DLOPY V1.0
README

Overview

This program estimates the geographic orientation of the horizontal components of a three-component broadband seismometer. The program is configured for data stored at the IRIS DMC but can also analyze data stored locally. Arrival angles of Rayleigh waves at seven discrete frequencies between 10 and 40 mHz are measured by cross-correlating the radial component with the phase-shifted vertical component. We perform a grid search, rotating the horizontal components by 0.25 degree increments from 0° to 360°, to maximize the correlation. Measurements above a specified cross-correlation threshold (usually 0.8) and within five times the median absolute deviation are retained, and the bootstrap method of random sampling with replacement is used to find the mean and the 4σ uncertainty of the mean. Since the program relies on low-frequency passive data, longer deployments and quieter stations generally provide more accurate results.

We refer to the entire orientation toolbox as DLOPy. The source code for the orientation algorithm is `Orient.py`. This package is written in Python 2.7. An installation script called `setup-DLOPy.sh` is included. See additional notes on installation below. This software is copyright by the Regents of the University of California and licensed under the Simplified BSD License; see `license.txt` for more info. Software author information is contained in `authors.txt`. Contact me at `adoran@ucsd.edu` with any comments, ideas, or bugs.

Reference:

Doran, A. K., and G. Laske (2017), Ocean-bottom seismometer instrument orientations via automated Rayleigh-wave arrival angle measurements, *Bulletin of the Seismological Society of America*, *107*(2), 691-708, doi: 10.1785/012016,165.

Installation

DLOPy has been tested on versions 2.7.6, 2.7.10, and 2.7.13, the most recent version as of April 2017 (more information at `https://www.python.org/`). These codes have been used on machines running Mac OS X 10.12.4 ('Sierra') and Mac OS X 10.11.6 ('El Capitan').

DLOPy requires the following additional Python packages:

- `numpy` (Version as of writing: 1.12.1)
  NumPy is a basic Python package for scientific computing. More information at `http://www.numpy.org`.
- `scipy` (Version as of writing: 0.19.0)
  SciPy is another basic scientific Python package that aids with signal processing and the surface wave correlations. More information at `https://www.scipy.org`.
- `matplotlib` (Version as of writing: 2.0.0)
  Matplotlib is a Python plotting library used by DLOPy to visualize the results. More information at `http://matplotlib.org`.

Date: April 2017.

- `obspy` (Version as of writing: 1.0.3)
  ObsPy is an open-source platform to download waveform data, access event catalog information, and perform basic data processing. DLOPy relies on ObsPy to acquire data and organize the earthquake catalog (see Section for details). More information at `https://www.obspy.org` and in the following publications: *Beyreuther et al.* (2010); *Megies et al.* (2011); *Krischer et al.* (2015).

Recommended Installation. To access and install the required Python software, we suggest using the command line package manager `pip`. All Python packages required by DLOPy are accessible through `pip`. For installation or upgrade information, see `https://pip.pypa.io/en/stable/installing/`.

Running `setup-DLOPy.sh` will install and upgrade the Python packages required by DLOPy using `pip`. The script proceeds as follows:

(1) Install and upgrade `numpy`
(2) Install and upgrade `scipy`
(3) Install and upgrade `matplotlib`
(4) Install and upgrade `obspy`

The Python package `geographiclib` (Version 1.47) is included with this DLOPy distribution (*Karney* (2013); more information at `http://geographiclib.sourceforge.net`). This software is used for geodetic computations along great-circle-arcs.

Troubleshooting. Here are some common problems I have run into while installing DLOPy on new machines:

- Permission problems
  Installing Python packages through `pip` may require root privileges. If you lack the necessary permission, `pip` will issue a warning and will not install the software. You may need to use the `sudo` command to run the installation script by entering:
  ```
  sudo setup-DLOPy.sh
  ```
  and then entering the required password.
- ObsPy installation problems
  ObsPy is a powerful package with many tools but requires a number of additional Python packages. During installation, `pip` will generally alert you to any problems and propose solutions. If one aspect of ObsPy fails to build, install and / or update the problematic software individually and then retry installing Obspy. See more information on ObsPy installation at `https://github.com/obspy/obspy/wiki#installation`.
- Python can't find installed packages
  If you have installed the necessary software but Python seems unable to locate it, you may be having trouble with your Python path. The command
  ```
  which python
  ```
  will show you the location of your Python distribution. In addition, to see the location of the installed packages (in this example `obspy`), enter
  ```
  pip show obspy
  ```
  If these show different directories, Python won't be able to import the downloaded packages. I suggest either reinstalling the software in the appropriate location or adjusting your `PYTHONPATH` environmental variable, which sets the search path for importing Python modules, to include the necessary directories.

Please contact me if you experience other installation problems.

Other installation options. The necessary DLOPy packages can be installed in other ways, including from source or through an interactive development environment such as Enthought Canopy or Continuum Anaconda. Other command line package managers such as `macports` (`https://www.macports.org`) also provide access to the necessary software.

Work flow

To analyze the sensor orientation at one station using DLOPy, proceed as follows:

(1) Set options in parameter file (`Orient_PF.py`)

All program inputs are set in this file. The user defines the station, network, channels, and location code of the sensor, defines which time period to consider, and sets a number of other variables. Table 2 described all the possible options that may be adjusted. The program utilizes SEED or miniSEED format for all data processing. An important consideration during the orientation calculation is the coordinate system of the seismometer. Generally, the codes assume a left-handed system, in which H1 is the nominally north component and H2 is nominally east, 90° clockwise from H1 (see Figure 1). Other coordinate systems can be specified in the parameter file. No matter which system is used, the program will calculate the orientation of the nominally north component.

(2) Run orientation program (`Orient.py`)

Once the parameter file is set, run the program by entering

```
python Orient.py
```

ObsPy accesses both catalog information and waveform data through an FDSN Web Service client. In default mode, it accesses the "IRIS" catalog, which is compiled from the USGS NEIC catalog, the ISC catalog, and the Global CMT project (more information and citation details at `http://service.iris.edu/fdsnws/event/docs/1/help/`). "IRIS" is also the default source of the waveform data, meaning data is accessed from the IRIS DMC. Different data and / or catalog servers may be selected in the parameter file. See Section to read local data.

DLOPy consults global dispersion maps (*Ma et al.*, 2014; *Ma and Masters*, 2014) to compute the Rayleigh wave window as a function of frequency. Although our original publication describes a hybrid Fortran-Python code, DLOPy now runs entirely in Python in order to simplify the code and reduce the time required per calculation. We include dispersion maps sampled at discrete frequencies as ASCII files. The longitude and latitude nodes are located at the center points of the equal area block model. Additional information is available at `https://igppweb.ucsd.edu/~gabi/litho1.0.html`.

The program will create the following files while running:

- An event catalog titled `STA.cat.txt` containing information regarding each earthquake considered in the analysis
- A folder titled `STA/` containing the arrival angles (R1phi, R2phi) and cross correlation values (R1cc, R2cc) for all events included in the analysis. The R1 and R2 files are the results for the 1st and 2nd great-circle-arc Raleigh waves used by DLOPy in the final calculation.

Experienced users interested in the individual arrival angles can increase the precision of the measurements by using a finer grid search than 0.25°. To do this, edit the function `SW1` in `locfuns.py` to search in whatever increment is desired.

(3) Perform final calculation (`compcalcs.py`)

Several options must be set in the final calculation file before it is run. First, the user must specify the folder of results to consider. The user may also adjust the cross-correlation threshold, which defines the minimum-quality measurement used in the final calculation. The default is 0.8, and this value is recommended for most stations. However, at certain stations the ambient noise conditions may require a lower or higher value to include or exclude more data. The user may also adjust the time period considered or which frequency results to include. All possible parameters are described in Table 3.

Run the final calculation by entering:
```
python compcalcs.py
```

In the default mode, the program will output the following values:
mean orientation, uncertainty, number of data, unique events.

The number of data refers to the number of individual measurements used in the final calculation, while the unique events counts how many distinct earthquakes these data are taken from. The program will also generate a figure showing the H1 orientation angle as a function of cross-correlation value. A clustering of similar angles at the highest correlation should appear, indicating the orientation of the seismometer. This figure is saved in the arrival angle folder as `cluster.eps`.

The final orientation estimate is computed from results above the cross-correlation threshold and within five times the median absolute deviation. The former criterion ensures only high-quality measurements are used, and the latter removes unrealistic outliers. We use the bootstrap method of random sampling with replacement to compute the arithmetic mean of 5000 runs. The mean of these values is taken as the orientation, and we report the error as the 4σ uncertainty. It is worth noting that this can lead to small variations between final calculations, but the discrepancy is usually quite small ($< 0.05°$), much less than the final uncertainty value.

<u>The output angle is the clockwise rotation in degrees from geographic North of the nominally north component.</u>

Although we have included a great many precautions to guard against bias in the absence of a formal inversion, in extreme cases, the user should be prepared to accept a potential bias of $\pm 2°$.

Reading Local Data

If the data is stored locally, the file `readlocal.py` must be used. The data files must be SEED or miniSEED format, and each file must contain one day of data for one channel. Several adjustments to the codes must be made.
In `readlocal`:

- Specify working directory with variable `dir1`
- Specify file names with variables `d` and `d2`
- Set coordinate system by manually specifying channels with variables `C1, C2, C3`
- Define sampling rate of instruments with variable `sps`

In `Orient_PF`:

- Set flag `localdata=1` so that program knows where to find data
- Set flag `localcoords=1` to manually set station location
- Specify `inputlat` and `inputlon` with correct latitude and longitude of station

Once these are set, `Orient` can be run and the program proceeds as usual.

Example calculation

The parameter file is preconfigured to analyze one month of data (01 May 1998 - 01 June 1998) from station OSN1B, an OBS package deployed southwest of Oahu as part of the Ocean Seismic Network Pilot Experiment (OSNPE; *Collins et al.*, 2001; *Stephen et al.*, 2003). Three seismometers were deployed during this experiment: one borehole (OSN1), one buried (OSN1B), and one seafloor (OSN1S). The stations operated for approximately four months in 1998. The data are available from the IRIS DMC under the network code XM.

To test that the codes are working propersly, run `Orient.py` without making any adjustments. The program will produce the arrival angle and catalog files described above. Run `compcalcs.py` to use these files to estimate the station orientation. You should get results consistent with those described in Table 1 (also described in *Doran and Laske* (2017)). Try adjusting the parameter file to consider a longer time period or a different OSNPE station.

Troubleshooting

- `Orient.py` runs but nothing happens

With the default verbosity level, `Orient.py` should output the 30 mHz R1 result for every event calculation. If nothing is happening, the program is not performing any calculations. The most common problem is that there is no data available for download. Check the input parameters (particularly the location code and the channel naming convention) and the time period to ensure data exists that matches what you've requested.
Another possibility is that the data exists but is flatline, i.e. a constant value throughout the time period considered. This will cause `Orient.py` to skip that event, even if just one component is corrupt, since the algorithm requires all three components to perform the measurements.

- The final orientation estimate make no sense

This may be the results of a number of situations. First, check to make sure you are using the right channel naming convention. Using the wrong convention is not simply a matter of rotating the results by 90°; the results can be entirely corrupted.
Additionally, this may be caused by insufficient data included in the final calculation. The bootstrap statistical treatment of the results attempts to guard against this effect, but a low number of data can still produce unrealistically high or low uncertainties.
Finally, the results could reflect a very noisy station where a well-constrained orientation estimate may be difficult. A different correlation threshold may produce more stable results. Occasionally measurements at a certain frequency may produce high correlation values but inconsistent arrival angles. Try excluding certain frequencies and see if a better result can be obtained. Very shallow (< 100m OBS sites) often do not produce enough high-quality long-period data to resolve the orientation using this technique.

Network	Station	Orientation	Uncertainty
XM	OSN1	126.72	1.86
XM	OSN1B	148.57	1.68
XM	OSN1S	281.46	3.64

Table 1. Orientation results for three OSN sites as detailed in *Doran and Laske* (2017). Orientation given as clockwise rotation from geographic north of BH1, the nominally north component (see Figure 1).

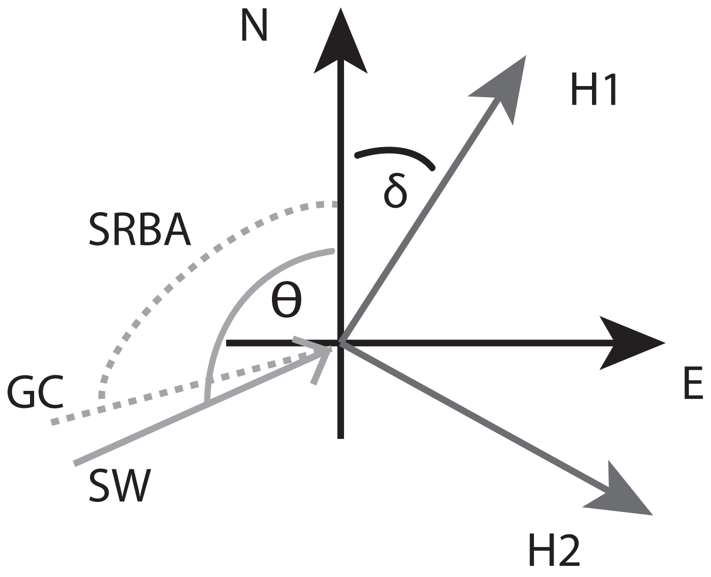

Figure 1. Orientation of the horizontal seismometer components (N, E) using the GSN/SEED naming convention for a left-handed coordinate system. Component Z emerges out of the page towards the reader. Misaligned components are named H1 and H2 accordingly. The angle δ is the instrument orientation. Angle $\alpha = \theta + \delta$ is determined by grid search. The measured arrival angle as used for further processing is α - SRBA (source-receiver back azimuth). In a heterogenous Earth, the actual approach of a surface wave (SW) may deviate from the source-receiver great-circle (GC) by several degrees. Figure taken from *Doran and Laske* (2017).

References

Beyreuther, M., R. Barsch, L. Krischer, T. Megies, Y. Behr, and J. Wassermann (2010), ObsPy: A Python toolbox for seismology, *Seismological Research Letters*, *81*(3), 530–533, doi: 10.1785/gssrl.81.3.530.

Collins, J. A., F. L. Vernon, J. A. Orcutt, R. A. Stephen, K. R. Peal, F. B. Wooding, F. N. Spiess, and J. A. Hildebrand (2001), Broadband seismology in the oceans: lessons from the Ocean Seismic Network Pilot Experiment, *Geophysical Research Letters*, *28*(1), 49–52.

Karney, C. F. F. (2013), Algorithms for geodesics, *Journal of Geodesy*, *87*(1), 43–55, doi: 10.1007/s00,190–012–0578–z.

Krischer, L., T. Megies, R. Barsch, M. Beyreuther, T. Lecocq, C. Caudron, and J. Wassermann (2015), ObsPy: a bridge for seismology into the scientific Python ecosystem, *Computational Science and Discovery*, *8*(1), 014,003, doi: 10.1088/1749–4699/8/1/014,003.

Ma, Z., and G. Masters (2014), A new global Rayleigh- and Love-wave group velocity dataset for constraining lithosphere properties, *Bulletin of the Seismological Society of America*, *104*(4), 2007–2026, doi: 10.1785/0120130,320.

Ma, Z., G. Masters, G. Laske, and M. Pasyanos (2014), A comprehensive dispersion model of surface wave phase and group velocity for the globe, *Geophysical Journal International*, *119*(1), 113–135, doi: 10.1093/gji/ggu246.

Megies, T., M. Beyreuther, R. Barsch, L. Krischer, and J. Wassermann (2011), ObsPy - what can it do for data centers and observatories?, *Annals of Geophysics*, *54*(1), 47–58, doi: 10.4401/ag–4838.

Stephen, R. A., F. N. Spiess, J. A. Collins, J. A. Hildebrand, J. A. Orcutt, K. R. Peal, F. L. Vernon, and F. B. Wooding (2003), Ocean Seismic Network Pilot Experiment, *Geochem. Geopyhs. Geosys.*, *4*(10), 1–38, doi: 10.1029/2002GC000,485.

Parameter	Description	Data Type	Options
`NET`	Instrument network	`str`	Example: II
`STA`	Instrument station name	`str`	Example: ESK
`CHA`	Instrument channels	`str`	Example: BH?
`LOC`	Instrument location code	`str`	Example: 00
`localdata`	Analyze local data instead of data stored at repository	`int`	0 - use data from repository (specified by `wf_client`) 1 - use local data (see Section)
`localcoords`	Manual input for Lat and Lon. Can be used with `localdata`, or if IRIS DMC is missing or inaccurate.	`int`	0 - use coordinates provided with station info 1 - use locally input coordinates
`inputlat`	Station latitude manual override. Must be in Degree decimal.	`float`	Example: 32.533617
`inputlat`	Station latitude manual override. Must be in Degree decimal.	`float`	Example: -120.49978
`time1`	Start time for calculation	`str`	Time in format of either: "YYYY-MM-DD HH:MM:SS" "YYYY-JDAY HH:MM:SS"
`time2`	End time for calculation	`str`	Same as `time1`
`cat_client`	Earthquake catalog to use	`str`	Examples: "IRIS", "SCEDC", "NCEDC", etc
`wf_client`	Waveform client to use	`str`	Same options as `cat_client`
`nameconv`	Instrument component organization	`int`	1 - HN / HE/ HZ (left-handed convention) 2 - H1 / H2 / HZ (left-handed convention) 3 - H2 / H1 / HZ (right-handed convention)
`minmag`	Minimum M_W events to analyze	`float`	Default: $M_{min} = 5.5$
`mindeg`	Minimum epicentral degree distance Δ	`float`	Default: $\Delta_{min} = 5.0°$
`maxdeg`	Maximum epicentral degree distance Δ	`float`	Default: $\Delta_{max} = 175.0°$
`maxdep`	Maximum event depth	`float`	Default: $z_{max} = 150$km
`verb`	Defines the level of text output during while the program is running	`int`	0 - Outputs nothing 1 - Outputs # event being analyzed 2 - Outputs # event being analyzed + R1-30mHz result
`constsave`	Whether to continuously save data after each event	`int`	0 (no) or 1 (yes)
`saveloc`	Location of saved data	`str`	
`savecat`	Whether to save EQ catalog	`int`	0 (no) or 1 (yes)
`catname`	Location of saved EQ catalog	`str`	

TABLE 2. Parameter file (`Orient_PF`) options

Parameter	Description	Data Type	Options
`loc1`	Location of arrival angle file	`str`	e.g., 'OSN1B/'
`LIM`	Cross-correlation threshold (only calculations with CC values higher than `LIM` are considered	`float`	Any decimal between 0 and 1, should remain at 0.8
`R1_40`	Whether or not to use this orbit and frequency results in the final calculation. The same applies to each other combination of orbit and frequency (e.g., `R2_20`)	`int`	0 (no) or 1 (yes)

Table 3. Final calculation (`compcalcs`) parameter options