.utils, 24

mttime.imaging.beachball, 28

mttime.imaging.source, 26

mttime.scripts.run, 29

mttime.core.configure
module, 15

mttime.core.inversion
module, 17

mttime.core.tensor
module, 19

mttime.core.utils
module, 24

mttime.imaging.beachball
module, 28

mttime.imaging.source
module, 26

mttime.scripts.run
module, 29

P

plot() (Inversion method), 19

plot_lune() (in module *mttime.imaging.source*), 27

plot_waveform_fits() (in module *mttime.imaging.source*), 28

R

RGF_from_SW4() (in module *mttime.core.utils*), 24

rotate_rt2ne() (in module *mttime.core.utils*), 25

T

Tensor (class in *mttime.core.tensor*), 21

to_int_or_zero() (in module *mttime.core.utils*), 26

W

write() (Configure method), 16

write() (Inversion method), 19

write() (Tensor method), 23

X

xcorr() (in module *mttime.core.utils*), 26