Pysmo Documentation

Release 0.8.3

Pysmo Project

Aug 22, 2023

Contents:

CHAPTER 1

Introduction

The pysmo package provides some simple to use tools for seismologists.

It's core components are an interface to the commonly used SAC file format, functions that work with this interface, and some additional useful tools.

CHAPTER 2

Installation and Upgrades

2.1 Requirements

2.1.1 Python

Pysmo is built on top of standard Python and uses several extra Python libraries as well. We develop and test pysmo on Python versions 3.6 and newer. Pysmo may run on older versions too, but we strongly suggest upgrading Python should you be running an older version (not just for pysmo!).

2.1.2 Compilers

C

While pysmo itself does not use C, some of the Python libraries it uses potentially require a C compiler during installation. This may vary depending on computer platform, or Python version/distribution used.

2.1.3 Operating System

Pysmo is designed to run on UNIX like systems (e.g. Mac OSX and Linux). Installation on Windows is probably possible (since Python can be installed on almost any platform), though untested.

2.2 Installing Pysmo

2.2.1 pip - Python package installer

Pysmo is available as a package from the Python Package Index. This means it can be easily installed with the `pip` command (available by default since Python version 3.4).

Caution: It is possible to have multiple versions of Python installed on a computer. If this is the case, then there will also be multiple versions of the `pip` command. It is therefore important to use the `pip` command belonging to the Python version you intend to use for pysmo! Running `pip --version` will show you which Python version it belongs to.

Note: On some systems Python 2 and Python 3 are installed alongside eachother. Typically there is a `pip` command belonging to Python 2 and a `pip3` command belonging to Python 3.

2.2.2 conda users

If you are using conda to manage Python packages, we recommend installing pysmo dependencies with `conda` before installing pysmo with `pip`. To do so issue this command:

```
$ conda install scipy numpy matplotlib pyyaml pyproj
```

Note: Similarly, if you are using a package manager on Linux, or something like brew or macports on OSX, you may want to install these dependencies (if available) via those mechanisms instead of `pip`.

2.2.3 Installing pysmo with pip

To install the latest stable version of pysmo and all dependencies not already installed, simply issue this command:

```
$ pip install pysmo
```

Caution: Unless you know what you are doing, we recommend to *not* install pysmo with administrative priveliges (i.e. using sudo or the root account). If the above command fails due to insuffienct rights, run the same command with the `--user` flag:

```
$ pip install --user pysmo
```

If you wish to install the latest developement version of pysmo *instead* of the stable release:

```
$ pip install git+https://github.com/pysmo/pysmo
```

Note: It is possible to install the stable release alongside the development version. Please read *Developing pysmo* for instructions.

2.2.4 Example Data

Get the repository data-example from Github. There is some example code inside *data-example/example_pkl_files* that will be needed for later demonstrations.

2.3 Upgrading pysmo

Upgrading pysmo with `pip` is done with the same command used to install, with the addition of the `-U` flag:

```
$ pip install -U pysmo
```

Note: If you used the `--user` flag during installation you also need to use it while upgrading

2.4 Uninstalling pysmo

To remove pysmo from your system with `pip` run this command:

```
$ pip uninstall pysmo
```

Note: Unfortunately `pip` currently does not remove dependencies that were automatically installed. We suggest running `pip list` to see the installed packages, which can then also be removed using `pip uninstall`

CHAPTER 3

Working with SAC files

The psymo.core.sac package consists of two modules, which provide:

1. The python class *SacIO*, to read and write SAC files.
2. Functions which operate on the *SacIO* class.

class pysmo.core.sac.**SacIO**(***kwargs*)

Bases: object

The *SacIO* class reads and writes data and header values to and from a SAC file. Additonal class attributes may be set, but are not written to a SAC file (because there is no space reserved for them there). Class attributes with corresponding header fields in a SAC file (for example the begin time *b*) are checked for a valid format before being saved in the *SacIO* instance.

Read and print data:

```
>>> from pysmo import SacIO
>>> my_sac = SacIO.from_file('file.sac')
>>> data = my_sac.data
>>> data
[-1616.0, -1609.0, -1568.0, -1606.0, -1615.0, -1565.0, ...
```

Read the sampling rate:

```
>>> delta = my_sac.delta
>>> delta
0.019999999552965164
```

Change the sampling rate:

```
>>> newdelta = 0.05
>>> my_sac.delta = newdelta
>>> my_sac.delta
0.05
```

Read from IRIS services:

```
>>> from pysmo import SacIO
>>> my_sac = SacIO.from_iris(
>>>             net="C1",
>>>             sta="VA01",
>>>             cha="BHZ",
>>>             loc="--",
>>>             start="2021-03-22T13:00:00",
>>>             duration=1 * 60 * 60,
>>>             scale="AUTO",
>>>             demean="true",
>>>             force_single_result=True)
>>> my_sac.npts
144001
```

There are a lot of header fields in a SAC file, which are all called the same way when using *SacIO*. They are all listed below.

a
: First arrival time (seconds relative to reference time.)

az
: Event to station azimuth (degrees).

b
: Beginning value of the independent variable.

baz
: Station to event azimuth (degrees).

cmpaz
: Component azimuth (degrees clockwise from north).

cmpinc
: Component incident angle (degrees from vertical).

data

First data section:

- dependent variable
- amplitude
- real component

Second data section (if it exists):

- independent variable unevenly spaced
- phase
- imaginary component

If there is only one data section, it is returned as a list of floats. Two data sections result in returning a list of tuples.

delta
: Increment between evenly spaced samples (nominal value).

depmax
: Maximum value of dependent variable.

depmen
: Mean value of dependent variable.

depmin
Minimum value of dependent variable.

dist
Station to event distance (km).

e
Ending value of the independent variable.

evdp
Event depth below surface (kilometers – previously meters)

evel
Event elevation (meters). [not currently used]

evla
Event latitude (degrees, north positive).

evlo
Event longitude (degrees, east positive).

f
Fini or end of event time (seconds relative to reference time.)

fmt
Internal

classmethod from_buffer(*buffer*)
Create a new SacIO instance from a SAC data buffer.

classmethod from_file(*filename*)
Create a new SacIO instance from a SAC file.

classmethod from_iris(*net*, *sta*, *cha*, *loc*, *force_single_result=False*, ***kwargs*)
Create a list of SacIO instances from a single IRIS request using the output format as "sac.zip".

Parameters **force_single_result** (*bool*) – If true, the function will return a single SacIO object or None if the requests returns nothing.

gcarc
Station to event great circle arc length (degrees).

idep
Type of dependent variable

ievreg
Event geographic region. [not currently used]

ievtyp
Type of event

iftype
Type of file

iinst
Type of recording instrument. [not currently used]

imagsrc
Source of magnitude information

imagtyp
Magnitude type

iqual
Quality of data [not currently used]

istreg
Station geographic region. [not currently used]

isynth
Synthetic data flag [not currently used]

iztype
Reference time equivalence

ka
First arrival time identification.

kcmpnm
Channel name. SEED volumes use three character names, and the third is the component/orientation. For horizontals, the current trend is to use 1 and 2 instead of N and E.

kdatrd
Date data was read onto computer.

kevnm
Event name.

kf
Fini identification.

khole
Nuclear: hole identifier; Other: location identifier.

kinst
Generic name of recording instrument.

knetwk
Name of seismic network.

ko
Event origin time identification.

kstnm
Station name.

kt0
User defined time pick identification.

kt1
User defined time pick identification.

kt2
User defined time pick identification.

kt3
User defined time pick identification.

kt4
User defined time pick identification.

kt5
User defined time pick identification.

kt6
User defined time pick identification.

kt7
User defined time pick identification.

kt8
User defined time pick identification.

kt9
User defined time pick identification.

kuser0
User defined variable storage area.

kuser1
User defined variable storage area.

kuser2
User defined variable storage area.

kzdate
ISO 8601 format of GMT reference date.

kztime
Alphanumeric form of GMT reference time.

lcalda
TRUE if DIST, AZ, BAZ, and GCARC are to be calculated from station and event coordinates.

leven
TRUE if data is evenly spaced

lovrok
TRUE if it is okay to overwrite this file on disk.

lpspol
TRUE if station components have a positive polarity (left-hand rule).

mag
Event magnitude.

nevid
Event ID (CSS 3.0)

norid
Origin ID (CSS 3.0)

npts
Number of points per data component

nsnpts
Internal.

nvhdr
Header version number.

nwfid
Waveform ID (CSS 3.0)

nxsize
Spectral Length (Spectral files only)

nysize
Spectral Length (Spectral files only)

nzhour
GMT hour.

nzjday
GMT julian day.

nzmin
GMT minute.

nzmsec
GMT millisecond.

nzsec
GMT second.

nzyear
GMT year corresponding to reference (zero) time in file.

o
Event origin time (seconds relative to reference time.)

odelta
Observed increment if different from nominal value.

read (*filename*)
Read data and header values from a SAC file into an existing SacIO instance.

read_buffer (*buffer*)
Read data and header values from a SAC byte buffer into an existing SacIO instance.

resp0
Instrument response parameter 0 (not currently used)

resp1
Instrument response parameter 1 (not currently used)

resp2
Instrument response parameter 2 (not currently used)

resp3
Instrument response parameter 3 (not currently used)

resp4
Instrument response parameter 4 (not currently used)

resp5
Instrument response parameter 5 (not currently used)

resp6
Instrument response parameter 6 (not currently used)

resp7
Instrument response parameter 7 (not currently used)

resp8
Instrument response parameter 8 (not currently used)

resp9
Instrument response parameter 9 (not currently used)

sb
Internal.

scale
Multiplying scale factor for dependent variable (not currently used)

sdelta
Internal.

stdp
Station depth below surface (meters). (not currently used)

stel
Station elevation above sea level (meters). (not currently used)

stla
Station latitude (degrees, north positive)

stlo
Station longitude (degrees, east positive).

t0
User defined time pick or marker 0 (seconds relative to reference time).

t1
User defined time pick or marker 1 (seconds relative to reference time).

t2
User defined time pick or marker 2 (seconds relative to reference time).

t3
User defined time pick or marker 3 (seconds relative to reference time).

t4
User defined time pick or marker 4 (seconds relative to reference time).

t5
User defined time pick or marker 5 (seconds relative to reference time).

t6
User defined time pick or marker 6 (seconds relative to reference time).

t7
User defined time pick or marker 7 (seconds relative to reference time).

t8
User defined time pick or marker 8 (seconds relative to reference time).

t9
User defined time pick or marker 9 (seconds relative to reference time).

unused10
Unused.

unused11
Unused.

unused12
Unused.

unused15
Unused.

unused16
Unused.

unused19
Unused

unused20
Unused

unused21
Unused

unused22
Unused

unused23
Unused

unused24
Unused

unused25
Unused

unused26
Unused

unused27
Unused

unused6
Unused.

unused7
Unused.

unused8
Unused.

unused9
Unused.

user0
User defined variable storage area

user1
User defined variable storage area

user2
User defined variable storage area

user3
User defined variable storage area

user4
User defined variable storage area

user5
User defined variable storage area

user6
User defined variable storage area

user7
User defined variable storage area

user8
User defined variable storage area

user9
User defined variable storage area

write (*filename*)
Write data and header values to a SAC file

xmaximum
Maximum value of X (Spectral files only)

xminimum
Minimum value of X (Spectral files only)

ymaximum
Maximum value of Y (Spectral files only)

yminimum
Minimum value of Y (Spectral files only)

3.1 pysmo.SacIO

Python module for reading/writing SAC files using the *SacIO* class.

class pysmo.core.sac.sacio.**SacIO**(***kwargs*)
Bases: object

The *SacIO* class reads and writes data and header values to and from a SAC file. Additonal class attributes may be set, but are not written to a SAC file (because there is no space reserved for them there). Class attributes with corresponding header fields in a SAC file (for example the begin time *b*) are checked for a valid format before being saved in the *SacIO* instance.

Read and print data:

```
>>> from pysmo import SacIO
>>> my_sac = SacIO.from_file('file.sac')
>>> data = my_sac.data
>>> data
[-1616.0, -1609.0, -1568.0, -1606.0, -1615.0, -1565.0, ...
```

Read the sampling rate:

```
>>> delta = my_sac.delta
>>> delta
0.019999999552965164
```

Change the sampling rate:

```
>>> newdelta = 0.05
>>> my_sac.delta = newdelta
>>> my_sac.delta
0.05
```

Read from IRIS services:

```
>>> from pysmo import SacIO
>>> my_sac = SacIO.from_iris(
>>>             net="C1",
>>>             sta="VA01",
>>>             cha="BHZ",
>>>             loc="--",
>>>             start="2021-03-22T13:00:00",
>>>             duration=1 * 60 * 60,
>>>             scale="AUTO",
>>>             demean="true",
>>>             force_single_result=True)
>>> my_sac.npts
144001
```

There are a lot of header fields in a SAC file, which are all called the same way when using *SacIO*. They are all listed below.

a
: First arrival time (seconds relative to reference time.)

az
: Event to station azimuth (degrees).

b
: Beginning value of the independent variable.

baz
: Station to event azimuth (degrees).

cmpaz
: Component azimuth (degrees clockwise from north).

cmpinc
: Component incident angle (degrees from vertical).

data

First data section:

- dependent variable
- amplitude
- real component

Second data section (if it exists):

- independent variable unevenly spaced
- phase
- imaginary component

If there is only one data section, it is returned as a list of floats. Two data sections result in returning a list of tuples.

delta
: Increment between evenly spaced samples (nominal value).

depmax
: Maximum value of dependent variable.

depmen
: Mean value of dependent variable.

depmin
Minimum value of dependent variable.

dist
Station to event distance (km).

e
Ending value of the independent variable.

evdp
Event depth below surface (kilometers – previously meters)

evel
Event elevation (meters). [not currently used]

evla
Event latitude (degrees, north positive).

evlo
Event longitude (degrees, east positive).

f
Fini or end of event time (seconds relative to reference time.)

fmt
Internal

classmethod from_buffer(*buffer*)
Create a new SacIO instance from a SAC data buffer.

classmethod from_file(*filename*)
Create a new SacIO instance from a SAC file.

classmethod from_iris(*net*, *sta*, *cha*, *loc*, *force_single_result=False*, ***kwargs*)
Create a list of SacIO instances from a single IRIS request using the output format as "sac.zip".

Parameters **force_single_result** (*bool*) – If true, the function will return a single SacIO object or None if the requests returns nothing.

gcarc
Station to event great circle arc length (degrees).

idep
Type of dependent variable

ievreg
Event geographic region. [not currently used]

ievtyp
Type of event

iftype
Type of file

iinst
Type of recording instrument. [not currently used]

imagsrc
Source of magnitude information

imagtyp
Magnitude type

iqual
Quality of data [not currently used]

istreg
Station geographic region. [not currently used]

isynth
Synthetic data flag [not currently used]

iztype
Reference time equivalence

ka
First arrival time identification.

kcmpnm
Channel name. SEED volumes use three character names, and the third is the component/orientation. For horizontals, the current trend is to use 1 and 2 instead of N and E.

kdatrd
Date data was read onto computer.

kevnm
Event name.

kf
Fini identification.

khole
Nuclear: hole identifier; Other: location identifier.

kinst
Generic name of recording instrument.

knetwk
Name of seismic network.

ko
Event origin time identification.

kstnm
Station name.

kt0
User defined time pick identification.

kt1
User defined time pick identification.

kt2
User defined time pick identification.

kt3
User defined time pick identification.

kt4
User defined time pick identification.

kt5
User defined time pick identification.

kt6
User defined time pick identification.

kt7
User defined time pick identification.

kt8
User defined time pick identification.

kt9
User defined time pick identification.

kuser0
User defined variable storage area.

kuser1
User defined variable storage area.

kuser2
User defined variable storage area.

kzdate
ISO 8601 format of GMT reference date.

kztime
Alphanumeric form of GMT reference time.

lcalda
TRUE if DIST, AZ, BAZ, and GCARC are to be calculated from station and event coordinates.

leven
TRUE if data is evenly spaced

lovrok
TRUE if it is okay to overwrite this file on disk.

lpspol
TRUE if station components have a positive polarity (left-hand rule).

mag
Event magnitude.

nevid
Event ID (CSS 3.0)

norid
Origin ID (CSS 3.0)

npts
Number of points per data component

nsnpts
Internal.

nvhdr
Header version number.

nwfid
Waveform ID (CSS 3.0)

nxsize
Spectral Length (Spectral files only)

nysize
Spectral Length (Spectral files only)

nzhour
GMT hour.

nzjday
GMT julian day.

nzmin
GMT minute.

nzmsec
GMT millisecond.

nzsec
GMT second.

nzyear
GMT year corresponding to reference (zero) time in file.

o
Event origin time (seconds relative to reference time.)

odelta
Observed increment if different from nominal value.

read (*filename*)
Read data and header values from a SAC file into an existing SacIO instance.

read_buffer (*buffer*)
Read data and header values from a SAC byte buffer into an existing SacIO instance.

resp0
Instrument response parameter 0 (not currently used)

resp1
Instrument response parameter 1 (not currently used)

resp2
Instrument response parameter 2 (not currently used)

resp3
Instrument response parameter 3 (not currently used)

resp4
Instrument response parameter 4 (not currently used)

resp5
Instrument response parameter 5 (not currently used)

resp6
Instrument response parameter 6 (not currently used)

resp7
Instrument response parameter 7 (not currently used)

resp8
Instrument response parameter 8 (not currently used)

resp9
Instrument response parameter 9 (not currently used)

sb
Internal.

scale
Multiplying scale factor for dependent variable (not currently used)

sdelta
Internal.

stdp
Station depth below surface (meters). (not currently used)

stel
Station elevation above sea level (meters). (not currently used)

stla
Station latitude (degrees, north positive)

stlo
Station longitude (degrees, east positive).

t0
User defined time pick or marker 0 (seconds relative to reference time).

t1
User defined time pick or marker 1 (seconds relative to reference time).

t2
User defined time pick or marker 2 (seconds relative to reference time).

t3
User defined time pick or marker 3 (seconds relative to reference time).

t4
User defined time pick or marker 4 (seconds relative to reference time).

t5
User defined time pick or marker 5 (seconds relative to reference time).

t6
User defined time pick or marker 6 (seconds relative to reference time).

t7
User defined time pick or marker 7 (seconds relative to reference time).

t8
User defined time pick or marker 8 (seconds relative to reference time).

t9
User defined time pick or marker 9 (seconds relative to reference time).

unused10
Unused.

unused11
Unused.

unused12
Unused.

unused15
Unused.

unused16
Unused.

unused19
Unused

unused20
Unused

unused21
Unused

unused22
Unused

unused23
Unused

unused24
Unused

unused25
Unused

unused26
Unused

unused27
Unused

unused6
Unused.

unused7
Unused.

unused8
Unused.

unused9
Unused.

user0
User defined variable storage area

user1
User defined variable storage area

user2
User defined variable storage area

user3
User defined variable storage area

user4
User defined variable storage area

user5
User defined variable storage area

user6
User defined variable storage area

user7
User defined variable storage area

user8
User defined variable storage area

user9
User defined variable storage area

write (*filename*)
Write data and header values to a SAC file

xmaximum
Maximum value of X (Spectral files only)

xminimum
Minimum value of X (Spectral files only)

ymaximum
Maximum value of Y (Spectral files only)

yminimum
Minimum value of Y (Spectral files only)

3.2 pysmo.sacfunc

Useful functions to use with SacIO objects.

pysmo.core.sac.sacfunc.**calc_az** (*name*, *ellps='WGS84'*)
Calculate azimuth (in DEG) from a SacIO object. The default ellipse used is 'WGS84', but others may be specified. For more information see:

http://trac.osgeo.org/proj/ http://code.google.com/p/pyproj/

Parameters

- **name** (SacIO) – Name of the SacIO object passed to this function.
- **ellps** (*string*) – Ellipsoid to use for azimuth calculation

Returns Azimuth

Return type float

Example usage:

```
>>> from pysmo import SacIO, sacfunc
>>> sacobj = SacIO.from_file('sacfile.sac')
>>> azimuth = sacfunc.calc_az(sacobj) # Use default WGS84.
>>> azimuth = sacfunc.calc_az(sacobj, ellps='clrk66') # Use Clarke 1966
```

pysmo.core.sac.sacfunc.**calc_baz** (*name*, *ellps='WGS84'*)
Calculate backazimuth (in DEG) from a SacIO object. The default ellipse used is 'WGS84', but others may be specified. For more information see:

http://trac.osgeo.org/proj/ http://code.google.com/p/pyproj/

Parameters

- **name** (SacIO) – Name of the SacIO object passed to this function.
- **ellps** (*string*) – Ellipsoid to use for backazimuth calculation

Returns Backazimuth

Return type float

Example usage:

```
>>> from pysmo import SacIO, sacfunc
>>> sacobj = SacIO.from_file('sacfile.sac')
>>> backazimuth = sacfunc.calc_baz(sacobj) # Use default WGS84.
>>> backazimuth = sacfunc.calc_baz(sacobj, ellps='clrk66') # Use Clarke 1966␣
↪ellipsoid.
```

pysmo.core.sac.sacfunc.**calc_dist**(*name*, *ellps='WGS84'*)

Calculate the great circle distance (in km) from a SacIO object. The default ellipse used is 'WGS84', but others may be specified. For more information see:

http://trac.osgeo.org/proj/ http://code.google.com/p/pyproj/

Parameters

- **name** (SacIO) – Name of the SacIO object passed to this function.
- **ellps** (*string*) – Ellipsoid to use for distance calculation

Returns Great Circle Distance

Return type float

Example usage:

```
>>> from pysmo import SacIO, sacfunc
>>> sacobj = SacIO.from_file('sacfile.sac')
>>> distance = sacfunc.calc_dist(sacobj) # Use default WGS84.
>>> distance = sacfunc.calc_dist(sacobj, ellps='clrk66') # Use Clarke 1966␣
↪ellipsoid.
```

pysmo.core.sac.sacfunc.**detrend**(*name*)

Remove linear and/or constant trends from SacIO object data.

Parameters **name** (SacIO) – Name of the SacIO object passed to this function.

Returns Detrended data.

Return type numpy.array

Example usage:

```
>>> from pysmo import SacIO, sacfunc
>>> sacobj = SacIO.from_file('sacfile.sac')
>>> detrended_data = sacfunc.detrend(sacobj)
```

pysmo.core.sac.sacfunc.**envelope**(*name*, *Tn*, *alpha*)

Calculate the envelope of a gaussian filtered seismogram.

Parameters

- **name** (SacIO) – Name of the SacIO object passed to this function.
- **Tn** (*float*) – Center period of Gaussian filter [in seconds]
- **alpha** (*float*) – Set alpha (which determines filterwidth)

Returns numpy array containing the envelope

Example:

```
>>> from pysmo import SacIO, sacfunc
>>> sacobj = SacIO.from_file('sacfile.sac')
>>> Tn = 50 # Center Gaussian filter at 50s period
>>> alpha = 50 # Set alpha (which determines filterwidth) to 50
>>> envelope = sacfunc.envelope(sacobj, Tn, alpha)
```

pysmo.core.sac.sacfunc.**gauss**(*name*, *Tn*, *alpha*)

Return data vector of a gaussian filtered seismogram.

Parameters

- **name** (SacIO) – Name of the SacIO object passed to this function.
- **Tn** (*float*) – Center period of Gaussian filter [in seconds]
- **alpha** (*float*) – Set alpha (which determines filterwidth)

Returns numpy array containing filtered data

Example usage:

```
>>> from pysmo import SacIO, sacfunc
>>> sacobj = SacIO.from_file('sacfile.sac')
>>> Tn = 50 # Center Gaussian filter at 50s period
>>> alpha = 50 # Set alpha (which determines filterwidth) to 50
>>> data = sacfunc.gauss(sacobj, Tn, alpha)
```

pysmo.core.sac.sacfunc.**plotsac**(*name*, *outfile=None*, *showfig=True*)

Simple plot of a single sac file.

Parameters

- **name** (SacIO) – Name of the SacIO object passed to this function.
- **outfile** (*bool*) – If specified, save figure to this file.
- **showfig** – Specifies if figure should be displayed.

Example usage:

```
>>> from pysmo import SacIO, sacfunc
>>> sacobj = SacIO.from_file('sacfile.sac')
>>> sacfunc.plotsac(sacobj)
```

pysmo.core.sac.sacfunc.**resample**(*name*, *delta_new*)

Resample SacIO object data using Fourier method.

Parameters

- **name** (SacIO) – Name of the SacIO object passed to this function.
- **delta_new** – New sampling rate.

Returns Resampled data.

Return type numpy.array

Example usage:

```
>>> from pysmo import SacIO, sacfunc
>>> sacobj = SacIO.from_file('sacfile.sac')
>>> delta_old = sacobj.delta
>>> delta_new = delta_old * 2
>>> data_new = sacfunc.resample(sac, delta_new)
```

`pysmo.core.sac.sacfunc.`**`sac2xy`**(*name*, *retarray=False*)

Return time and amplitude from a SacIO object.

Parameters

- **`name`** (`SacIO`) – Name of the SacIO object passed to this function.
- **`retarray`** (*`bool`*) –
  - True: return numpy array
  - False: return list (default)

Returns Time and amplitude of a SacIO object.

Return type numpy.array or list

Example usage:

```
>>> from pysmo import SacIO, sacfunc
>>> sacobj = SacIO.from_file('sacfile.sac')
>>> time, vals = sacfunc.sac2xy(sacobj, retarray=True)
```

CHAPTER 4

Tools

The psymo.tools package provides some extra tools that do not involve usage of SAC files.

4.1 Subpackages

4.1.1 pysmo.tools.noise package

This module provides support for calculating random synthetic noise that matches the naturally observed amplitude spectrum.

See Also:

Peterson, J., 1993. Observations and modelling of background seismic noise. Open-file report 93-322, U. S. Geological Survey, Albuquerque, New Mexico.

pysmo.tools.noise.**genNoiseNHNM**(*delta*, *npts*, *velocity=False*)

Generate a random signal that matches Peterson's new high noise model NHNM.

Parameters

- **delta** (*float*) – Sampling rate of generated noise
- **npts** (*int*) – Number of points of generated noise
- **velocity** (*bool*) – Return velocity instead of acceleration.

Returns numpy array containing synthetic data

Example usage:

```
>>> import pysmo.tools.noise as noise
>>> delta = 0.05
>>> npts = 5000
>>> high_noise = noise.genNoiseNHNM(delta, npts)
```

pysmo.tools.noise.**genNoiseNLNM**(*delta*, *npts*, *velocity=False*)

Generate a random signal that matches Peterson's new low noise model NLNM.

Parameters

- **delta** (*float*) – Sampling rate of generated noise
- **npts** (*int*) – Number of points of generated noise
- **velocity** (*bool*) – Return velocity instead of acceleration.

Returns numpy array containing synthetic data

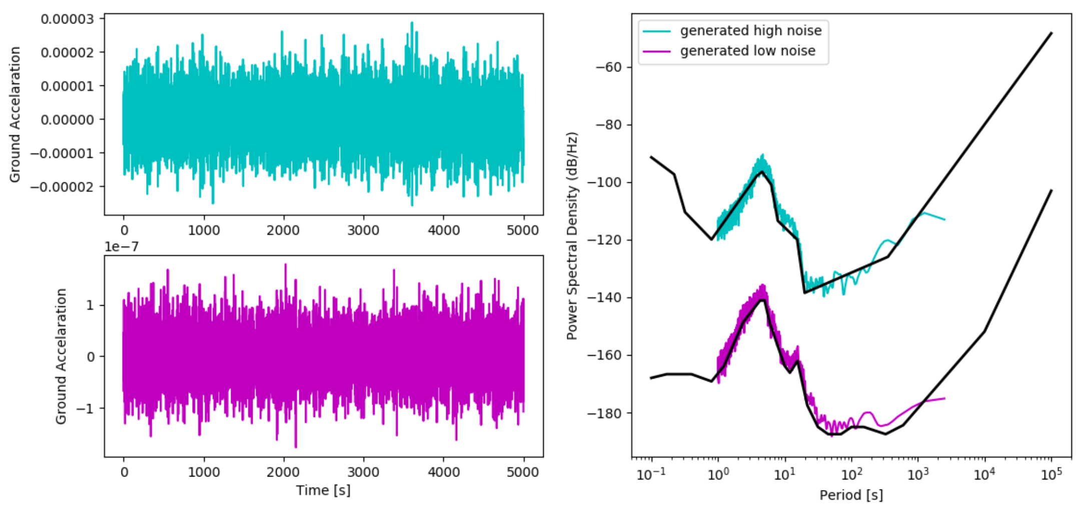
Example usage:

```
>>> import pysmo.tools.noise as noise
>>> delta = 0.05
>>> npts = 5000
>>> low_noise = noise.genNoiseNLNM(delta, npts)
```

4.2 Code Examples

4.2.1 Noise

In this example two realistic random noise signals are generated and their PSDs are calculated.




```
#!/usr/bin/env python

# Example script for pysmo.tools.noise
import pysmo.tools.noise as noise
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal

npts = 10000
delta = 0.5
```

(continues on next page)

(continued from previous page)

```
fs = 1 / delta
nperseg = npts/4
nfft = npts/2
time = np.linspace(0, npts*delta, npts)

# Generate random noise
high_noise = noise.genNoiseNHNM(delta, npts)
low_noise = noise.genNoiseNLNM(delta, npts)

# Calculuate power spectral density
f_high, Pxx_dens_high = signal.welch(high_noise, fs, nperseg=nperseg,
                                     nfft=nfft, scaling='density')
f_low, Pxx_dens_low = signal.welch(low_noise, fs, nperseg=nperseg,
                                   nfft=nfft, scaling='density')

fig = plt.figure(figsize=(13, 6))
# Plot random high and low noise
plt.subplot(221)
plt.plot(time, high_noise, 'c')
plt.ylabel('Ground Accelaration')

plt.subplot(223)
plt.plot(time, low_noise, 'm')
plt.ylabel('Ground Accelaration')
plt.xlabel('Time [s]')

# Plot PSD of noise
plt.subplot(122)
plt.plot(1/f_high, 10*np.log10(Pxx_dens_high), 'c',
         label='generated high noise')
plt.plot(1/f_low, 10*np.log10(Pxx_dens_low), 'm', label='generated low noise')
plt.plot(noise.NLNM['T'], noise.NLNM['dB'], 'k', linewidth=2)
plt.plot(noise.NHNM['T'], noise.NHNM['dB'], 'k', linewidth=2)
plt.gca().set_xscale("log")
plt.xlabel('Period [s]')
plt.ylabel('Power Spectral Density (dB/Hz)')
plt.legend()
plt.savefig('peterson.png')
plt.show()
```

CHAPTER 5

Contributing

We welcome all bug reports and feature suggestions to pysmo. Please submit these on our GitHub issue page. If you are submitting a bug report, please also include information about your environment (operating system, Python version etc.).

If you are also able to contribute code we will be even more delighted! Please keep the following in mind when submitting a pull request on GitHub:

- Adhere to the PEP 8 style guide.
- If applicable, provide a test script using pytest or the built in unittest module.
- Provide docstrings inside your Python code that can be used for this documentation. Please also provide references if applicable.

CHAPTER 6

Developing pysmo

6.1 Github

We use Github for the development of pysmo. Github is easiest to use with authentication using SSH keys. Please refer to the Github documentation for setup instructions on your platform.

Once your account is properly set up you can copy the pysmo repository to the your workstation with the `git clone` command:

```
$ git clone git@github.com:pysmo/pysmo.git
```

You can now navigate to the `pysmo` directory and explore the files:

```
$ cd pysmo
$ ls
build docs ...
```

Note: If you are not a member of the pysmo group on Github you will have to first fork the pysmo repository (from the GUI on Github).

6.2 Poetry

In order to develop pysmo in a consistent and isolated environment we use Poetry. Poetry creates a Python virtual environment and manages the Python packages that are installed in that environment. This allows developing and testing while also having the stable version of pysmo installed at the same time. Please see the Poetry documentation for installation and basic usage instructions. Once Poetry is installed its commands can be viewed by running:

```
$ poetry --help
```

Caution: Please note that poetry only creates virtual environments for Python - these are not comparable to a virtual machine and do not offer the same separation!

Note: For convenience we wrap the most used poetry commands in a `Makefile`.

6.3 Makefile

Pysmo provides a `Makefile` to aid with setting up and using a development environment. To list the available make commands run:

```
$ make help
```

The most commonly used ones will likely be:

```
$ make install
```

to create a Python virtual environment for development of pysmo (unless it already exists), then install pysmo and its dependencies,

```
$ make tests
```

to run the test scripts from the tests directory, and

```
$ make shell
```

which starts a new new shell in the virtual environment. This shell uses the pysmo virtual Python environment instead of the default one. In other words, executing Python will use the development version of pysmo (i.e. the source files in the working directory).

Note: Within the virtual environment, any changes made to the the pysmo source files are used without having to re-issue `make install`.

6.4 Git workflows

There are different ways to use and achieve the same result with git. The guidelines here are aimed at developers not yet familiar with how git is best used to work as a team with other people.

Please avoid committing changes to the master branch directly. Even for small changes it is good practice to create a branch, and then either merge the changes in this branch into the master branch directly or via a pull request on Github (preferred). It is also generally a good idea to push and pull changes frequently, so that the local working copy on a workstation does not diverge too far from the *origin*, which is the version on Github.

6.4.1 Create a new branch

To pull the latest master revision from Github and create and switch to a new branch on your local workstation in one step:

```
$ git pull
$ git checkout -b <newbranch>
```

Next make changes to the pysmo code. In order to get the changes into the branch, first stage them with `git add` and then commit them with `git commit`:

```
$ git add <MyChangedFile>
$ git commit
```

Note: `git status` will not only show you the status of your branch, it will also provide you with useful hints.

The final step is to upload your changes to Github, where a pull request can be created in the GUI to have these changes merged into the master branch:

```
$ git push --set-upstream origin
```

If you now longer need the working branch you can switch back to master and delete the working branch:

```
$ git checkout master
$ git branch -d <newbranch>
```

6.4.2 Use an existing branch

When working on the same feature or bug as another developer, you will likely also be working on the same branch. As with most git operations, you first pull the latest changes to your workstation first. Then you switch to the branch you want to work on:

```
$ git pull
$ git checkout <existingbranch>
```

Just like with a new branch, you must first stage and then commit your changes:

```
$ git add <MyChangedFile>
$ git commit
```

Since the branch already exists on Github, the `push` command is a bit simpler:

```
$ git push
```

Again you can delete the working branch if you don't need it anymore (and if you do, you can always check it out again):

```
$ git checkout master
$ git branch -d <newbranch>
```

6.5 Unit testing

Unit tests execute a piece of code (a unit) and compare the output of that execution with a known reference value. Hence if changes to the pysmo code accidentally break functionality, unit tests are able to detect these before they are commited to the master branch (or worse to a stable release!). The unit tests in pysmo use the pytest framework.

6.5.1 Running the Tests

To run all the tests in one go from the root directory of the pysmo repository:

```
$ make test
```

Individual test scripts may also be specified:

```
$ poetry run py.test -mpl -v tests/<test_script>.py
```

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

K

L

M

N

O

P

R

S

T

U

W

X

Y