QuakeMigrate

Release 1.0.0

Nov 09, 2020

Contents

`QuakeMigrate` is a Python package for the detection and location of earthquakes using waveform migration and stacking.

QuakeMigrate uses a waveform stacking algorithm to search for coherent seismic phase arrivals across a network of instruments. It produces, from raw data, a catalogue of earthquakes with locations, origin times and phase arrival picks, as well as estimates of the uncertainties associated with these measurements.

The source code for the project is hosted on .

CHAPTER 1

Supported operating systems

QuakeMigrate was developed and tested on Ubuntu 16.04/18.04, with the intention of being "platform agnostic". As of July 2020, the package has been successfully built and run on:

- Ubuntu 16.04/18.04/20.04
- Red Hat Enterprise Linux
- Windows 10
- macOSX High Sierra 10.13.6

CHAPTER 2

Citation

If you use this package in your work, please cite the following paper:

Bacon, C.A., Smith, J.D., Winder, T., Hudson, T., Greenfield, T. and White, R.S. QuakeMigrate: a Modular, Open-Source Python Package for Earthquake Detection and Location. In AGU Fall Meeting 2019. AGU.

or, if this is not possible, please cite the following journal article:

Smith, J.D., White, R.S., Avouac, JP, and S. Bourne (2020), Probabilistic earthquake locations of induced seismicity in the Groningen region, Netherlands, Geophysical Journal International.

We hope to have a publication coming out soon:

Winder, T., Smith, J.D., Bacon, C.A., Hudson, T.S., Drew, J., Greenfield, T. and White, R.S. QuakeMigrate: a Python Package for Automatic Earthquake Detection and Location Using Waveform Migration and Stacking. Seismological Research Letters.

CHAPTER 3

Contact

You can contact us directly at - quakemigrate.developers@gmail.com

Any additional comments/questions can be directed to: * **Tom Winder** - tom.winder@esc.cam.ac.uk * **Conor Bacon** - conor.bacon@esc.cam.ac.uk

CHAPTER 4

License

CHAPTER 5

Contents:

5.1 Installation

`QuakeMigrate` is a predominantly Python package with some routines written and optimised in C. These are built and linked to QuakeMigrate at installation, which means you will need to ensure that there is a suitable compiler available (more details below).

5.1.1 Supported operating systems

QuakeMigrate was developed and tested on Ubuntu 16.04/18.04, with the intention of being "platform agnostic". As of July 2020, the package has been successfully built and run on:

- Ubuntu 16.04/18.04/20.04
- Red Hat Enterprise Linux
- Debian
- Windows 10
- macOSX High Sierra 10.13.6

5.1.2 Prerequisites

QuakeMigrate supports Python 3.6 or newer (3.7/3.8). We recommend using Anaconda as a package manager and environment management system to isolate and install the specific dependencies of QuakeMigrate. Instructions for downloading and installing Anaconda can be found here. If drive space is limited, consider using Miniconda instead, which ships with a minimal collection of useful packages.

Setting up an environment

Using conda, you can use our quakemigrate.yml file to create and activate a minimally complete environment:

```
conda env create -f quakemigrate.yml
conda activate quakemigrate
```

This will install the explicit dependencies of QuakeMigrate (as well as some additional sub-dependencies/useful packages). The full list of dependencies (and versions, where relevant) is:

- matplotlib < 3.3
- numpy
- obspy >= 1.2
- pandas >= 1 and < 1.1
- pyproj >= 2.6
- scipy

Note: These version pins are subject to change. We defer to ObsPy to select suitable versions for NumPy/SciPy.

Warning: Some changes to datetime handling were introduced in matplotlib 3.3, which caused some conflicts with pandas versions <= 1.0.5. A patch was applied, but for the time being we have pinned these two packages until we find time to fully resolve the issues arising from these changes.

In addition, we use NonLinLoc and scikit fmm as backends for producing 1-D traveltime lookup tables.

NonLinLoc

To download, unpack, and compile NonLinLoc, you can use:

```
curl http://alomax.free.fr/nlloc/soft7.00/tar/NLL7.00_src.tgz -o NLL7.00_src.tgz
tar -xzvf NLL7.00_src.tgz
cd src
mkdir bin; export MYBIN=./bin
make -R all
```

Once the source code has been compiled, we recommend you add the bin to your system path. For Unix systems, this can be done by adding the following to your .bashrc file (typically found in your home directory, `~/`):

```
export PATH=/path/to/nonlinloc/bin:$PATH
```

replacing the `/path/to/nonlinloc` with the path to where you downloaded/installed NonLinLoc. Save your .bashrc and open a new terminal window to activate the change. This will allow your shell to access the `Vel2Grid` and `Grid2Time` programs anywhere.

scikit-fmm

scikit-fmm is a 3rd-party package which implements the fast-marching method. We specify the version `2019.1.30` as previous versions did not catch a potential numerical instability which may lead to unphysical traveltimes. It can be installed using:

```
pip install scikit-fmm==2019.1.30
```

It can also be installed along with the rest of package (detailed below).

Note: In order to install scikit-fmm, you will need an accessible C++ compiler, such as gxx (see below for details).

C compilers

In order to install and use QuakeMigrate, you will need a C compiler that will build the migration extension library.

If you already have a suitable compiler (e.g. gcc, MSVC) at the OS level, then you can proceed to the Installing section.

If you do not, or to be sure, we recommend installing a compiler using conda. Instructions for doing this on Linux and macOSX operating systems are given below.

Note: In order to build the (optional) dependency scikit-fmm you will need a C++ compiler (e.g. gxx, MSVC). This can also be done either at the OS level, or using conda (see guidance on the conda compiler tools page, linked below).

Linux

We recommend installing the GNU compiler collection (GCC, which previously stood for the GNU C Compiler) through conda.

```
conda install gcc_linux-64
```

It is generally useful to install compilers at the OS level, including a C++ compiler (e.g. gxx), which is required to build the scikit-fmm package.

Once installed, you can proceed with the QuakeMigrate installation.

macOS

As with Linux, we recommend installing GCC through conda.

```
conda install gcc
```

Note: We have not yet tested compiling and/or running QuakeMigrate against the Clang compiler.

Installation of compilers at an OS level can be done using `Homebrew`, a package manager for macOS. It is then as simple as:

```
brew install gcc
```

Once installed, you can proceed with the QuakeMigrate installation.

Windows

Compilation and linking of the C extensions has been successful using the Microsoft Visual C++ (MSVC) build tools. We strongly recommend that you download and install these tools in order to use QuakeMigrate. You can either install

Visual Studio in its entirety, or just the Build Tools - available here. You will need to restart your computer once the installation process has completed. We recommend using the anaconda command line interface (unix shell-like) to install QuakeMigrate over command prompt.

> **Warning:** QuakeMigrate has been tested and validated on Windows, but there may yet remain some unknown issues. If you encounter an issue (and/or resolve it), please let us know!

Once installed, you can proceed with the QuakeMigrate installation.

5.1.3 Installing

There are a few ways to get a copy of QuakeMigrate:

From source

Clone the repository from our GitHub (note: you will need `git` installed on your system), or alternatively download the source code directly through the GitHub web interface. Once you have a local copy, navigate to the new QuakeMigrate directory and run (ensuring your environment is activated):

```
pip install .
```

You can optionally pass a `-e` argument to install the package in 'editable' mode.

If you wish to use `scikit-fmm`, you can install it here as an optional package using:

```
pip install .[fmm]
```

You should now be able to import quakemigrate within a Python session:

```
python
>>> import quakemigrate
```

pip install

We will be linking the package to PyPI (the Python Package Index) soon, after which you will be able to use the following command to install the package:

```
pip install quakemigrate
```

conda install

We hope to link the package with the conda forge soon, after which you will be able to use the following command to install the package:

```
conda install -c conda-forge quakemigrate
```

5.1.4 Testing your installation

In order to test your installation, you will need to have cloned the GitHub repository. This will ensure you have all of the required benchmarked data (which is not included in pip/conda installs). Then, navigate to `QuakeMigrate/examples/Icequake_Iceland` and run the example scripts in the following order:

```
python iceland_lut.py
python iceland_detect.py
python iceland_trigger.py
python iceland_locate.py
```

Once these have all run successfully, navigate to `QuakeMigrate/tests` and run:

```
python test_benchmarks.py
```

This should execute with no failed tests.

Note: We hope to work this into a more complete suite of tests that can be run in a more automated sense.

5.1.5 Notes

There is a known issue with PROJ version 6.2.0 which causes vertical coordinates to be incorrectly transformed when using units other than metres (the PROJ default). If you encounter this issue (you will get an `ImportError` when trying to use the `lut` subpackage), you should update `pyproj`. Using conda will install an up-to-date PROJ backend, but you may need to clear your cache of downloaded packages. This can be done using:

```
conda clean --all
```

Then reinstall `pyproj`:

```
conda uninstall pyproj
conda install pyproj
```

5.2 Tutorials

Here we provide a few tutorials that explore each element of the package in more detail and provide code snippets the user can use in their own research.

5.2.1 The traveltime lookup table

This tutorial will cover the basic ideas and definitions underpinning the traveltime lookup table, as well as showing how they can be created.

In order to reduce computational costs during runtime, we pre-compute traveltime lookup tables (LUTs) for each seismic phase and each station in the network to every node in a regularised 3-D grid. This grid spans the volume of interest, herein termed the coalescence volume, within which QuakeMigrate will search for events.

Defining the underlying 3-D grid

Before we can create our traveltime lookup table, we have to define the underlying 3-D grid which spans the volume of interest.

Coordinate projections

First, we choose a pair of coordinate reference systems to represent the input coordinate space (`cproj`) and the Cartesian grid space (`gproj`). We do this using *pyproj*, which provides the Python bindings for the PROJ library. It is important to think about which projection is best suited to your particular study region. More information can be found [in their documentation](https://pyproj4.github.io/pyproj/stable/).

> **Warning:** The default units of `Proj` are *metres*! It is strongly advised that you explicitly state which units you wish to use.

We use here the WGS84 reference ellipsoid (used as standard by the Global Positioning System) as our input space and the Lambert Conformal Conic projection to form our Cartesian space. The units of the Cartesian space are specified as kilometres. The values used in the LCC projection are for a study region in Sabah, Borneo.

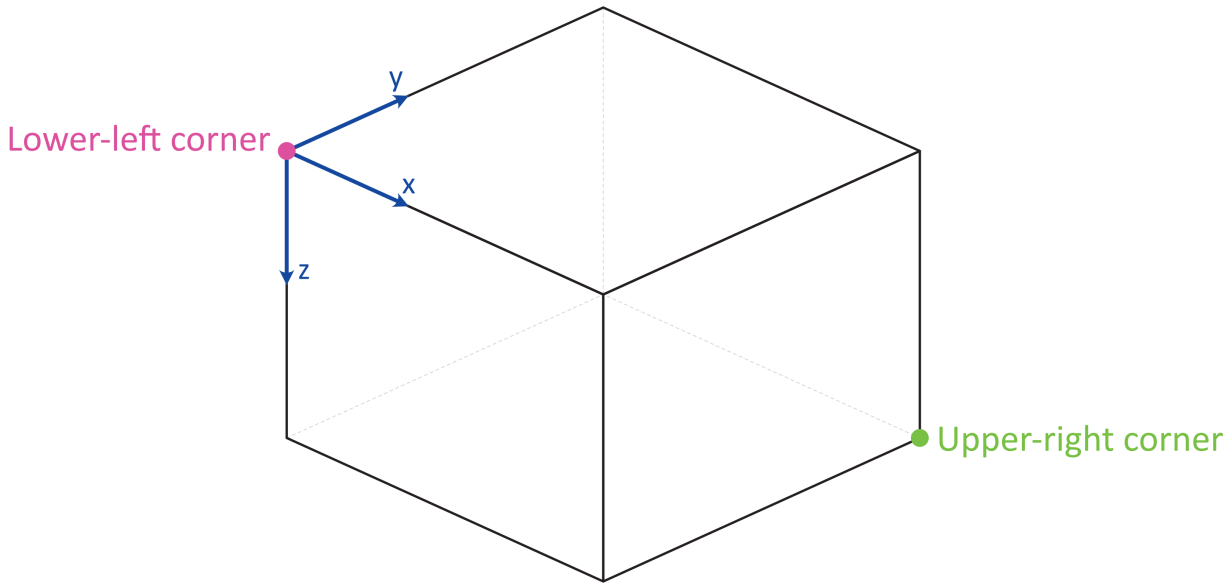
```
from pyproj import Proj

cproj = Proj(proj="longlat", ellps="WGS84", datum"=WGS84", no_defs=True)
gproj = Proj(proj="lcc", lon_0=116.75, lat_0=6.25, lat_1=4.0, lat_2=7.5,
         datum="WGS84", ellps="WGS84", units="km", no_defs=True)
```

Geographical location and spatial extent

In order to geographically situate our lookup table, we choose two reference points in the input coordinate space, herein called the lower-left and upper-right corners (`ll_corner` and `ur_corner`, respectively). By default, we work in a depth-positive frame (i.e. positive-down or left-handed coordinate system) and use units of kilometres. It is possible to run QuakeMigrate with distances measured in metres, as long as the user specifies this requirement when defining the grid projection and all inputs (station elevations, grid specification, velocities, etc) are in metres.

This schematic shows the relative positioning of the two corners:



The final piece of information required to fully define the grid on which we will compute traveltimes is the spacing (in each dimension, *x*, *y*, *z*) between each node in the grid (`node_spacing`). The LUT class will automatically find the number of nodes required in each dimension to span the specified geographical region. If the node spacing doesn't fit into the corresponding grid dimension an integer number of times, the location of the upper-right corner is shifted to accommodate an additional node.

Note: The corners (`ll_corner` and `ur_corner`) are nodes - hence a grid that is 20 x 20 x 20 km, with 2 km node spacing in each dimension, will have 11 nodes in x, y, and z.

```
ll_corner = [116.075, 5.573, -1.750]
ur_corner = [117.426, 6.925, 27.750]
node_spacing = [0.5, 0.5, 0.5]
```

Bundling the grid specification

The grid specification needs to be bundled into a dictionary to be used as an input for the *compute_traveltimes* function. We use here the `AttribDict` from ObsPy, which extends the standard Python *dict* data structure to also have .-style access.

```
grid_spec = AttribDict()
grid_spec.ll_corner = ll_corner
grid_spec.ur_corner = ur_corner
grid_spec.node_spacing = node_spacing
grid_spec.grid_proj = gproj
grid_spec.coord_proj = cproj
```

Computing traveltimes

We have bundled a few methods of computing traveltimes into QuakeMigrate.

In addition to the grid specification, we need to provide a list of stations for which to compute traveltime tables.

```
from quakemigrate.io import read_stations

stations = read_stations("/path/to/station_file")
```

The *read_stations* function is a passthrough for *pandas.read_csv*, so we can handle any delimiting characters (e.g. by specifying *read_stations("station_file", delimiter=",")*). There are four required (case-sensitive) column headers - "Name", "Longitude", "Latitude", "Elevation".

Note: Station elevations are in the positive-up/right-handed coordinate frame. An elevation of 2 would correspond to 2 (km) above sea level.

The *compute_traveltimes* function used in the following sections returns a lookup table (a fully-populated instance of the LUT class) which can be used for *detect*, *trigger*, and *locate*.

Homogeneous velocity model

Simply calculates the straight line traveltimes between stations and points in the grid. It is possible to use stations that are outside the specified span of the grid if desired. For example, if you have a good prior constraint on the possible location of the seismicity you are hoping to detect; for basal icequakes you may limit the LUT grid to span a small range of depths around the ice-bed interface. Any reduction in grid size can greatly reduce the computational cost of running QuakeMigrate, as runtime scales with the number of nodes - so *n^3* for an equidimensional lookup table grid of side-length *n*.

```
from quakemigrate.lut import compute_traveltimes

compute_traveltimes(grid_spec, stations, method="homogeneous", vp=5., vs=3.,
                    log=True, save_file=/path/to/save_file)
```

1-D velocity models

1-D velocity models are read in from an (arbitrarily delimited) textfile using *quakemigrate.io.read_vmodel*. There is only 1 required (case-sensitive) column header - "Depth", which corresponds to the depths for each block in the velocity model. Each additional column should contain a velocity model that corresponds to a particular seismic phase, with a (case-sensitive) header, e.g. *Vp* (Note: Uppercase *V*, lowercase phase code).

Note: The units for velocities should correspond to the units used in specifying the grid projection. km -> km / s; m -> m / s.

Note: Depths are in the positive-down/left-handed coordinate frame. A depth of 5 would correspond to 5 (km) below sea level.

1-D fast-marching method

The fast-marching method implicitly tracks the evolution of the wavefront. Our current backend is the *scikit-fmm* package. It is possible to use this package to compute traveltimes to 1-D, 2-D, or 3-D velocity models. Currently we provide a utility function that computes traveltime tables for 1-D velocity models. The format of this velocity model file is specified below. See the *scikit-fmm* documentation and Rawlinson & Sambridge (2005) for more details.

Note: Traveltime calculation can only be performed between grid nodes: the station location is therefore taken as the closest grid node. Note that for large node spacings this may cause a modest error in the calculated traveltimes.

Note: All stations must be situated within the grid on which traveltimes are to be computed.

```
from quakemigrate.lut import compute_traveltimes
from quakemigrate.io import read_vmodel

vmod = read_vmodel("/path/to/vmodel_file")
compute_traveltimes(grid_spec, stations, method="1dfmm", vmod=vmod,
                    log=True, save_file=/path/to/save_file)
```

1-D NonLinLoc-style sweep

Uses the Eikonal solver from NonLinLoc under the hood to generate a traveltime grid for a 2-D slice that passes through the station and the point in the grid furthest away from that station. This slice is then "swept" using a bilinear interpolation scheme to produce a 3-D traveltime grid. The format of the input velocity model file is specified below. This also has the benefit of being able to include stations outside of the volume of interest, without having to increase the size of the grid.

Note: Requires the user to install the NonLinLoc software package (available from http://alomax.free.fr/nlloc/)

```
from quakemigrate.lut import compute_traveltimes
from quakemigrate.io import read_vmodel

vmod = read_vmodel("/path/to/vmodel_file")
compute_traveltimes(grid_spec, stations, method="1dsweep", vmod=vmod,
                    block_model=True, log=True, save_file=/path/to/save_file)
```

Other formats

It is also easy to import traveltime lookup tables generated by other means. We have provided a parser for lookup tables in the NonLinLoc format (`read_nlloc()`). It is straightforward to adapt this code to read any other traveltime lookup table, so long as it is stored as an array. Create an instance of the LUT class with the correct grid dimensions, then add the (C-ordered) traveltime arrays to the `LUT.traveltimes` dictionary using:

```
lut.traveltimes.setdefault(STATION, {}).update(
    {PHASE.upper(): traveltime_table})
```

where *STATION* and *PHASE* are station name and seismic phase strings, respectively.

Saving your LUT

If you provided a `save_file` argument to the `compute_traveltimes` function, the LUT will already be saved. In any case, the lookup table object is returned by the *compute_traveltimes* function if you wish to explore the object further. We use the *pickle* library (a Python standard library) to serialise the LUT, which essentially freezes the state of the LUT. If you have added 3rd-party traveltime lookup tables to the LUT, you will need to save using:

```
lut.save("/path/to/output/lut")
```

Reading in a saved LUT

When running the main stages of QuakeMigrate (*detect*, *trigger*, and *locate*) it is necessary to read in the saved LUT, which can be done as:

```
from quakemigrate.io import read_lut
lut = read_lut(lut_file="/path/to/lut_file")
```

5.3 Source code

Explore the documentation and source code for the QuakeMigrate package.

5.3.1 quakemigrate.core

The *quakemigrate.core* module provides Python bindings for the library of compiled C routines that form the core of QuakeMigrate:

- Migrate onsets - This routine performs the continuous migration through time and space of the onset functions. It has been parallelised with openMP.
- Find maximum coalescence - This routine finds the continuous maximum coalescence amplitude in the 4-D coalesence volume.

  copyright 2020, QuakeMigrate developers.

  license GNU General Public License, Version 3 (https://www.gnu.org/licenses/gpl-3.0.html)

Functions

Bindings for the C library functions, migrate and find_max_coa.

copyright 2020, QuakeMigrate developers.

license GNU General Public License, Version 3 (https://www.gnu.org/licenses/gpl-3.0.html)

quakemigrate.core.lib.**find_max_coa**(*map4d*, *threads*)

Finds time series of the maximum coalescence/normalised coalescence in the 3-D volume, and the corresponding grid indices.

Parameters

- **map4d** (*numpy.ndarray* of *numpy.double*) – 4-D coalescence map, shape(nx, ny, nz, nsamples).
- **threads** (*int*) – Number of threads with which to perform the scan.

Returns

- **max_coa** (*numpy.ndarray* of *numpy.double*) – Time series of the maximum coalescence value in the 3-D volume.
- **max_norm_coa** (*numpy.ndarray* of *numpy.double*) – Times series of the maximum normalised coalescence value in the 3-D volume.

- **max_coa_idx** (*numpy.ndarray* of int) – Time series of the flattened grid indices corresponding to the maximum coalescence value in the 3-D volume.

quakemigrate.core.lib.**migrate**(*onsets*, *traveltimes*, *first_idx*, *last_idx*, *available*, *threads*)

Computes 4-D coalescence map by migrating seismic phase onset functions.

Parameters

- **onsets** (*numpy.ndarry* of float) – Onset functions for each seismic phase, shape(nstations, nsamples).
- **traveltimes** (*numpy.ndarry* of int) – Grids of seismic phase traveltimes converted to an integer multiple of the sampling rate, shape(nx, ny, nz, nstations).
- **first_idx** (*int*) – Index of first sample in array from which to scan.
- **last_idx** (*int*) – Index of last sample in array up to which to scan.
- **available** (*int*) – Number of available onset functions.
- **threads** (*int*) – Number of threads with which to perform the scan.

Returns **map4d** – 4-D coalescence map, shape(nx, ny, nz, nsamples).

Return type *numpy.ndarray* of *numpy.double*

Raises

- ValueError – If mismatch between number of onset functions and traveltime lookup tables - expect both to be equal to the no. stations * no. phases.
- ValueError – If the number of samples in the onset functions is less than the number of samples array is smaller than map4d[0, 0, 0, :].

5.3.2 quakemigrate.export

The *quakemigrate.export* module provides some utility functions to export the outputs of QuakeMigrate to other catalogue formats/software inputs:

- Input files for NonLinLoc
- ObsPy Catalog object
- Snuffler pick/event files for manual phase picking
- MFAST for shear-wave splitting analysis

Warning: Export modules are an ongoing work in progress. The functionality

of the core module *to_obspy* has been validated, but there may still be bugs elsewhere. If you are interested in using these, or wish to add additional functionality, please contact the QuakeMigrate developers at quakemigrate.developers@gmail.com .

quakemigrate.export.to_mfast

This module provides parsers to generate SAC waveform files from an ObsPy Catalog, with headers correctly populated for MFAST.

quakemigrate.export.to_mfast.**sac_mfast**(*event*, *stations*, *output_path*, *filename=None*)

Function to create the SAC file.

Parameters

- **event** (*ObsPy Event object*) – Contains information about the origin time and a list of associated picks.
- **stations** (*pandas DataFrame*) – DataFrame containing station information.
- **output_path** (*str*) – Location to save the SAC file.
- **filename** (*str, optional*) – Name of SAC file - defaults to "eventid/eventid.station.{comp}".

quakemigrate.export.to_nlloc

This module provides parsers to export an ObsPy Catalog to the NonLinLoc input file format. We prefer this to the one offered by ObsPy as it includes the additional weighting term.

quakemigrate.export.to_nlloc.**nlloc_obs**(*event*, *filename*)

Write a NonLinLoc Phase file from an obspy Catalog object.

Parameters

- **event** (*obspy Catalog object*) – Contains information on a single event.
- **filename** (*str*) – Name of NonLinLoc phase file.

quakemigrate.export.to_obspy

This module provides parsers to export the output of a QuakeMigrate run to an ObsPy Catalog.

quakemigrate.export.to_obspy.**read_quakemigrate**(*run_dir*, *units*, *run_subname=''*, *local_mag_ph='S'*)

Reads the .event and .picks outputs, and .amps outputs if available, from a QuakeMigrate run into an obspy Catalog object.

NOTE: if a station_corrections dict was used to calculate the network-averaged local magnitude, this information will not be included in the obspy event object. There might therefore be a discrepancy between the mean of the StationMagnitudes and the event magnitude.

Parameters

- **run_dir** (*str*) – Path to QuakeMigrate run directory.

- **units** (*{"km", "m"}*) – Grid projection coordinates for QM LUT (determines units of depths and uncertainties in the .event files).
- **run_subname** (*str, optional*) – Run_subname string (if applicable).
- **local_mag_ph** (*{"S", "P"}, optional*) – Amplitude measurement used to calculate local magnitudes. (Default "S")

Returns **cat** – Catalog containing events in the specified QuakeMigrate run directory.

Return type *obspy.Catalog* object

quakemigrate.export.to_snuffler

This module provides parsers to generate input files for Snuffler, a manual phase picking interface from the Pyrocko package.

copyright 2020, QuakeMigrate developers.

license GNU General Public License, Version 3 (https://www.gnu.org/licenses/gpl-3.0.html)

quakemigrate.export.to_snuffler.**snuffler_markers**(*event*, *output_path*, *filename=None*)

Function to create marker files compatible with snuffler

Parameters

- **event** (*ObsPy Event object*) – Contains information about the origin time and a list of associated picks
- **output_path** (*str*) – Location to save the marker file
- **filename** (*str, optional*) – Name of marker file - defaults to eventid/eventid.markers

quakemigrate.export.to_snuffler.**snuffler_stations**(*stations*, *output_path*, *filename*, *network_code=None*)

Function to create station files compatible with snuffler.

Parameters

- **stations** (*pandas DataFrame*) – DataFrame containing station information.
- **output_path** (*str*) – Location to save snuffler station file.
- **filename** (*str*) – Name of output station file.
- **network_code** (*str*) – Unique identifier for the seismic network.

5.3.3 quakemigrate.io

The *quakemigrate.io* module handles the various input/output operations performed by QuakeMigrate. This includes:

- Reading waveform data - The submodule data.py can handle any waveform data archives with regular directory structures.
- Writing results - The submodule quakeio.py provides a suite of functions to output QuakeMigrate results in the QuakeMigrate format.
- Parse QuakeMigrate results into the ObsPy Catalog structure.
- Various parsers to input files for different pieces of software. Feel free to contribute more!

copyright 2020, QuakeMigrate developers.

license GNU General Public License, Version 3 (https://www.gnu.org/licenses/gpl-3.0.html)

quakemigrate.io.amplitudes

Module to handle input/output of .amps files.

copyright 2020, QuakeMigrate developers.

license GNU General Public License, Version 3 (https://www.gnu.org/licenses/gpl-3.0.html)

`quakemigrate.io.amplitudes.`**`write_amplitudes`**(*run*, *amplitudes*, *event*)

Write amplitude results to a new .amps file. This includes amplitude measurements, and the magnitude estimates derived from them (with station correction terms appied, if provided).

Parameters

- **`run`** (`Run` object) – Light class encapsulating i/o path information for a given run.
- **`amplitudes`** (*pandas.DataFrame* object) – P- and S-wave amplitude measurements for each component of each station in the station file, and individual local magnitude estimates derived from them. Columns = ["epi_dist", "z_dist", "P_amp", "P_freq", "P_time",

  "S_amp", "S_freq", "S_time", "Noise_amp", "is_picked", "ML", "ML_Err"]

  Index = Trace ID (see *obspy.Trace* object property 'id')
- **`event`** (`Event` object) – Light class encapsulating signal, onset, and location information for a given event.

quakemigrate.io.availability

Module to handle input/output of StationAvailability.csv files.

copyright 2020, QuakeMigrate developers.

license GNU General Public License, Version 3 (https://www.gnu.org/licenses/gpl-3.0.html)

`quakemigrate.io.availability.`**`read_availability`**(*run*, *starttime*, *endtime*)

Read in station availability data to a *pandas.DataFrame* from csv files split by Julian day.

Parameters

- **`run`** (`Run` object) – Light class encapsulating i/o path information for a given run.
- **`starttime`** (*obspy.UTCDateTime* object) – Timestamp from which to read the station availability.
- **`endtime`** (*obspy.UTCDateTime* object) – Timestamp up to which to read the station availability.

Returns **availability** – Details the availability of each station for each timestep of detect.

Return type *pandas.DataFrame* object

`quakemigrate.io.availability.`**`write_availability`**(*run*, *availability*)

Write out csv files (split by Julian day) containing station availability data.

Parameters

- **`run`** (`Run` object) – Light class encapsulating i/o path information for a given run.
- **`availability`** (*pandas.DataFrame* object) – Details the availability of each station for each timestep of detect.

quakemigrate.io.core

Module to handle input/output for QuakeMigrate.

copyright 2020, QuakeMigrate developers.

license GNU General Public License, Version 3 (https://www.gnu.org/licenses/gpl-3.0.html)

class quakemigrate.io.core.**Run**(*path*, *name*, *subname=''*, *stage=None*, *loglevel='info'*)
Bases: object

Light class to encapsulate i/o path information for a given run.

Parameters

- **stage** (*str*) – Specifies run stage of QuakeMigrate ("detect", "trigger", or "locate").
- **path** (*str*) – Points to the top level directory containing all input files, under which the specific run directory will be created.
- **name** (*str*) – Name of the current QuakeMigrate run.
- **subname** (*str, optional*) – Optional name of a sub-run - useful when testing different trigger parameters, for example.

path
Points to the top level directory containing all input files, under which the specific run directory will be created.

Type *pathlib.Path* object

name
Name of the current QuakeMigrate run.

Type str

run_path
Points to the run directory into which files will be written.

Type *pathlib.Path* object

subname
Optional name of a sub-run - useful when testing different trigger parameters, for example.

Type str

stage
Track which stage of QuakeMigrate is being run.

Type {"detect", "trigger", "locate"}, optional

loglevel
Set the logging level. (Default "info")

Type {"info", "debug"}, optional

logger(*log*)
Spins up a logger configured to output to stdout or stdout + log file.

logger(*log*)
Configures the logging feature.

Parameters **log** (*bool*) – Toggle for logging. If True, will output to stdout and generate a log file.

name
Get the run name as a formatted string.

quakemigrate.io.core.**read_lut**(*lut_file*)
Read the contents of a pickle file and restore state of the lookup table object.

Parameters **lut_file** (*str*) – Path to pickle file to load.

Returns **lut** – Lookup table populated with grid specification and traveltimes.

Return type LUT object

quakemigrate.io.core.**read_response_inv**(*response_file*, *sac_pz_format=False*)
Reads response information from file, returning it as a *obspy.Inventory* object.

Parameters

- **response_file** (*str*) – Path to response file. Please see the *obspy.read_inventory()* documentation for a full list of supported file formats. This includes a dataless.seed volume, a concatenated series of RESP files or a stationXML file.
- **sac_pz_format** (*bool, optional*) – Toggle to indicate that response information is being provided in SAC Pole-Zero files. NOTE: not yet supported.

Returns **response_inv** – ObsPy response inventory.

Return type *obspy.Inventory* object

Raises

- NotImplementedError – If the user selects sac_pz_format.
- TypeError – If the user provides a response file that is not readable by ObsPy.

quakemigrate.io.core.**read_stations**(*station_file*, ***kwargs*)
Reads station information from file.

Parameters

- **station_file** (*str*) – Path to station file. File format (header line is REQUIRED, case sensitive, any order):

  Latitude, Longitude, Elevation (units of metres), Name

- **kwargs** (*dict*) – Passthrough for *pandas.read_csv* kwargs.

Returns **stn_data** – Columns: "Latitude", "Longitude", "Elevation", "Name"

Return type *pandas.DataFrame* object

Raises StationFileHeaderException – Raised if the input file is missing required entries in the header.

quakemigrate.io.core.**read_vmodel**(*vmodel_file*, ***kwargs*)
Reads velocity model information from file.

Parameters

- **vmodel_file** (*str*) – Path to velocity model file. File format: (header line is REQUIRED, case sensitive, any order): Depth (units of metres), Vp, Vs (units of metres per second)
- **kwargs** (*dict*) – Passthrough for *pandas.read_csv* kwargs.

Returns **vmodel_data** – Columns: "Depth", "Vp", "Vs"

Return type *pandas.DataFrame* object

Raises `VelocityModelFileHeaderException` – Raised if the input file is missing required entries in the header.

`quakemigrate.io.core.`**`stations`**(*station_file*, ***kwargs*)
Alias for read_stations.

quakemigrate.io.cut_waveforms

Module to handle input/output of cut waveforms.

copyright 2020, QuakeMigrate developers.

license GNU General Public License, Version 3 (https://www.gnu.org/licenses/gpl-3.0.html)

`quakemigrate.io.cut_waveforms.`**`write_cut_waveforms`**(*run*, *event*, *file_format*, *pre_cut=0.0*, *post_cut=0.0*)
Output raw cut waveform data as a waveform file – defaults to mSEED.

Parameters

- **`run`** (`Run` object) – Light class encapsulating i/o path information for a given run.
- **`event`** (`Event` object) – Light class encapsulating signal, onset, and location information for a given event.
- **`file_format`** (*`str, optional`*) – File format to write waveform data to. Options are all file formats supported by obspy, including: "MSEED" (default), "SAC", "SEGY", "GSE2"
- **`pre_cut`** (*`float, optional`*) – Specify how long before the event origin time to cut the waveform data from
- **`post_cut`** (*`float, optional`*) – Specify how long after the event origin time to cut the waveform data to

quakemigrate.io.data

Module for processing waveform files stored in a data archive.

copyright 2020, QuakeMigrate developers.

license GNU General Public License, Version 3 (https://www.gnu.org/licenses/gpl-3.0.html)

`class` `quakemigrate.io.data.`**`Archive`**(*archive_path*, *stations*, *archive_format=None*, ***kwargs*)
Bases: `object`

The Archive class handles the reading of archived waveform data.

It is capable of handling any regular archive structure. Requests to read waveform data are served up as a *quakemigrate.data.WaveformData* object. Data will be checked for availability within the requested time period, and optionally resampled to meet a unified sampling rate. The raw data read from the archive will also be retained.

If provided, a response inventory provided for the archive will be stored with the waveform data for response removal, if needed.

Parameters

- **`archive_path`** (*`str`*) – Location of seismic data archive: e.g.: ./DATA_ARCHIVE.
- **`stations`** (*pandas.DataFrame* object) – Station information. Columns ["Latitude", "Longitude", "Elevation", "Name"]

- **archive_format** (*str, optional*) – Sets path type for different archive formats.
- **kwargs** (**dict) – See Archive Attributes for details.

archive_path
Location of seismic data archive: e.g.: ./DATA_ARCHIVE.

Type *pathlib.Path* object

stations
Series object containing station names.

Type *pandas.Series* object

format
File naming format of data archive.

Type str

read_all_stations
If True, read all stations in archive for that time period. Else, only read specified stations.

Type bool, optional

resample
If true, perform resampling of data which cannot be decimated directly to the desired sampling rate.

Type bool, optional

response_inv
ObsPy response inventory for this waveform archive, containing response information for each channel of each station of each network.

Type *obspy.Inventory* object, optional

upfactor
Factor by which to upsample the data to enable it to be decimated to the desired sampling rate, e.g. 40Hz -> 50Hz requires upfactor = 5.

Type int, optional

path_structure (*path_type="YEAR/JD/STATION"*)
Set the file naming format of the data archive.

read_waveform_data (*starttime*, *endtime*, *sampling_rate*)
Read in all waveform data between two times, decimate / resample if required to reach desired sampling rate. Return all raw data as an obspy Stream object and processed data for specified stations as an array for use by QuakeScan to calculate onset functions for migration.

path_structure (*archive_format='YEAR/JD/STATION'*, *channels='*'*)
Define the path structure of the data archive.

Parameters

- **archive_format** (*str, optional*) – Sets path type for different archive formats.
- **channels** (*str, optional*) – Channel codes to include. E.g. channels="[B,H]H*". (Default "*")

Raises ArchivePathStructureError – If the archive_format specified by the user is not a valid option.

read_waveform_data (*starttime*, *endtime*, *sampling_rate*, *pre_pad=0.0*, *post_pad=0.0*)
Read in the waveform data for all stations in the archive between two times and return station availability

of the stations specified in the station file during this period. Decimate / resample (optional) this data if required to reach desired sampling rate.

Output both processed data for stations in station file and all raw data in an obspy Stream object.

By default, data with mismatched sampling rates will only be decimated. If necessary, and if the user specifies *resample = True* and an upfactor to upsample by *upfactor = int*, data can also be upsampled and then, if necessary, subsequently decimated to achieve the desired sampling rate.

For example, for raw input data sampled at a mix of 40, 50 and 100 Hz, to achieve a unified sampling rate of 50 Hz, the user would have to specify an upfactor of 5; 40 Hz x 5 = 200 Hz, which can then be decimated to 50 Hz.

NOTE: data will be detrended and a cosine taper applied before decimation, in order to avoid edge effects when applying the lowpass filter. Otherwise, data for migration will be added tp data.signal with no processing applied.

Supports all formats currently supported by ObsPy, including: "MSEED" (default), "SAC", "SEGY", "GSE2" .

Parameters

- **starttime** (*obspy.UTCDateTime* object, optional) – Timestamp from which to read waveform data.
- **endtime** (*obspy.UTCDateTime* object, optional) – Timestamp up to which to read waveform data.
- **sampling_rate** (*int*) – Desired sampling rate for data to be added to signal. This will be achieved by resampling the raw waveform data. By default, only decimation will be applied, but data can also be upsampled if specified by the user when creating the Archive object.
- **pre_pad** (*float, optional*) – Additional pre pad of data to cut based on user-defined pre_cut parameter. Defaults to none: pre_pad calculated in QuakeScan will be used (included in starttime).
- **post_pad** (*float, optional*) – Additional post pad of data to cut based on user-defined post_cut parameter. Defaults to none: post_pad calculated in QuakeScan will be used (included in endtime).

Returns **data** – Object containing the archived data that satisfies the query.

Return type *WaveformData* object

class quakemigrate.io.data.**WaveformData**(*starttime*, *endtime*, *sampling_rate*, *stations=None*, *response_inv=None*, *read_all_stations=False*, *pre_pad=0.0*, *post_pad=0.0*)

Bases: object

The WaveformData class encapsulates the waveform data returned by an‘ Archive query.

This includes the waveform data which has been pre-processed to a unified sampling rate, and checked for gaps, ready for use to calculate onset functions.

Parameters

- **starttime** (*obspy.UTCDateTime* object) – Timestamp of first sample of waveform data.
- **endtime** (*obspy.UTCDateTime* object) – Timestamp of last sample of waveform data.
- **sampling_rate** (*int*) – Desired sampling rate of signal data.
- **stations** (*pandas.Series* object, optional) – Series object containing station names.

- **read_all_stations** (*bool, optional*) – If True, raw_waveforms contain all stations in archive for that time period. Else, only selected stations will be included.
- **response_inv** (*obspy.Inventory* object, optional) – ObsPy response inventory for this waveform archive, containing response information for each channel of each station of each network.
- **pre_pad** (*float, optional*) – Additional pre pad of data cut based on user-defined pre_cut parameter.
- **post_pad** (*float, optional*) – Additional post pad of data cut based on user-defined post_cut parameter.

starttime
Timestamp of first sample of waveform data.

Type *obspy.UTCDateTime* object

endtime
Timestamp of last sample of waveform data.

Type *obspy.UTCDateTime* object

sampling_rate
Sampling rate of signal data.

Type int

stations
Series object containing station names.

Type *pandas.Series* object

read_all_stations
If True, raw_waveforms contain all stations in archive for that time period. Else, only selected stations will be included.

Type bool

raw_waveforms
Raw seismic data found and read in from the archive within the specified time period. This may be for all stations in the archive, or only those specified by the user. See *read_all_stations*.

Type *obspy.Stream* object

pre_pad
Additional pre pad of data cut based on user-defined pre_cut parameter.

Type float

post_pad
Additional post pad of data cut based on user-defined post_cut parameter.

Type float

signal
3-component seismic data at the desired sampling rate; only for desired stations, which have continuous data on all 3 components throughout the desired time period and where (if necessary) the data could be successfully resampled to the desired sampling rate.

Type *numpy.ndarray*, shape(3, nstations, nsamples)

availability
Array containing 0s (no data) or 1s (data), corresponding to whether data for each station met the requirements outlined in *signal*

Type *np.ndarray* of ints, shape(nstations)

filtered_signal
Filtered data originally from signal.

Type *numpy.ndarray*, shape(3, nstations, nsamples)

add_stream(*stream*, *resample*, *upfactor*)
Function to add data supplied in the form of an *obspy.Stream* object.

get_wa_waveform(*trace*, ***response_removal_params*)
Calculate the Wood-Anderson corrected waveform for a *obspy.Trace* object.

times()
Utility function to generate the corresponding timestamps for the waveform and coalescence data.

Raises `NotImplementedError` – If the user attempts to use the get_real_waveform() method.

add_stream(*stream*, *resample*, *upfactor*)
Add signal data supplied in an *obspy.Stream* object. Perform resampling if necessary (decimation and/or upsampling), and determine availability of selected stations.

stream [*obspy.Stream* object] Contains list of *obspy.Trace* objects containing the waveform data to add.

resample [bool, optional] If true, perform resampling of data which cannot be decimated directly to the desired sampling rate.

upfactor [int, optional] Factor by which to upsample the data to enable it to be decimated to the desired sampling rate, e.g. 40Hz -> 50Hz requires upfactor = 5.

get_real_waveforms(*tr*, *remove_full_response=False*, *velocity=True*)
Coming soon.

get_wa_waveform(*tr*, *water_level*, *pre_filt*, *remove_full_response=False*, *velocity=False*)
Calculate simulated Wood Anderson displacement waveform for a Trace.

Parameters

- **tr** (*obspy.Trace* object) – Trace containing the waveform to be corrected to a Wood-Anderson response
- **water_level** (*float*) – Water-level to be used in the instrument correction.
- **pre_filt** (*tuple of floats, or None*) – Filter corners describing filter to be applied to the trace before deconvolution. E.g. (0.05, 0.06, 30, 35) (in Hz)
- **remove_full_response** (*bool, optional*) – Remove all response stages, inc. FIR (st.remove_response()), not just poles-and-zero response stage. Default: False.
- **velocity** (*bool, optional*) – Output velocity waveform, instead of displacement. Default: False.

Returns **tr** – Trace corrected to Wood-Anderson response.

Return type *obspy.Trace* object

Raises

- `AttributeError` – If no response inventory has been supplied.
- `ResponseNotFoundError` – If the response information for a trace can't be found in the supplied response inventory.
- `ResponseRemovalError` – If the deconvolution of the instrument response and simulation of the Wood-Anderson response is unsuccessful.

- `NotImplementedError` – If the user selects velocity=True.

sample_size
s).

Type Get the size of a sample (units

times(***kwargs*)
Utility function to generate timestamps between *data.starttime* and *data.endtime*, with a sample size of *data.sample_size*

Returns **times** – Timestamps for the timeseries data.

Return type *numpy.ndarray*, shape(nsamples)

quakemigrate.io.scanmseed

Module to handle input/output of .scanmseed files.

copyright 2020, QuakeMigrate developers.

license GNU General Public License, Version 3 (https://www.gnu.org/licenses/gpl-3.0.html)

class quakemigrate.io.scanmseed.**ScanmSEED**(*run*, *continuous_write*, *sampling_rate*)
Bases: `object`

Light class to encapsulate the data output by the detect stage of QuakeMigrate. This data is stored in an *obspy.Stream* object with the channels: ["COA", "COA_N", "X", "Y", "Z"].

Parameters

- **run** (`Run` object) – Light class encapsulating i/o path information for a given run.
- **continuous_write** (*bool*) – Option to continuously write the .scanmseed file output by detect() at the end of every time step. Default behaviour is to write in day chunks where possible.
- **sampling_rate** (*int*) – Desired sampling rate of input data; sampling rate at which to compute the coalescence function. Default: 50 Hz.

stream
Output of detect() stored in *obspy.Stream* object. The values have been multiplied by a factor to make use of more efficient compression. Channels: ["COA", "COA_N", "X", "Y", "Z"]

Type *obspy.Stream* object

written
Tracker for whether the data appended has been written recently.

Type bool

append(*times*, *max_coa*, *max_coa_n*, *coord*, *map4d=None*)
Append the output of QuakeScan._compute() to the coalescence stream.

empty(*starttime*, *timestep*, *i*, *msg*)
Create an set of empty arrays for a given timestep and append to the coalescence stream.

write(*write_start=None*, *write_end=None*)
Write the coalescence stream to a .scanmseed file.

append(*starttime*, *max_coa*, *max_coa_n*, *coord*, *ucf*)
Append latest timestep of detect() output to *obspy.Stream* object. Multiply channels ["COA", "COA_N", "X", "Y", "Z"] by factors of ["1e5", "1e5", "1e6", "1e6", "1e3"] respectively, round and convert to int32

as this dramatically reduces memory usage, and allows the coastream data to be saved in mSEED format with STEIM2 compression. The multiplication factor is removed when the data is read back in.

Parameters

- **starttime** (*obspy.UTCDateTime* object) – Timestamp of first sample of coalescence data.
- **max_coa** (*numpy.ndarray* of floats, shape(nsamples)) – Coalescence value through time.
- **max_coa_n** (*numpy.ndarray* of floats, shape(nsamples)) – Normalised coalescence value through time.
- **coord** (*numpy.ndarray* of floats, shape(nsamples)) – Location of maximum coalescence through time in input projection space.
- **ucf** (*float*) – A conversion factor based on the lookup table grid projection. Used to ensure the same level of precision (millimetre) is retained during compression, irrespective of the units of the grid projection.

empty(*starttime*, *timestep*, *i*, *msg*, *ucf*)

Create an empty set of arrays to write to .scanmseed; used where there is no data available to run _compute().

Parameters

- **starttime** (*obspy.UTCDateTime* object) – Timestamp of first sample in the given timestep.
- **timestep** (*float*) – Length (in seconds) of timestep used in detect().
- **i** (*int*) – The ith timestep of the continuous compute.
- **msg** (*str*) – Message to output to log giving details as to why this timestep is empty.
- **ucf** (*float*) – A conversion factor based on the lookup table grid projection. Used to ensure the same level of precision (millimetre) is retained during compression, irrespective of the units of the grid projection.

write(*write_start=None*, *write_end=None*)

Write a new .scanmseed file from an *obspy.Stream* object containing the data output from detect(). Note: values have been multiplied by a power of ten, rounded and converted to an int32 array so the data can be saved as mSEED with STEIM2 compression. This multiplication factor is removed when the data is read back in with read_scanmseed().

Parameters

- **write_start** (*obspy.UTCDateTime* object, optional) – Timestamp from which to write the coalescence stream to file.
- **write_end** (*obspy.UTCDateTime* object, optional) – Timestamp up to which to write the coalescence stream to file.

quakemigrate.io.scanmseed.**read_scanmseed**(*run*, *starttime*, *endtime*, *pad*, *ucf*)

Read .scanmseed files between two time stamps. Files are labelled by year and Julian day.

Parameters

- **run** (Run object) – Light class encapsulating i/o path information for a given run.
- **starttime** (*obspy.UTCDateTime* object) – Timestamp from which to read the coalescence stream.
- **endtime** (*obspy.UTCDateTime* object) – Timestamp up to which to read the coalescence stream.

- **pad** (*float*) – Read in "pad" seconds of additional data on either end.
- **ucf** (*float*) – A conversion factor based on the lookup table grid projection. Used to ensure the same level of precision (millimetre) is retained during compression, irrespective of the units of the grid projection.

Returns

- **data** (*pandas.DataFrame* object) – Data output by detect() – decimated scan. Columns: ["DT", "COA", "COA_N", "X", "Y", "Z"] - X/Y/Z as lon/lat/m
- **stats** (*obspy.trace.Stats* object) – Container for additional header information for coalescence trace. Contains keys: network, station, channel, starttime, endtime,

  sampling_rate, delta, npts, calib, _format, mseed

quakemigrate.io.triggered_events

Module to handle input/output of TriggeredEvents.csv files.

copyright 2020, QuakeMigrate developers.

license GNU General Public License, Version 3 (https://www.gnu.org/licenses/gpl-3.0.html)

quakemigrate.io.triggered_events.**read_triggered_events**(*run*, ***kwargs*)
Read triggered events from .csv file.

Parameters

- **run** (Run object) – Light class encapsulating i/o path information for a given run.
- **starttime** (*obspy.UTCDateTime* object, optional) – Timestamp from which to include events in the locate scan.
- **endtime** (*obspy.UTCDateTime* object, optional) – Timestamp up to which to include events in the locate scan.
- **trigger_file** (*str, optional*) – File containing triggered events to be located.

Returns **events** – Triggered events information. Columns: ["EventID", "CoaTime", "TRIG_COA", "COA_X", "COA_Y", "COA_Z", "COA", "COA_NORM"].

Return type *pandas.DataFrame* object

quakemigrate.io.triggered_events.**write_triggered_events**(*run*, *events*, *starttime*)
Write triggered events to a .csv file.

Parameters

- **run** (Run object) – Light class encapsulating i/o path information for a given run.
- **events** (*pandas.DataFrame* object) – Triggered events information. Columns: ["EventID", "CoaTime", "TRIG_COA", "COA_X", "COA_Y", "COA_Z", "COA", "COA_NORM"].
- **starttime** (*obspy.UTCDateTime* object) – Timestamp from which events have been triggered.

5.3.4 quakemigrate.lut package

The *quakemigrate.lut* module handles the definition and generation of the traveltime lookup tables used in QuakeMigrate.

quakemigrate.lut.**update_lut**(*old_lut_file*, *save_file*)

Utility function to convert old-style LUTs to new-style LUTs.

Parameters

- **old_lut_file** (*str*) – Path to lookup table file to update.
- **save_file** (*str, optional*) – Output path for updated lookup table.

quakemigrate.lut.create_lut

Module to produce traveltime lookup tables defined on a Cartesian grid.

quakemigrate.lut.create_lut.**compute_traveltimes**(*grid_spec*, *stations*, *method*, *phases=['P', 'S']*, *fraction_tt=0.1*, *save_file=None*, *log=False*, ***kwargs*)

Top-level method for computing traveltime lookup tables.

This function takes a grid specification and is capable of computing traveltimes for an arbitrary number of phases using a variety of techniques.

Parameters

- **grid_spec** (*dict*) – Dictionary containing all of the defining parameters for the underlying 3-D grid on which the traveltimes are to be calculated. For expected keys, see *Grid3D*.
- **stations** (*pandas.DataFrame*) – DataFrame containing station information (lat/lon/elev).
- **method** (*str*) –

  Method to be used when computing the traveltime lookup tables. "homogeneous" - straight line velocities "1dfmm" - 1-D fast-marching method using scikit-fmm "1dsweep" - a 2-D traveltime grid for a 1-D velocity model is swept

  over the 3-D grid using a bilinear interpolation scheme
- **phases** (*list of str, optional*) – List of seismic phases for which to calculate traveltimes.
- **fraction_tt** (*float, optional*) – An estimate of the uncertainty in the velocity model as a function of a fraction of the traveltime. (Default 0.1 == 10%)
- **filename** (*str, optional*) – Path to location to save pickled lookup table.
- **log** (*bool, optional*) – Toggle for logging - default is to only print information to stdout. If True, will also create a log file.
- **kwargs** (*dict*) – Dictionary of all keyword arguments passed to compute when called. For lists of valid arguments, please refer to the relevant method.

Returns **lut** – Lookup table populated with traveltimes from the NonLinLoc files.

Return type LUT object

quakemigrate.lut.create_lut.**read_nlloc**(*path, stations, phases=['P', 'S'], fraction_tt=0.1, log=False*)

Read in a traveltime lookup table that is saved in the NonLinLoc format.

Parameters

- **path** (*str*) – Path to directory containing .buf and .hdr files.
- **stations** (*pandas.DataFrame*) – DataFrame containing station information (lat/lon/elev).
- **phases** (*list of str, optional*) – List of seismic phases for which to calculate traveltimes.
- **fraction_tt** (*float, optional*) – An estimate of the uncertainty in the velocity model as a function of a fraction of the traveltime. (Default 0.1 == 10%)
- **log** (*bool, optional*) – Toggle for logging - default is to only print information to stdout. If True, will also create a log file.

Returns **lut** – Lookup table populated with traveltimes from the NonLinLoc files.

Return type LUT object

quakemigrate.lut.lut

Module to produce traveltime lookup tables defined on a Cartesian grid.

copyright 2020, QuakeMigrate developers.

license GNU General Public License, Version 3 (https://www.gnu.org/licenses/gpl-3.0.html)

class quakemigrate.lut.lut.**Grid3D**(*ll_corner*, *ur_corner*, *node_spacing*, *grid_proj*, *coord_proj*)

Bases: object

A grid object represents a collection of points in a 3-D Cartesian space that can be used to produce regularised traveltime lookup tables that sample the continuous traveltime space for each station in a seismic network.

This class also provides the series of transformations required to move between the input projection, the grid projection and the grid index coordinate spaces.

The size and shape specifications of the grid are defined by providing the (input projection) coordinates for the lower-left and upper-right corners, a node spacing and the projections (defined using pyproj) of the input and grid spaces.

coord_proj
Input coordinate space projection.

Type *pyproj.Proj* object

grid_corners
Positions of the corners of the grid in the grid coordinate space.

Type array-like, shape (8, 3)

grid_proj
Grid space projection.

Type *pyproj.Proj* object

grid_xyz
Positions of the grid nodes in the grid coordinate space. The shape of each element of the list is defined by the number of nodes in each dimension.

Type array-like, shape (3, nx, ny, nz)

ll_corner
Location of the lower-left corner of the grid in the grid projection. Should also contain the minimum depth in the grid.

Type array-like, [float, float, float]

node_count
Number of nodes in each dimension of the grid. This is calculated by finding the number of nodes with a given node spacing that fit between the lower-left and upper-right corners. This value is rounded up if the number of nodes returned is non-integer, to ensure the requested area is included in the grid.

Type array-like, [int, int, int]

node_spacing
Distance between nodes in each dimension of the grid.

Type array-like, [float, float, float]

precision
An appropriate number of decimal places for distances as a function of the node spacing and coordinate projection.

Type list of float

unit_conversion_factor
A conversion factor based on the grid projection, used to convert between units of metres and kilometres.

Type float

unit_name
Shorthand string for the units of the grid projection.

Type str

ur_corner
Location of the upper-right corner of the grid in the grid projection. Should also contain the maximum depth in the grid.

Type array-like, [float, float, float]

coord2grid(*value*, *inverse=False*, *clip=False*)
Provides a transformation between the input projection and grid coordinate spaces.

decimate(*df*, *inplace=False*)
Downsamples the traveltime lookup tables by some decimation factor.

index2coord(*value*, *inverse=False*, *unravel=False*, *clip=False*)
Provides a transformation between grid indices (can be a flattened index or an [i, j, k] position) and the input projection coordinate space.

index2grid(*value*, *inverse=False*, *unravel=False*)
Provides a transformation between grid indices (can be a flattened index or an [i, j, k] position) and the grid coordinate space.

cell_count
Handler for deprecated attribute name 'cell_count'

cell_size
Handler for deprecated attribute name 'cell_size'

coord2grid(*value*, *inverse=False*)
Convert between input coordinate space and grid coordinate space.

Parameters

- **value** (*array-like*) – Array (of arrays) containing the coordinate locations to be transformed. Each sub-array should describe a single point in the 3-D input space.
- **inverse** (*bool, optional*) – Reverses the direction of the transform. Default input coordinates -> grid coordinates

Returns **out** – Returns an array of arrays of the transformed values.

Return type array-like

decimate(*df*, *inplace=False*)

Resample the traveltime lookup tables by decimation by some factor.

Parameters

- **df** (*array-like [int, int, int]*) – Decimation factor in each dimension.
- **inplace** (*bool, optional*) – Perform the operation on the lookup table object or a copy.

Returns **grid** – Returns a Grid3D object with decimated traveltime lookup tables.

Return type Grid3D object (optional)

get_grid_extent(*cells=False*)

Get the minimum/maximum extent of each dimension of the grid.

The default returns the grid extent as the convex hull of the grid nodes. It is useful, for visualisation purposes, to also be able to determine the grid extent as the convex hull of a grid of cells centred on the grid nodes.

Parameters **cells** (*bool, optional*) – Specifies the grid mode (nodes / cells) for which to calculate the extent.

Returns **extent** – Pair of arrays representing two corners for the grid.

Return type array-like

grid_corners

Get the xyz positions of the nodes on the corners of the grid.

grid_extent

Get the minimum/maximum extent of each dimension of the grid.

The default returns the grid extent as the convex hull of the grid nodes. It is useful, for visualisation purposes, to also be able to determine the grid extent as the convex hull of a grid of cells centred on the grid nodes.

Parameters **cells** (*bool, optional*) – Specifies the grid mode (nodes / cells) for which to calculate the extent.

Returns **extent** – Pair of arrays representing two corners for the grid.

Return type array-like

grid_xyz

Get the xyz positions of all of the nodes in the grid.

index2coord(*value*, *inverse=False*, *unravel=False*)

Convert between grid indices and input coordinate space.

This is a utility function that wraps the other two defined transforms.

Parameters

- **value** (*array-like*) – Array (of arrays) containing the grid indices (grid coordinates) to be transformed. Can be an array of flattened indices.
- **inverse** (*bool, optional*) – Reverses the direction of the transform. Default indices -> input projection coordinates.
- **unravel** (*bool, optional*) – Convert a flat index or array of flat indices into a tuple of coordinate arrays.

Returns **out** – Returns an array of arrays of the transformed values.

Return type array-like

index2grid(*value*, *inverse=False*, *unravel=False*)
Convert between grid indices and grid coordinate space.

Parameters

- **value** (*array-like*) – Array (of arrays) containing the grid indices (grid coordinates) to be transformed. Can be an array of flattened indices.
- **inverse** (*bool, optionale*) – Reverses the direction of the transform. Default indices -> grid coordinates.
- **unravel** (*bool, optional*) – Convert a flat index or array of flat indices into a tuple of coordinate arrays.

Returns **out** – Returns an array of arrays of the transformed values.

Return type array-like

node_count
Get and set the number of nodes in each dimension of the grid.

node_spacing
Get and set the spacing of nodes in each dimension of the grid.

precision
Get appropriate number of decimal places as a function of the node spacing and coordinate projection.

unit_conversion_factor
Expose unit_conversion_factor of the grid projection.

unit_name
Expose unit_name of the grid_projection and return shorthand.

class quakemigrate.lut.lut.**LUT**(*fraction_tt=0.1*, *lut_file=None*, ***grid_spec*)
Bases: *quakemigrate.lut.lut.Grid3D*

A lookup table (LUT) object is a simple data structure that is used to store a series of regularised tables that, for each seismic station in a network, store the traveltimes to every point in the 3-D volume. These lookup tables are pre-computed to reduce the computational cost of the back-projection method.

This class provides utility functions that can be used to serve up or query these pre-computed lookup tables.

This object is-a Grid3D.

fraction_tt
An estimate of the uncertainty in the velocity model as a function of a fraction of the traveltime. (Default 0.1 == 10%)

Type float

max_traveltime
The maximum traveltime between any station and a point in the grid.

Type float

phases
Seismic phases for which there are traveltime lookup tables available.

Type list of str

stations_xyz
Positions of the stations in the grid coordinate space.

Type array-like, shape (n, 3)

traveltimes
A dictionary containing the traveltime lookup tables. The structure of this dictionary is:

traveltimes

- **"<Station1-ID>"**
  - "<PHASE>"
  - "<PHASE>"
- **"<Station2-ID"**
  - "<PHASE>"
  - "<PHASE>"

etc

Type dict

velocity_model
Contains the input velocity model specification.

Type *~pandas.DataFrame* object

serve_traveltimes(*sampling_rate*)
Serve up the traveltime lookup tables.

traveltime_to(*phase*, *ijk*)
Query traveltimes to a grid location (in terms of indices) for a particular phase.

save(*filename*)
Dumps the current state of the lookup table object to a pickle file.

load(*filename*)
Restore the state of the saved LUT object from a pickle file.

plot(*fig*, *gs*, *slices=None*, *hypocentre=None*, *station_clr="k"*)
Plot cross-sections of the LUT with station locations. Optionally plot slices through a coalescence volume.

load(*filename*)
Read the contents of a pickle file and restore state of the lookup table object.

Parameters **filename** (*str*) – Path to pickle file to load.

max_extent
Get the minimum/maximum geographical extent of the stations/grid.

max_traveltime
Get the maximum traveltime from any station across the grid.

plot(*fig*, *gs*, *slices=None*, *hypocentre=None*, *station_clr='k'*)
Plot the lookup table for a particular station.

Parameters

- **fig** (*~matplotlib.Figure* object) – Canvas on which LUT is plotted.
- **gs** (*tuple(int, int)*) – Grid specification for the plot.
- **slices** (*array of arrays, optional*) – Slices through a coalescence volume to plot.
- **hypocentre** (*array of floats*) – Event hypocentre - will add cross-hair to plot.
- **station_clr** (*str, optional*) – Plot the stations with a particular colour.

save (*filename*)
Dump the current state of the lookup table object to a pickle file.

Parameters **filename** (*str*) – Path to location to save pickled lookup table.

serve_traveltimes (*sampling_rate*)
Serve up the traveltime lookup tables.

The traveltimes are multiplied by the scan sampling rate and converted to integers.

Parameters **sampling_rate** (*int*) – Samples per second used in the scan run.

Returns **traveltimes** – Stacked traveltime lookup tables for all seismic phases, stacked along the station axis, with shape(nx, ny, nz, nstations)

Return type *numpy.ndarray* of *numpy.int*

station_extent
Get the minimum/maximum extent of the seismic network.

stations_xyz
Get station locations in the grid space [X, Y, Z].

traveltime_to (*phase*, *ijk*)
Serve up the traveltimes to a grid location for a particular phase.

Parameters

- **phase** (*str*) – The seismic phase to lookup.
- **ijk** (*array-like*) – Grid indices for which to serve traveltime.

Returns **traveltimes** – Array of interpolated traveltimes to the requested grid position.

Return type array-like

5.3.5 quakemigrate.plot

The *quakemigrate.plot* module provides methods for the generation of figures in QuakeMigrate, including:

- Event summaries
- Phase pick summaries
- Triggered event summaries

quakemigrate.plot.event

Module containing methods to generate event summaries and videos.

copyright 2020, QuakeMigrate developers.

license GNU General Public License, Version 3 (https://www.gnu.org/licenses/gpl-3.0.html)

quakemigrate.plot.event.**event_summary**(*run*, *event*, *marginal_coalescence*, *lut*, *xy_files=None*)

Plots an event summary illustrating the locate results: slices through the marginalised coalescence with the location estimates (best-fitting spline to interpolated coalescence; Gaussian fit; covariance fit) and associated uncertainties; a gather of the filtered station data, sorted by distance from the event; and the maximum coalescence through time.

Parameters

- **run** (Run object) – Light class encapsulating i/o path information for a given run.
- **event** (Event object) – Light class encapsulating signal, onset, and location information for a given event.
- **marginal_coalescence** (*~numpy.ndarray* of *~numpy.double*) – Marginalised 3-D coalescence map, shape(nx, ny, nz).
- **lut** (LUT object) – Contains the traveltime lookup tables for seismic phases, computed for some pre-defined velocity model.
- **xy_files** (*str, optional*) – Path to comma-separated value file (.csv) containing a series of coordinate files to plot. Columns: ["File", "Color", "Linewidth", "Linestyle"], where "File" is the absolute path to the file containing the coordinates to be plotted. E.g: "/home/user/volcano_outlines.csv,black,0.5,-". Each .csv coordinate file should contain coordinates only, with columns: ["Longitude", "Latitude"]. E.g.: "-17.5,64.8". Lines pre-pended with # will be treated as a comment - this can be used to include references. See the Volcanotectonic_Iceland example XY_files for a template.

Note: Do not include a header line in either file.

---

quakemigrate.plot.phase_picks

Module to produce a summary plot for the phase picking.

copyright 2020, QuakeMigrate developers.

license GNU General Public License, Version 3 (https://www.gnu.org/licenses/gpl-3.0.html)

quakemigrate.plot.phase_picks.**pick_summary**(*event*, *station*, *signal*, *picks*, *onsets*, *ttimes*, *window*)

Plot figure showing the filtered traces for each data component and the characteristic functions calculated from them (P and S) for each station. The search window to make a phase pick is displayed, along with the dynamic pick threshold (defined as a percentile of the background noise level), the phase pick time and its uncertainty (if made) and the Gaussian fit to the characteristic function.

Parameters

- **event** (*str*) – Unique identifier for the event.
- **station** (*str*) – Station code.
- **signal** (*numpy.ndarray* of int) – Seismic data for the Z N and E components.

- **picks** (*pandas DataFrame object*) – Phase pick times with columns ["Name", "Phase", "ModelledTime", "PickTime", "PickError", "SNR"] Each row contains the phase pick from one station/phase.
- **onsets** (*numpy.ndarray* of float) – Onset functions for each seismic phase, shape(nstations, nsamples).
- **ttimes** (*list, [int, int]*) – Modelled phase travel times.
- **window** (*list, [int, int]*) – Indices specifying the window within which the pick was made.

Returns **fig** – Figure showing basic phase picking information.

Return type *matplotlib.Figure* object

quakemigrate.plot.trigger

Module to plot the triggered events on a decimated grid.

copyright 2020, QuakeMigrate developers.

license GNU General Public License, Version 3 (https://www.gnu.org/licenses/gpl-3.0.html)

quakemigrate.plot.trigger.**trigger_summary**(*events*, *starttime*, *endtime*, *run*, *marginal_window*, *min_event_interval*, *detection_threshold*, *normalise_coalescence*, *lut*, *data*, *region*, *savefig*, *discarded_events*, *xy_files=None*)

Plots the data from a .scanmseed file with annotations illustrating the trigger results: event triggers and marginal windows on the coalescence traces, and map and cross section view of the gridded triggered earthquake locations.

Parameters

- **events** (*pandas.DataFrame*) – Triggered events information, columns: ["EventID", "CoaTime", "TRIG_COA", "COA_X", "COA_Y", "COA_Z", "MinTime", "MaxTime", "COA", "COA_NORM"].
- **starttime** (*obspy.UTCDateTime*) – Start time of trigger run.
- **endtime** (*obspy.UTCDateTime*) – End time of trigger run.
- **run** (Run object) – Light class encapsulating i/o path information for a given run.
- **marginal_window** (*float*) – Estimate of time error over which to marginalise the coalescence.
- **min_event_interval** (*float*) – Minimum time interval between triggered events.
- **detection_threshold** (*array-like*) – Coalescence value above which to trigger events.
- **normalise_coalescence** (*bool*) – If True, use coalescence normalised by the average background noise.
- **lut** (LUT object) – Contains the traveltime lookup tables for P- and S-phases, computed for some pre-defined velocity model.
- **data** (*pandas.DataFrame*) – Data output by detect() – decimated scan, columns ["COA", "COA_N", "X", "Y", "Z"]
- **region** (*list*) – Geographical region within which earthquakes have been triggered.

- **savefig** (*bool*) – Output the plot as a file. The plot is shown by default, and not saved.
- **discarded_events** (*pandas.DataFrame*) – Discarded triggered events information, columns: ["EventID", "CoaTime", "TRIG_COA", "COA_X", "COA_Y", "COA_Z", "MinTime", "MaxTime", "COA", "COA_NORM"].
- **xy_files** (*str, optional*) – Path to comma-separated value file (.csv) containing a series of coordinate files to plot. Columns: ["File", "Color", "Linewidth", "Linestyle"], where "File" is the absolute path to the file containing the coordinates to be plotted. E.g: "/home/user/volcano_outlines.csv,black,0.5,-". Each .csv coordinate file should contain coordinates only, with columns: ["Longitude", "Latitude"]. E.g.: "-17.5,64.8". Lines pre-pended with # will be treated as a comment - this can be used to include references. See the Volcanotectonic_Iceland example XY_files for a template.

Note: Do not include a header line in either file.

5.3.6 quakemigrate.signal

The *quakemigrate.signal* module handles the core of the QuakeMigrate methods. This includes:

- Generation of onset functions from raw data.
- Picking of waveforms from onset functions.
- Raw scan for detect and locate.
- Measurement of amplitudes and calculation of local earthquake magnitudes.

Subpackages

quakemigrate.signal.onsets

The `quakemigrate.onsets` module handles the generation of Onset functions. The default method uses the ratio between the short-term and long-term averages.

Feel free to contribute more Onset function options!

quakemigrate.signal.onsets.base

A simple abstract base class with method stubs to enable users to extend QuakeMigrate with custom onset functions that remain compatible with the core of the package.

class quakemigrate.signal.onsets.base.**Onset**(***kwargs*)
Bases: abc.ABC

QuakeMigrate default onset function class.

sampling_rate
Desired sampling rate for input data; sampling rate at which the onset functions will be computed.

Type int

pre_pad
Option to override the default pre-pad duration of data to read before computing 4-D coalescence in detect() and locate().

Type float, optional

post_pad
Option to override the default post-pad duration of data to read before computing 4-D coalescence in detect() and locate().

Type float

calculate_onsets()
Generate onset functions that represent seismic phase arrivals

calculate_onsets()
Method stub for calculation of onset functions.

gaussian_halfwidth(*phase*)
Method stub for Gaussian half-width estimate.

pad(*timespan*)
Determine the number of samples needed to pre- and post-pad the timespan.

Parameters **timespan** (*float*) – The time window to pad.

Returns

- **pre_pad** (*float*) – Option to override the default pre-pad duration of data to read before computing 4-D coalescence in detect() and locate().
- **post_pad** (*float*) – Option to override the default post-pad duration of data to read before computing 4-D coalescence in detect() and locate().

post_pad
Get property stub for pre_pad.

pre_pad
Get property stub for pre_pad.

quakemigrate.signal.onsets.stalta

The default onset function class - performs some pre-processing on raw seismic data and calculates STA/LTA onset (characteristic) function.

copyright 2020, QuakeMigrate developers.

license GNU General Public License, Version 3 (https://www.gnu.org/licenses/gpl-3.0.html)

class quakemigrate.signal.onsets.stalta.**CentredSTALTAOnset**(***kwargs*)
Bases: *quakemigrate.signal.onsets.stalta.STALTAOnset*

QuakeMigrate default onset function class - uses a centred STA/LTA onset.

NOTE: THIS CLASS HAS BEEN DEPRECATED AND WILL BE REMOVED IN A FUTURE UPDATE

class quakemigrate.signal.onsets.stalta.**ClassicSTALTAOnset**(***kwargs*)
Bases: *quakemigrate.signal.onsets.stalta.STALTAOnset*

QuakeMigrate default onset function class - uses a classic STA/LTA onset.

NOTE: THIS CLASS HAS BEEN DEPRECATED AND WILL BE REMOVED IN A FUTURE UPDATE

class quakemigrate.signal.onsets.stalta.**STALTAOnset**(***kwargs*)
Bases: *quakemigrate.signal.onsets.base.Onset*

QuakeMigrate default onset function class - uses a classic STA/LTA onset.

p_bp_filter
Butterworth bandpass filter specification [lowpass (Hz), highpass (Hz), corners*] *NOTE: two-pass filter effectively doubles the number of corners.

Type array-like, [float, float, int]

s_bp_filter
Butterworth bandpass filter specification [lowpass (Hz), highpass (Hz), corners*] *NOTE: two-pass filter effectively doubles the number of corners.

Type array-like, [float, float, int]

p_onset_win
P onset window parameters [STA, LTA] (both in seconds)

Type array-like, [float, float]

s_onset_win
S onset window parameters [STA, LTA] (both in seconds)

Type array-like, [float, float]

sampling_rate
Desired sampling rate for input data, in Hz; sampling rate at which the onset functions will be computed.

Type int

pre_pad
Option to override the default pre-pad duration of data to read before computing 4-D coalescence in detect() and locate(). Default value is calculated from the onset function parameters.

Type float, optional

position
Compute centred STA/LTA (STA window is preceded by LTA window; value is assigned to end of LTA window / start of STA window) or classic STA/LTA (STA window is within LTA window; value is assigned to end of STA & LTA windows). Default: "classic".

Centred gives less phase-shifted (late) onset function, and is closer to a Gaussian approximation, but is far more sensitive to data with sharp offsets due to instrument failures. We recommend using classic for detect() and centred for locate() if your data quality allows it. This is the default behaviour; override by setting this variable.

Type str, optional

calculate_onsets()
Generate onset functions that represent seismic phase arrivals

calculate_onsets(*data*, *log=True*, *run=None*)
Returns a stacked pair of onset (characteristic) functions for the P and S phase arrivals.

Parameters

- **data** (SignalData object) – Light class encapsulating data returned by an archive query.
- **log** (*bool*) – Calculate log(onset) if True, otherwise calculate the raw onset.
- **run** –

gaussian_halfwidth(*phase*)
Return the phase-appropriate Gaussian half-width estimate based on the short-term average window length.

Parameters **phase** (*{'P', 'S'}*) – Seismic phase for which to serve the estimate.

onset_centred
Handle deprecated onset_centred kwarg / attribute

post_pad
Post-pad is determined as a function of the max traveltime in the grid and the onset windows

pre_pad
Pre-pad is determined as a function of the onset windows

quakemigrate.signal.onsets.stalta.**pre_process**(*sig*, *sampling_rate*, *lc*, *hc*, *order=2*)
Detrend raw seismic data and apply cosine taper and zero phase-shift Butterworth band-pass filter.

Parameters

- **sig** (*array-like*) – Data signal to be pre-processed.
- **sampling_rate** (*int*) – Number of samples per second, in Hz.
- **lc** (*float*) – Lowpass frequency of band-pass filter, in Hz.
- **hc** (*float*) – Highpass frequency of band-pass filter, in Hz.
- **order** (*int, optional*) – Number of filter corners. NOTE: two-pass filter effectively doubles the number of corners.

Returns **fsig** – Filtered seismic data.

Return type array-like

Raises NyquistException – If the high-cut filter specified for the bandpass filter is higher than the Nyquist frequency of the *Waveform.signal* data.

quakemigrate.signal.onsets.stalta.**sta_lta_centred**(*a*, *nsta*, *nlta*)
Calculates the ratio of the average signal in a short-term (signal) window to a preceding long-term (noise) window. STA/LTA value is assigned to the end of the LTA / start of the STA.

Parameters

- **a** (*array-like*) – Signal array
- **nsta** (*int*) – Number of samples in short-term window
- **nlta** (*int*) – Number of samples in long-term window

Returns **sta / lta** – Ratio of short term average window to a preceding long term average window. STA/LTA value is assigned to end of LTA window / start of STA window – "centred"

Return type array-like

quakemigrate.signal.onsets.stalta.**sta_lta_onset**(*fsig*, *stw*, *ltw*, *position*, *log*)
Calculate STA/LTA onset (characteristic) function from pre-processed seismic data.

Parameters

- **fsig** (*array-like*) – Filtered (pre-processed) data signal to be used to generate an onset function.
- **stw** (*int*) – Short term window length (# of samples).
- **ltw** (*int*) – Long term window length (# of samples)
- **position** (*str*) –

  "centred": Compute centred STA/LTA (STA window is preceded by LTA window; value is assigned to end of LTA window / start of STA window) or:

  "classic": classic STA/LTA (STA window is within LTA window; value is assigned to end of STA & LTA windows).

  Centred gives less phase-shifted (late) onset function, and is closer to a Gaussian approximation, but is far more sensitive to data with sharp offsets due to instrument failures. We recommend using classic for detect() and centred for locate() if your data quality allows it. This is the default behaviour; override by setting self.onset_centred.
- **log** (*bool*) – Will return log(onset) if True, otherwise it will return the raw onset.

Returns **onset** – onset_raw or log(onset_raw); both are clipped between 0.8 and infinity.

Return type array-like

quakemigrate.signal.pickers

The `quakemigrate.pickers` module handles the picking of seismic phases. The default method makes the phase picks by fitting a 1-D Gaussian to the Onset function.

Feel free to contribute more phase picking methods!

copyright 2020, QuakeMigrate developers.

license GNU General Public License, Version 3 (https://www.gnu.org/licenses/gpl-3.0.html)

quakemigrate.signal.pickers.base

A simple abstract base class with method stubs enabling simple modification of QuakeMigrate to use custom phase picking methods that remain compatible with the core of the package.

copyright 2020, QuakeMigrate developers.

license GNU General Public License, Version 3 (https://www.gnu.org/licenses/gpl-3.0.html)

class quakemigrate.signal.pickers.base.**PhasePicker**(***kwargs*)
Bases: abc.ABC

Abstract base class providing a simple way of modifying the default picking function in QuakeMigrate.

plot_picks
Toggle plotting of phase picks.

Type bool

pick_phases()
Abstract method stub providing interface with QuakeMigrate scan.

write(*event_uid*, *phase_picks*, *output*)
Outputs phase picks to file.

plot()
Method stub for phase pick plotting.

pick_phases()
Method stub for phase picking.

plot()
Method stub for phase pick plotting.

write(*run*, *event_uid*, *phase_picks*)
Write phase picks to a new .picks file.

Parameters

- **event_uid** (*str*) – Unique identifier for the event.
- **phase_picks** (*pandas DataFrame object*) –

  Phase pick times with columns: ["Name", "Phase", "ModelledTime", "PickTime", "PickError", "SNR"]

  Each row contains the phase pick from one station/phase.
- **output** (*QuakeMigrate input/output control object*) – Contains useful methods controlling output for the scan.

quakemigrate.signal.pickers.gaussian

The default seismic phase picking class - fits a 1-D Gaussian to the calculated onset functions.

copyright 2020, QuakeMigrate developers.

license GNU General Public License, Version 3 (https://www.gnu.org/licenses/gpl-3.0.html)

class quakemigrate.signal.pickers.gaussian.**GaussianPicker**(*onset=None*, ***kwargs*)
Bases: *quakemigrate.signal.pickers.base.PhasePicker*

This class details the default method of making phase picks shipped with QuakeMigrate, namely fitting a 1-D Gaussian function to the STA/LTA onset function trace for each station.

phase_picks

"GAU_P" [array-like] Numpy array stack of Gaussian pick info (each as a dict) for P phase

"GAU_S" [array-like] Numpy array stack of Gaussian pick info (each as a dict) for S phase

Type dict

pick_threshold
Picks will only be made if the onset function exceeds this percentile of the noise level (average amplitude of onset function outside pick windows). Recommended starting value: 1.0

Type float (between 0 and 1)

plot_picks
Toggle plotting of phase picks.

Type bool

pick_phases(*data*, *lut*, *event*, *event_uid*, *output*)
Picks phase arrival times for located earthquakes by fitting a 1-D Gaussian function to the P and S onset functions

DEFAULT_GAUSSIAN_FIT = {'PickValue': -1, 'popt': 0, 'xdata': 0, 'xdata_dt': 0}

fraction_tt
Handler for deprecated attribute 'fraction_tt'

pick_phases(*event*, *lut*, *run*)
Picks phase arrival times for located earthquakes.

Parameters

- **event** (`Event` object) – Contains pre-processed waveform data on which to perform picking, the event location, and a unique identifier.
- **lut** (`LUT` object) – Contains the traveltime lookup tables for seismic phases, computed for some pre-defined velocity model.
- **run** (`Run` object) – Light class encapsulating i/o path information for a given run.

Returns

- **event** (`Event` object) – Event object provided to pick_phases(), but now with phase picks!
- **picks** (*pandas.DataFrame*) – DataFrame that contains the measured picks with columns: ["Name", "Phase", "ModelledTime", "PickTime", "PickError", "SNR"] Each row contains the phase pick from one station/phase.

plot(*event*, *lut*, *picks*, *ttimes*, *run*)
Plot figure showing the filtered traces for each data component and the characteristic functions calculated from them (P and S) for each station. The search window to make a phase pick is displayed, along with the dynamic pick threshold (defined as a percentile of the background noise level), the phase pick time and its uncertainty (if made) and the Gaussian fit to the characteristic function.

Parameters **event_uid** (*str, optional*) – Earthquake UID string; for subdirectory naming within directory {run_path}/traces/

quakemigrate.signal.local_mag

The `quakemigrate.local_mag` extension module handles the calculation of local magnitudes from Wood-Anderson simulated waveforms.

Warning: The local_mag modules are an ongoing work in progress. We hope to

continue to extend their functionality, which may result in some API changes. If you have comments or suggestions, please contact the QuakeMigrate developers

at quakemigrate.developers@gmail.com , or submit an issue on GitHub.

quakemigrate.signal.local_mag.local_mag

Module containing methods to calculate the local magnitude for an event located by *QuakeMigrate*.

copyright 2020, QuakeMigrate developers.

license GNU General Public License, Version 3 (https://www.gnu.org/licenses/gpl-3.0.html)

class quakemigrate.signal.local_mag.local_mag.**LocalMag**(*amp_params*, *mag_params*, *plot_amplitudes=True*)

Bases: object

QuakeMigrate extension class for calculating local magnitudes.

Provides functions for measuring amplitudes of earthquake waveforms and using these to calculate local magnitudes.

Parameters

- **amp_params** (*dict*) – All keys are optional, including: pre_filt : tuple of floats

  Pre-filter to apply during the instrument response removal. E.g. (0.03, 0.05, 30., 35.) - all in Hz. (Default None)

  water_level [float] Water level to use in instrument response removal. (Default 60)

  signal_window [float] Length of S-wave signal window, in addition to the time window associated with the marginal_window and traveltime uncertainty. (Default 0 s)

  noise_window [float] Length of the time window before the P-wave signal window in which to measure the noise amplitude. (Default 10 s)

  noise_measure [{"RMS", "STD"}] Method by which to measure the noise amplitude; root-mean-quare or standard deviation of the signal. (Default "RMS")

  loc_method [{"spline", "gaussian", "covariance"}] Which event location estimate to use. (Default "spline")

  remove_full_response [bool] Whether to remove the full response (including the effect of digital FIR filters) or just the instrument transform function (as defined by the PolesZeros Response Stage. Significantly slower. (Default False)

  highpass_filter [bool] Whether to apply a highpass filter to the data before measuring amplitudes. (Default False)

  highpass_freq [float] High-pass filter frequency. Required if highpass_filter is True.

  bandpass_filter [bool] Whether to apply a band-pass filter before measuring amplitudes. (Default: False)

  bandpass_lowcut [float] Band-pass filter low-cut frequency. Required if bandpass_filter is True.

  bandpass_highcut [float] Band-pass filter high-cut frequency. Required if bandpass_filter is True.

  filter_corners [int] Number of corners for the chosen filter. Default: 4.

  prominence_multiplier [float] To set a prominence filter in the peak-finding algorithm. (Default 0. = off). NOTE: not recommended for use in combination with a filter; filter gain corrections can lead to spurious results. Please see the *scipy.signal.find_peaks* documentation for further guidance.

- **mag_params** (*dict*) – Required keys: A0 : str or func

Name of the attenuation function to use. Available options include {"Hutton-Boore", "keir2006", "UK", ... }. Alternatively specify a function which returns the attenuation factor at a specified (epicentral or hypocentral) distance. (Default "Hutton-Boore")

All other keys are optional, including: station_corrections : dict {str : float}

Dictionary of trace_id : magnitude-correction pairs. (Default None)

amp_feature [{"S_amp", "P_amp"}] Which phase amplitude measurement to use to calculate local magnitude. (Default "S_amp")

amp_multiplier [float] Factor by which to multiply all measured amplitudes.

use_hyp_dist [bool, optional] Whether to use the hypocentral distance instead of the epicentral distance in the local magnitude calculation. (Default False)

trace_filter [regex expression] Expression by which to select traces to use for the mean_magnitude calculation. E.g. '.*H[NE]$'. (Default None)

station_filter [list of str] List of stations to exclude from the mean_magnitude calculation. E.g. ["KVE", "LIND"]. (Default None)

dist_filter [float or False] Whether to only use stations less than a specified (epicentral or hypocentral) distance from an event in the mean_magnitude() calculation. Distance in kilometres. (Default False)

pick_filter [bool] Whether to only use stations where at least one phase was picked by the autopicker in the mean_magnitude calculation. (Default False)

noise_filter [float] Factor by which to multiply the measured noise amplitude before excluding amplitude observations below the noise level. (Default 1.)

weighted_mean [bool] Whether to do a weighted mean of the magnitudes when calculating the mean_magnitude. (Default False)

- **plot_amplitudes** (*bool, optional*) – Plot amplitudes vs. distance plot for each event. (Default True)

amp

The Amplitude object for this instance of LocalMag. Contains functions to measure Wood-Anderson corrected displacement amplitudes for an event.

Type `Amplitude` object

mag

The Magnitude object for this instance of LocalMag. Contains functions to calculate magnitudes from Wood-Anderson corrected displacement amplitudes, and to combine them into a single magnitude estimate for the event.

Type `Magnitude` object

calc_magnitude(*event*, *lut*, *run*)

calc_magnitude(*event*, *lut*, *run*)

Wrapper function to calculate the local magnitude of an event by first making Wood-Anderson corrected displacement amplitude measurements on each trace, then calculating magnitudes from these individual measurements, and a network-averaged (weighted) mean magnitude estimate and associated uncertainty.

Additional functionality includes calculating an r^2 fit of the predicted amplitude with distance curve to the observed amplitudes, and an associated plot of amplitudes vs. distance.

Parameters

- **event** (Event object) – Light class encapsulating waveform data, onset, pick and location information for a given event.
- **lut** (LUT object) – Contains the traveltime lookup tables for seismic phases, computed for some pre-defined velocity model.
- **run** (Run object) – Light class encapsulating i/o path information for a given run.

Returns

- **event** (Event object) – Light class encapsulating waveform data, onset, pick and location information for a given event. Now also contains local magnitude information.
- **mag** (*float*) – Network-averaged local magnitude estimate for this event.

quakemigrate.signal.local_mag.amplitude

Module containing methods to measure Wood-Anderson corrected waveform amplitudes to be used for local magnitude calculation.

copyright 2020, QuakeMigrate developers.

license GNU General Public License, Version 3 (https://www.gnu.org/licenses/gpl-3.0.html)

class quakemigrate.signal.local_mag.amplitude.**Amplitude**(*amplitude_params={}*)
Bases: object

Part of the QuakeMigrate LocalMag class; measures Wood-Anderson corrected waveform amplitudes to be used for local magnitude calculation.

Simulates the Wood-Anderson waveforms using a user-supplied set of response removal parameters, then measures the maximum peak-to-trough amplitude in time windows around the P and S phase arrivals. These windows are calculated from the phase pick times from the autopicker, if available, or from the modelled pick times. The length of the S-wave signal window is calculated according to a user-specified *signal_window* parameter.

The user may optionally specify a filter to apply to the waveforms before amplitudes are measured, in order (for example) to reduce the impact of high-amplitude noise associated with the oceanic microseisms on the measurement of low-amplitude wavetrains associated with microseismic events. Note this will generally result in an underestimate of the true earthquake waveform amplitude, even when the filter gain is corrected for.

A measurement of the signal amplitude in a window preceding the P-wave arrival is used to characterise the "noise" amplitude. This can be used to filter out null observations, and to provide an estimate of the uncertainty on the max amplitude measurements contributed by extraneous noise.

pre_filt
Pre-filter to apply during the instrument response removal. E.g. (0.03, 0.05, 30., 35.) - all in Hz. (Default None)

Type tuple of floats

water_level
Water level to use in instrument response removal. (Default 60.)

Type float

signal_window
Length of S-wave signal window, in addition to the time window associated with the marginal_window and traveltime uncertainty. (Default 0 s)

Type float

noise_window
Length of the time window before the P-wave signal window in which to measure the noise amplitude. (Default 5 s)

Type float

noise_measure
Method by which to measure the noise amplitude; root-mean-quare or standard deviation of the signal. (Default "RMS")

Type {"RMS", "STD"}

loc_method
Which event location estimate to use. (Default "spline")

Type {"spline", "gaussian", "covariance"}

remove_full_response
Whether to remove the full response (including the effect of digital FIR filters) or just the instrument transform function (as defined by the PolesZeros Response Stage). Significantly slower. (Default False)

Type bool

highpass_filter
Whether to apply a high-pass filter before measuring amplitudes. (Default False)

Type bool

highpass_freq
High-pass filter frequency. Required if highpass_filter is True.

Type float

bandpass_filter
Whether to apply a band-pass filter before measuring amplitudes. (Default False)

Type bool

bandpass_lowcut
Band-pass filter low-cut frequency. Required if bandpass_filter is True.

Type float

bandpass_highcut
Band-pass filter high-cut frequency. Required if bandpass_filter is True.

Type float

filter_corners
number of corners for the chosen filter. (Default 4)

Type int

prominence_multiplier
To set a prominence filter in the peak-finding algorithm. (Default 0. = off) NOTE: not recommended for use in combination with a filter; filter gain corrections can lead to spurious results. Please see the *scipy.signal.find_peaks* documentation for further guidance.

Type float

get_amplitudes(*event*, *lut*)

Raises `AttributeError` – If both highpass_filter and bandpass_filter are selected, or if the user selects to apply a filter but does not provide the relevant frequencies.

get_amplitudes(*event*, *lut*)
Measure phase amplitudes for an event.

Parameters

- **event** (Event object) – Light class encapsulating waveform data, onset, pick and location information for a given event.
- **lut** (LUT object) – Contains the traveltime lookup tables for seismic phases, computed for some pre-defined velocity model.

Returns

amplitudes – P- and S-wave amplitude measurements for each component of each station in the station file. Columns:

epi_dist [float] Epicentral distance between the station and the event hypocentre.

z_dist [float] Vertical distance between the station and the event hypocentre.

P_amp [float] Half maximum peak-to-trough amplitude in the P signal window. In *millimetres*.

P_freq [float] Approximate frequency of the maximum amplitude P-wave signal. Calculated from the peak-to-trough time of the max peak-to-trough amplitude.

P_time [*obspy.UTCDateTime* object] Approximate time of amplitude observation (halfway between peak and trough times).

S_amp [float] As for P, but in the S wave signal window.

S_freq [float] As for P.

S_time [*obspy.UTCDateTime* object] As for P.

Noise_amp [float] An estimate of the signal amplitude in the noise window. In millimetres.

is_picked [bool] Whether at least one of the phase arrivals was picked by the autopicker.

Index = Trace ID (see *obspy.Trace* object property 'id')

Return type *pandas.DataFrame* object

quakemigrate.signal.local_mag.magnitude

Module that supplies functions to calculate magnitudes from observations of trace amplitudes, earthquake location, station locations, and an estimated attenuation curve for the region of interest.

copyright 2020, QuakeMigrate developers.

license GNU General Public License, Version 3 (https://www.gnu.org/licenses/gpl-3.0.html)

class quakemigrate.signal.local_mag.magnitude.**Magnitude**(*magnitude_params={}*)
Bases: object

Part of the QuakeMigrate LocalMag class; calculates local magnitudes from Wood-Anderson corrected waveform amplitude measurements.

Takes waveform amplitude measurements from the LocalMag Amplitude class, and from these calculates local magnitude estimates using a local magnitude attenuation function. Magnitude corrections for individual stations and channels thereof can be applied, if provided.

Individual estimates are then combined to calculate a network-averaged (weighted) mean local magnitude for the event. Also includes the function to measure the r-squared statistic assessing the goodness of fit between the predicted amplitude with distance from the nework-averaged local magnitude for the event and chosen attenuation function, and the observed amplitudes. This, provides a tool to distinguish between real microseismic events and artefacts.

A summary plot illustrating the amplitude observations, their uncertainties, and the predicted amplitude with distance for the network- averaged local magnitude (and its uncertainties) can optionally be output.

A0
Name of the attenuation function to use. Available options include {"Hutton-Boore", "keir2006", "UK", ...}. Alternatively specify a function which returns the attenuation factor at a specified (epicentral or hypocentral) distance. (Default "Hutton-Boore")

Type str or func

use_hyp_dist
Whether to use the hypocentral distance instead of the epicentral distance in the local magnitude calculation. (Default False)

Type bool, optional

amp_feature
Which phase amplitude measurement to use to calculate local magnitude. (Default "S_amp")

Type {"S_amp", "P_amp"}

station_corrections
Dictionary of trace_id : magnitude-correction pairs. (Default None)

Type dict {str : float}

amp_multiplier
Factor by which to multiply all measured amplitudes.

Type float

weighted_mean
Whether to use a weighted mean to calculate the network-averaged local magnitude estimate for the event. (Default False)

Type bool

trace_filter
Expression by which to select traces to use for the mean_magnitude calculation. E.g. ".*H[NE]$" . (Default None)

Type regex expression

noise_filter
Factor by which to multiply the measured noise amplitude before excluding amplitude observations below the noise level. (Default 1.)

Type float

station_filter
List of stations to exclude from the mean_magnitude calculation. E.g. ["KVE", "LIND"]. (Default None)

Type list of str

dist_filter
Whether to only use stations less than a specified (epicentral or hypocentral) distance from an event in the mean_magnitude() calculation. Distance in kilometres. (Default False)

Type float or False

pick_filter
Whether to only use stations where at least one phase was picked by the autopicker in the mean_magnitude calculation. (Default False)

Type bool

calculate_magnitudes(*amplitudes*)

mean_magnitude(*magnitudes*)

plot_amplitudes(*event*, *run*)

Raises

- `AttributeError` – If the user does not specify an A0 attenuation curve.
- `ValueError` – If the user specifies an invalid A0 attenuation curve.

calculate_magnitudes(*amplitudes*)
Calculate magnitude estimates from amplitude measurements on individual stations / components.

Parameters **amplitudes** (*pandas.DataFrame* object) – P- and S-wave amplitude measurements for each component of each station in the station file. Columns:

epi_dist [float] Epicentral distance between the station and the event hypocentre.

z_dist [float] Vertical distance between the station and the event hypocentre.

P_amp [float] Half maximum peak-to-trough amplitude in the P signal window. In *millimetres*.

P_freq [float] Approximate frequency of the maximum amplitude P-wave signal. Calculated from the peak-to-trough time of the max peak-to-trough amplitude.

P_time [*obspy.UTCDateTime* object] Approximate time of amplitude observation (halfway between peak and trough times).

S_amp [float] As for P, but in the S wave signal window.

S_freq [float] As for P.

S_time [*obspy.UTCDateTime* object] As for P.

Noise_amp [float] An estimate of the signal amplitude in the noise window. In millimetres.

is_picked [bool] Whether at least one of the phase arrivals was picked by the autopicker.

Index = Trace ID (see *obspy.Trace* object property 'id')

Returns

magnitudes – The original amplitudes DataFrame, with columns containing the calculated magnitude and an associated error now added. Columns = ["epi_dist", "z_dist", "P_amp", "P_freq", "P_time",

"S_amp", "S_freq", "S_time", "Noise_amp", "is_picked", "ML", "ML_Err"]

Index = Trace ID (see *obspy.Trace.id*) Additional fields: ML : float

Magnitude calculated from the chosen amplitude measurement, using the specified attenuation curve and station_corrections.

ML_Err [float] estimate of the error on the calculated magnitude, based on potential errors in the maximum amplitude measurement according to the measured noise amplitude.

Return type *pandas.DataFrame* object

Raises `AttributeError` – If A0 attenuation correction is not specified.

mean_magnitude(*magnitudes*)

Calculate the network-averaged local magnitude for an event based on the magnitude estimates calculated from amplitude measurements made on each component of each station.

The user may specify distance, station, channel and a number of other filters to restrict which observations are included in this best estimate of the local magnitude of the event.

Parameters **magnitudes** (*pandas.DataFrame*) – Contains P- and S-wave amplitude measurements for each component of each station in the station file, and local magnitude estimates calculated from them (output by calculate_magnitudes()). Note that the amplitude observations are raw, but the ML estimates derived from them include station corrections, if provided. Columns:

epi_dist [float] Epicentral distance between the station and the event hypocentre.

z_dist [float] Vertical distance between the station and the event hypocentre.

P_amp [float] Half maximum peak-to-trough amplitude in the P signal window. In *millimetres*.

P_freq [float] Approximate frequency of the maximum amplitude P-wave signal. Calculated from the peak-to-trough time of the max peak-to-trough amplitude.

P_time [*obspy.UTCDateTime* object] Approximate time of amplitude observation (halfway between peak and trough times).

S_amp [float] As for P, but in the S wave signal window.

S_freq [float] As for P.

S_time [*obspy.UTCDateTime* object] As for P.

Noise_amp [float] An estimate of the signal amplitude in the noise window. In millimetres.

is_picked [bool] Whether at least one of the phase arrivals was picked by the autopicker.

ML [float] Magnitude calculated from the chosen amplitude measurement, using the specified attenuation curve and station_corrections.

ML_Err [float] estimate of the error on the calculated magnitude, based on potential errors in the maximum amplitude measurement according to the measured noise amplitude.

Index = Trace ID (see *obspy.Trace* object property 'id')

Returns

- **mean_mag** (*float or NaN*) – Network-averaged local magnitude estimate for the event. Mean (or weighted mean) of the magnitude estimates calculated from each individual channel after optionally removing some observations based on trace ID, distance, etcetera.

- **mean_mag_err** (*float or NaN*) – Standard deviation (or weighted standard deviation) of the magnitude estimates calculated from individual channels which contributed to the calculation of the (weighted) mean magnitude.
- **mag_r_squared** (*float or NaN*) – r-squared statistic describing the fit of the amplitude vs. distance curve predicted by the calculated mean_mag and chosen attenuation model to the measured amplitude observations. This is intended to be used to help discriminate between 'real' events, for which the predicted amplitude vs. distance curve should provide a good fit to the observations, from artefacts, which in general will not.

plot_amplitudes(*magnitudes*, *event*, *run*, *unit_conversion_factor*, *noise_measure='RMS'*)

Plot a figure showing the measured amplitude with distance vs. predicted amplitude with distance derived from mean_mag and the chosen attenuation model.

The amplitude observations (both for noise and signal amplitudes) are corrected according to the same station corrections that were used in calculating the individual local magnitude estimates that were used to calculate the network-averaged local magnitude for the event.

Parameters

- **magnitudes** (*pandas.DataFrame* object) – Contains P- and S-wave amplitude measurements for each component of each station in the station file, and local magnitude estimates calculated from them (output by calculate_magnitudes()). Note that the amplitude observations are raw, but the ML estimates derived from them include station corrections, if provided. Columns = ["epi_dist", "z_dist", "P_amp", "P_freq", "P_time",

  "S_amp", "S_freq", "S_time", "Noise_amp", "is_picked", "ML", "ML_Err", "Noise_Filter", "Trace_Filter", "Station_Filter", "Dist_Filter", "Dist", "Used"]
- **event** (`Event` object) – Light class encapsulating waveform data, onset, pick, location and local magnitude information for a given event.
- **run** (`Run` object) – Light class encapsulating i/o path information for a given run.
- **unit_conversion_factor** (*float*) – A conversion factor based on the lookup table grid projection, used to ensure the location uncertainties have units of kilometres.

quakemigrate.signal.scan

Module to perform core QuakeMigrate functions: detect() and locate().

copyright 2020, QuakeMigrate developers.

license GNU General Public License, Version 3 (https://www.gnu.org/licenses/gpl-3.0.html)

class quakemigrate.signal.scan.**QuakeScan**(*archive*, *lut*, *onset*, *run_path*, *run_name*, ***kwargs*)

Bases: `object`

QuakeMigrate scanning class.

Provides an interface for the wrapped compiled C functions, used to perform the continuous scan (detect) or refined event migrations (locate).

Parameters

- **archive** (`Archive` object) – Details the structure and location of a data archive and provides methods for reading data from file.

- **lut** (LUT object) – Contains the traveltime lookup tables for seismic phases, computed for some pre-defined velocity model.
- **onset** (Onset object) – Provides callback methods for calculation of onset functions.
- **run_path** (*str*) – Points to the top level directory containing all input files, under which the specific run directory will be created.
- **run_name** (*str*) – Name of the current QuakeMigrate run.
- **kwargs** (**dict) – See QuakeScan Attributes for details. In addition to these:

continuous_scanmseed_write
Option to continuously write the .scanmseed file output by detect() at the end of every time step. Default behaviour is to write in day chunks where possible. Default: False.

Type bool

cut_waveform_format
File format used when writing waveform data. We support any format also supported by ObSpy - "MSEED" (default), "SAC", "SEGY", "GSE2".

Type str, optional

log
Toggle for logging. If True, will output to stdout and generate a log file. Default is to only output to stdout.

Type bool, optional

loglevel
Toggle to set the logging level: "debug" will print out additional diagnostic information to the log and stdout. (Default "info")

Type {"info", "debug"}, optional

mags
Provides methods for calculating local magnitudes, performed during locate.

Type LocalMag object, optional

marginal_window
Half-width of window centred on the maximum coalescence time. The 4-D coalescence functioned is marginalised over time across this window such that the earthquake location and associated uncertainty can be appropriately calculated. It should be an estimate of the time uncertainty in the earthquake origin time, which itself is some combination of the expected spatial uncertainty and uncertainty in the seismic velocity model used. Default: 2 seconds.

Type float, optional

picker
Provides callback methods for phase picking, performed during locate.

Type PhasePicker object, optional

plot_event_summary
Plot event summary figure - see *quakemigrate.plot* for more details. Default: True.

Type bool, optional

plot_event_video
Plot coalescence video for each located earthquake. Default: False.

Type bool, optional

post_pad
Additional amount of data to read in after the timestep, used to ensure the correct coalescence is calculated at every sample.

Type float

pre_pad
Additional amount of data to read in before the timestep, used to ensure the correct coalescence is calculated at every sample.

Type float

run
Light class encapsulating i/o path information for a given run.

Type `Run` object

sampling_rate
Desired sampling rate of input data; sampling rate at which to compute the coalescence function. Default: 50 Hz.

Type int, optional

threads
The number of threads for the C functions to use on the executing host. Default: 1 thread.

Type int, optional

timestep
Length (in seconds) of timestep used in detect(). Note: total detect run duration should be divisible by timestep. Increasing timestep will increase RAM usage during detect, but will slightly speed up overall detect run. Default: 120 seconds.

Type float, optional

write_cut_waveforms
Write raw cut waveforms for all data found in the archive for each event located by locate(). Default: False. Note: this data has not been processed or quality-checked!

Type bool, optional

xy_files
Path to comma-separated value file (.csv) containing a series of coordinate files to plot. Columns: ["File", "Color", "Linewidth", "Linestyle"], where "File" is the absolute path to the file containing the coordinates to be plotted. E.g: "/home/user/volcano_outlines.csv,black,0.5,-". Each .csv coordinate file should contain coordinates only, with columns: ["Longitude", "Latitude"]. E.g.: "-17.5,64.8". Lines pre-pended with # will be treated as a comment - this can be used to include references. See the Volcanotectonic_Iceland example XY_files for a template.

Note: Do not include a header line in either file.

Type str, optional

+++ TO BE REMOVED TO ARCHIVE CLASS +++

pre_cut
Specify how long before the event origin time to cut the waveform data from

Type float, optional

post_cut
Specify how long after the event origin time to cut the waveform data to

Type float, optional

+++ TO BE REMOVED TO ARCHIVE CLASS +++

detect (*starttime*, *endtime*)
Core detection method – compute decimated 3-D coalescence continuously throughout entire time period; output as .scanmseed (in mSEED format).

locate (*starttime*, *endtime) or locate(file*)
Core locate method – compute 3-D coalescence over short time window around candidate earthquake triggered from coastream; output location & uncertainties (.event file), phase picks (.picks file), plus multiple optional plots / data for further analysis and processing.

Raises

- `OnsetTypeError` – If an object is passed in through the *onset* argument that does not derive from the `Onset` base class.
- `PickerTypeError` – If an object is passed in through the *picker* argument that does not derive from the `PhasePicker` base class.
- `RuntimeError` – If the user does not supply the locate function with valid arguments.
- `TimeSpanException` – If the user supplies a starttime that is after the endtime.
- `NoMagObjectError` – If the user selects to calculate magnitudes but does not provide a `LocalMag` object.

detect (*starttime*, *endtime*)
Scans through continuous data calculating coalescence on a (decimated) 3-D grid by back-migrating onset (characteristic) functions.

Parameters

- **starttime** (*str*) – Timestamp from which to run continuous scan (detect).
- **endtime** (*str*) – Timestamp up to which to run continuous scan (detect). Note: the last sample returned will be that which immediately precedes this timestamp.

locate (*starttime=None*, *endtime=None*, *trigger_file=None*)
Re-computes the 3-D coalescence on an undecimated grid for a short time window around each candidate earthquake triggered from the (decimated) continuous detect scan. Calculates event location and uncertainties, makes phase arrival picks, plus multiple optional plotting / data outputs for further analysis and processing.

Parameters

- **starttime** (*str, optional*) – Timestamp from which to include events in the locate scan.
- **endtime** (*str, optional*) – Timestamp up to which to include events in the locate scan.
- **trigger_file** (*str, optional*) – File containing triggered events to be located.

n_cores
Handler for deprecated attribute name 'n_cores'

sampling_rate
Get sampling_rate

time_step
Handler for deprecated attribute name 'time_step'

quakemigrate.signal.trigger

Module to perform the trigger stage of QuakeMigrate.

copyright 2020, QuakeMigrate developers.

license GNU General Public License, Version 3 (https://www.gnu.org/licenses/gpl-3.0.html)

class quakemigrate.signal.trigger.**Trigger**(*lut*, *run_path*, *run_name*, ***kwargs*)
Bases: object

QuakeMigrate triggering class.

Triggers candidate earthquakes from the maximum coalescence through time data output by the decimated detect scan, ready to be run through locate().

Parameters

- **lut** (LUT object) – Contains the traveltime lookup tables for P- and S-phases, computed for some pre-defined velocity model.
- **run_path** (*str*) – Points to the top level directory containing all input files, under which the specific run directory will be created.
- **run_name** (*str*) – Name of the current QuakeMigrate run.
- **kwargs** (**dict) – See Trigger Attributes for details. In addition to these: log : bool, optional

  Toggle for logging. If True, will output to stdout and generate a log file. Default is to only output to stdout.

  loglevel [{"info", "debug"}, optional] Toggle to set the logging level: "debug" will print out additional diagnostic information to the log and stdout. (Default "info")

  trigger_name [str] Optional name of a sub-run - useful when testing different trigger parameters, for example.

mad_window_length
Length of window within which to calculate the Median Average Deviation. Default: 3600 seconds (1 hour).

Type float, optional

mad_multiplier
A scaling factor for the MAD output to make the calculated MAD factor a consistent estimation of the standard deviation of the distribution. Default: 1.4826, which is the appropriate scaling factor for a normal distribution.

Type float, optional

marginal_window
Time window over which to marginalise the coalescence, making it solely a function of the spatial dimensions. This should be an estimate of the time error, as derived from an estimate of the spatial error and error in the velocity model. Default: 2 seconds.

Type float, optional

min_event_interval
Minimum time interval between triggered events. Must be at least twice the marginal window. Default: 4 seconds.

Type float, optional

normalise_coalescence
If True, triggering is performed on the maximum coalescence normalised by the mean coalescence value in the 3-D grid. Default: False.

Type bool, optional

pad
Additional time padding to ensure events close to the starttime/endtime are not cut off and missed. Default: 120 seconds.

Type float, optional

run
Light class encapsulating i/o path information for a given run.

Type `Run` object

static_threshold
Static threshold value above which to trigger candidate events.

Type float, optional

threshold_method
Toggle between a "static" threshold and a "dynamic" threshold, based on the Median Average Deviation. Default: "static".

Type str, optional

xy_files
Path to comma-separated value file (.csv) containing a series of coordinate files to plot. Columns: ["File", "Color", "Linewidth", "Linestyle"], where "File" is the absolute path to the file containing the coordinates to be plotted. E.g: "/home/user/volcano_outlines.csv,black,0.5,-". Each .csv coordinate file should contain coordinates only, with columns: ["Longitude", "Latitude"]. E.g.: "-17.5,64.8". Lines pre-pended with # will be treated as a comment - this can be used to include references. See the Volcanotectonic_Iceland example XY_files for a template.

Note: Do not include a header line in either file.

Type str, optional

trigger(*starttime*, *endtime*, *region=None*, *savefig=True*)
Trigger candidate earthquakes from decimated detect scan results.

Raises

- `ValueError` – If *min_event_interval* < 2 * *marginal_window*.
- `InvalidThresholdMethodException` – If an invalid threshold method is passed in by the user.
- `TimeSpanException` – If the user supplies a starttime that is after the endtime.

min_event_interval
Get and set the minimum event interval.

minimum_repeat
Handler for deprecated attribute name 'minimum_repeat'.

trigger(*starttime*, *endtime*, *region=None*, *savefig=True*)
Trigger candidate earthquakes from decimated scan data.

Parameters

- **starttime** (*str*) – Timestamp from which to trigger.
- **endtime** (*str*) – Timestamp up to which to trigger.
- **region** (*list of floats, optional*) – Only write triggered events within this region to the triggered events csv file (for use in locate.) Format is:

  [Xmin, Ymin, Zmin, Xmax, Ymax, Zmax]

  Units are longitude / latitude / metres (in positive-down frame).
- **savefig** (*bool, optional*) – Save triggered events figure (default) or open interactive view.

Raises TimeSpanException – If *starttime* is after *endtime*.

quakemigrate.signal.trigger.**calculate_mad**(*x*, *scale=1.4826*)
Calculates the Median Absolute Deviation (MAD) of the input array x.

Parameters

- **x** (*array-like*) – Coalescence array in.
- **scale** (*float, optional*) – A scaling factor for the MAD output to make the calculated MAD factor a consistent estimation of the standard deviation of the distribution.

Returns **scaled_mad** – Array of scaled mean absolute deviation values for the input array, x, scaled to provide an estimation of the standard deviation of the distribution.

Return type array-like

quakemigrate.signal.trigger.**chunks2trace**(*a*, *new_shape*)
Create a trace filled with chunks of the same value.

a [array-like] Array of chunks.

new_shape [tuple of ints] (number of chunks, chunk_length)

b [array-like] Single array of values contained in *a*.

5.3.7 quakemigrate.util

Module that supplies various utility functions and classes.

copyright 2020, QuakeMigrate developers.

license GNU General Public License, Version 3 (https://www.gnu.org/licenses/gpl-3.0.html)

exception quakemigrate.util.**ArchiveEmptyException**
Bases: Exception

Custom exception to handle empty archive

exception quakemigrate.util.**ArchiveFormatException**
Bases: Exception

Custom exception to handle case where Archive.format is not set.

exception quakemigrate.util.**ArchivePathStructureError**(*archive_format*)
Bases: Exception

Custom exception to handle case where an invalid Archive path structure is selected.

exception quakemigrate.util.**BadUpfactorException**(*trace*)
Bases: Exception

Custom exception to handle case when the chosen upfactor does not create a trace with a sampling rate that can be decimated to the target sampling rate

exception quakemigrate.util.**ChannelNameException**(*trace*)
Bases: Exception

Custom exception to handle case when waveform data header has channel names which do not conform to the IRIS SEED standard.

exception quakemigrate.util.**DataGapException**
Bases: Exception

Custom exception to handle case when all data has gaps for a given timestep

class quakemigrate.util.**DateFormatter**(*fmt*, *precision=3*)
Bases: matplotlib.ticker.Formatter

Extend the *matplotlib.ticker.Formatter* class to allow for millisecond precision when formatting a tick (in days since the epoch) with a *~datetime.datetime.strftime* format string.

exception quakemigrate.util.**InvalidThresholdMethodException**
Bases: Exception

Custom exception to handle case when the user has not selected a valid threshold method.

exception quakemigrate.util.**InvalidVelocityModelHeader**(*key*)
Bases: Exception

Custom exception to handle incorrect header columns in station file

exception quakemigrate.util.**MagsTypeError**
Bases: Exception

Custom exception to handle case when an object has been provided to calculate magnitudes during locate, but it isn't supported.

exception quakemigrate.util.**NoScanMseedDataException**
Bases: Exception

Custom exception to handle case when no .scanmseed files can be found by read_coastream()

exception quakemigrate.util.**NoStationAvailabilityDataException**
Bases: Exception

Custom exception to handle case when no .StationAvailability files can be found by read_availability()

exception quakemigrate.util.**NoTriggerFilesFound**
Bases: Exception

Custom exception to handle case when no trigger files are found during locate. This can occur for one of two reasons - an entirely invalid time period was used (i.e. one that does not overlap at all with a period of time for which there exists TriggeredEvents.csv files) or an invalid run name was provided.

exception quakemigrate.util.**NyquistException**(*freqmax*, *f_nyquist*, *tr_id*)
Bases: Exception

Custom exception to handle the case where the specified filter has a lowpass corner above the signal Nyquist frequency.

Parameters

- **freqmax** (*float*) – Specified lowpass frequency for filter
- **f_nyquist** (*float*) – Nyquist frequency for the relevant waveform data
- **tr_id** (*str*) – ID string for the Trace

exception quakemigrate.util.**OnsetTypeError**
Bases: Exception

Custom exception to handle case when the onset object passed to QuakeScan is not of the default type defined in QuakeMigrate.

exception quakemigrate.util.**PeakToTroughError**(*err*)
Bases: Exception

Custom exception to handle case when amplitude._peak_to_trough_amplitude encounters an anomalous set of peaks and troughs, so can't calculate an amplitude.

exception quakemigrate.util.**PickOrderException**(*event_uid*, *station*, *p_pick*, *s_pick*)
Bases: Exception

Custom exception to handle the case when the pick for the P phase is later than the pick for the S phase.

exception quakemigrate.util.**PickerTypeError**
Bases: Exception

Custom exception to handle case when the phase picker object passed to QuakeScan is not of the default type defined in QuakeMigrate.

exception quakemigrate.util.**ResponseNotFoundError**(*e*, *tr_id*)
Bases: Exception

Custom exception to handle the case where the provided response inventory doesn't contain the response information for a trace.

Parameters

- **e** (*str*) – Error message from ObsPy *Inventory.get_response()*
- **tr_id** (*str*) – ID string for the Trace for which the response cannot be found

exception quakemigrate.util.**ResponseRemovalError**(*e*, *tr_id*)
Bases: Exception

Custom exception to handle the case where the response removal was not successful.

Parameters

- **e** (*str*) – Error message from ObsPy *Trace.remove_response()* or *Trace.simulate()*
- **tr_id** (*str*) – ID string for the Trace for which the response cannot be removed

exception quakemigrate.util.**StationFileHeaderException**
Bases: Exception

Custom exception to handle incorrect header columns in station file

exception quakemigrate.util.**TimeSpanException**
Bases: Exception

Custom exception to handle case when the user has submitted a start time that is after the end time.

quakemigrate.util.**decimate**(*trace*, *sr*)
Decimate a trace to achieve the desired sampling rate, sr.

NOTE: data will be detrended and a cosine taper applied before decimation, in order to avoid edge effects when applying the lowpass filter.

trace [*obspy.Trace* object] Trace to be decimated.

sr [int] Output sampling rate.

trace [*obspy.Trace* object] Decimated trace.

quakemigrate.util.**gaussian_1d**(*x*, *a*, *b*, *c*)
Create a 1-dimensional Gaussian function.

Parameters

- **x** (*array-like*) – Array of x values
- **a** (*float / int*) – Amplitude (height of Gaussian)
- **b** (*float / int*) – Mean (centre of Gaussian)
- **c** (*float / int*) – Sigma (width of Gaussian)

Returns **f** – 1-dimensional Gaussian function

Return type function

quakemigrate.util.**gaussian_3d**(*nx*, *ny*, *nz*, *sgm*)
Create a 3-dimensional Gaussian function.

Parameters

- **nx** (*array-like*) – Array of x values
- **ny** (*array-like*) – Array of y values
- **nz** (*array-like*) – Array of z values
- **sgm** (*float / int*) – Sigma (width of gaussian in all directions)

Returns **f** – 3-dimensional Gaussian function

Return type function

quakemigrate.util.**logger**(*logstem*, *log*, *loglevel='info'*)
Simple logger that will output to both a log file and stdout.

Parameters

- **logstem** (*str*) – Filestem for log file.
- **log** (*bool*) – Toggle for logging - default is to only print information to stdout. If True, will also create a log file.
- **loglevel** (*str, optional*) – Toggle for logging level - default is to print only "info" messages to log. To print more detailed "debug" messages, set to "debug".

quakemigrate.util.**make_directories**(*run*, *subdir=None*)
Make run directory, and optionally make subdirectories within it.

Parameters

- **run** (*pathlib Path object*) – Location of parent output directory, named by run name.
- **subdir** (*string, optional*) – subdir to make beneath the run level.

quakemigrate.util.**time2sample**(*time*, *sampling_rate*)
Utility function to convert from seconds and sampling rate to number of samples.

Parameters

- **time** (*float*) – Time to convert
- **sampling_rate** (*int*) – Sampling rate of input data/sampling rate at which to compute the coalescence function.

Returns **out** – Time that correpsonds to an integer number of samples at a specific sampling rate.

Return type int

quakemigrate.util.**timeit**(**args_*, ***kwargs_*)
Function wrapper that measures the time elapsed during its execution.

quakemigrate.util.**trim2sample**(*time*, *sampling_rate*)
Utility function to ensure time padding results in a time that is an integer number of samples.

Parameters

- **time** (*float*) – Time to trim.
- **sampling_rate** (*int*) – Sampling rate of input data/sampling rate at which to compute the coalescence function.

Returns **out** – Time that correpsonds to an integer number of samples at a specific sampling rate.

Return type int

quakemigrate.util.**upsample**(*trace*, *upfactor*)
Upsample a data stream by a given factor, prior to decimation. The upsampling is done using a linear interpolation.

Parameters

- **trace** (*obspy.Trace* object) – Trace to be upsampled.
- **upfactor** (*int*) – Factor by which to upsample the data in trace.

Returns **out** – Upsampled trace.

Return type *obpsy.Trace* object

quakemigrate.util.**wa_response**(*convert='DIS2DIS'*, *obspy_def=True*)
Generate a Wood Anderson response dictionary.

Parameters

- **convert** (*{'DIS2DIS', 'VEL2VEL', 'VEL2DIS'}*) – Type of output to convert between; determines the number of complex zeros used.
- **obspy_def** (*bool, optional*) – Use the ObsPy definition of the Wood Anderson response (Default). Otherwise, use the IRIS/SAC definition.

Returns **WOODANDERSON** – Poles, zeros, sensitivity and gain of the Wood-Anderson torsion seismograph.

Return type dict

Python Module Index

q

Index

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

X