pyblockseis

Release 0.1.dev

Ana Aguiar Moya and Andrea Chiang

May 18, 2021

CONTENTS

1 Getting started **3**
- 1.1 Requirements . . . 3
- 1.2 Installation . . . 3
- 1.3 Usage . . . 3
- 1.4 Acknowledgements . . . 3
- 1.5 License . . . 3

2 API Reference **5**
- 2.1 Core Modules . . . 5
  - 2.1.1 pyblockseis.block . . . 5
    - 2.1.1.1 pyblockseis.block.read . . . 5
    - 2.1.1.2 pyblockseis.block.Block . . . 6
    - 2.1.1.3 pyblockseis.block.Parameter . . . 10
  - 2.1.2 pyblockseis.waveforms . . . 11
    - 2.1.2.1 pyblockseis.waveforms.Waveforms . . . 11
  - 2.1.3 pyblockseis.wavelet . . . 12
    - 2.1.3.1 pyblockseis.wavelet.blockbandpass . . . 12
    - 2.1.3.2 pyblockseis.wavelet.cwt . . . 13
    - 2.1.3.3 pyblockseis.wavelet.inverse_cwt . . . 13
    - 2.1.3.4 pyblockseis.wavelet.wfiltfn . . . 13
    - 2.1.3.5 pyblockseis.wavelet.wfilth . . . 14
    - 2.1.3.6 pyblockseis.wavelet.Wavelet . . . 14
    - 2.1.3.7 pyblockseis.wavelet.WaveletCollection . . . 15
  - 2.1.4 pyblockseis.threshold . . . 16
    - 2.1.4.1 pyblockseis.threshold.SNR_detect . . . 16
    - 2.1.4.2 pyblockseis.threshold.ecdf . . . 17
    - 2.1.4.3 pyblockseis.threshold.noise_model . . . 17
    - 2.1.4.4 pyblockseis.threshold.noise_thresholding . . . 17
    - 2.1.4.5 pyblockseis.threshold.signal_thresholding . . . 18

3 Indices and tables **19**

Python Module Index **21**

Index **23**

pyBlockSeis is a Python tool for processing seismic data using the continuous wavelet transform (CWT).

CHAPTER

ONE

GETTING STARTED

1.1 Requirements

- ObsPy and its dependencies

1.2 Installation

1.3 Usage

- Refer to the notebook tutorials.

1.4 Acknowledgements

- The tool is adapted from the Matlab software Block Choice Seismic Analysis (BCseis, version 1.1) by Charles A. Langston and S. Mostafa Mousavi.
- Forward and inverse CWTs functions based on the Synchrosqueezing Toolbox by Eugene Brevdo and Gaurav Thakur.

1.5 License

CHAPTER

TWO

API REFERENCE

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

2.1 Core Modules

pyblockseis.block	Core module for pyBlockSeis
pyblockseis.waveforms	Routines for handling waveform dataå
pyblockseis.wavelet	Routines for handling the wavelet transform
pyblockseis.threshold	Non-linear thresholding operations using the continuous wavelet transform

2.1.1 pyblockseis.block

Core module for pyBlockSeis

Functions

read	Read waveform files and processing parameters into a pyBlockSeis Block object

2.1.1.1 pyblockseis.block.read

read(*params=None*, ***kwargs*)

Read waveform files and processing parameters into a pyBlockSeis Block object

The *read* function accepts waveform files in either ObsPy Stream or Trace objects, or something ObsPy can read. If a *Parameter* object is not provided, the default processing values will be used instead.

Parameters

- **params** (*Parameter*) – parameters required to run the denoiser.
- **event** (Event or Catalog, optional) – Object containing information that describes the seismic event.
- **data** (Stream, Trace, str, ...) – Waveform data in ObsPy trace or stream formats or something ObsPy can read.
- **asdf_file** (*str*) – HDF5 file name.

- **field** (*str*) – path of waveform files within the ASDF volume, default is "raw_observed".

Examples

(1) Reading from a SAC file

```
>>> import pyblockseis as bcs
>>> params = bcs.params((block_threshold=1.0, noise_threshold="hard")
>>> block = bcs.read(params=params, data="testdata/5014.YW.0.sp0011.DPZ")
>>> # Using default values
>>> block = bcs.read(data="testdata/5014.YW.0.sp0011.DPZ")
```

(2) Reading from a ASDF file

```
>>> import pyblockseis as bcs
>>> params = bcs.params((block_threshold=1.0, noise_threshold="hard")
>>> block = bcs.read(params=params, asdf_file="testdata/578449.h5")
```

Classes

Block	Root object for the time series and CWT
Parameter	A container of parameters required to run *Block*

2.1.1.2 pyblockseis.block.Block

class Block(*params*, *event*, *waveforms*)

Bases: object

Root object for the time series and CWT

Main class object that handles the processing of seismic data using the continuous wavelet transform. The tags attribute is a list of available processed waveforms and their wavelet transforms, depending on the choices set in *Parameter*. See all possible tags below.

Parameters

- **params** (*Parameter*) – CWT operations.
- **event** (Event) – seismic event information.
- **waveforms** (*Waveforms*) – seismic data.

Attributes

params [Parameter object] CWT operations.

event [ObsPy event] event information.

waveforms [Waveforms object] seismic waveforms.

wavelets [WaveletCollection object] wavelet transforms of the processed data.

noise_model_tag [str] wavelet transform of data used to estimate the noise model, depending on the processing choices this will be "input", "band_rejected" or None.

Available Tags

tag	parameters	description
"input"	none	input data
"band_rejected"	bandpass_blocking	applied a band rejection filter
"noise_removed"	noise_threshold	noise removed from data
"signal_removed"	signal_threshold	signal removed from data

Basic Usage

```
>>> import pyblockseis as bcs
>>> block = bcs.read(data="testdata/5014.YW.0.sp0011.DPZ")
>>> block.run()
```

Methods

get_station_list	Function to return a list of available stations
get_waveforms	Returns a *Waveforms* object
get_wavelets	Returns a *WaveletCollection* object
plot	Plot the time-frequency representation, or scalogram, of the current pyblockseis Block object
reconstruct	Reconstruct time series
run	Function to run the CWT-based thresholding operations
write	Write the processed data to file

pyblockseis.block.Block.get_station_list

Block.**get_station_list**()
Function to return a list of available stations

pyblockseis.block.Block.get_waveforms

Block.**get_waveforms**(*tag*)
Returns a *Waveforms* object

A function that returns the waveforms of a tagged dataset

Parameters **tag** (*str*) – processed data

pyblockseis.block.Block.get_wavelets

Block.**get_wavelets**(*tag*)
Returns a *WaveletCollection* object

A function that returns the wavelet transform of a tagged dataset

Parameters **tag** (*str*) – processed data

pyblockseis.block.Block.plot

Block.**plot**(*tag*, *network=None*, *station=None*, *location=None*, *channel=None*, *component=None*)
Plot the time-frequency representation, or scalogram, of the current pyblockseis Block object

Plot the scalograms for all traces in the object. Alternatively, specific traces can be selected that matches the given station criteria.

Parameters

- **tag** (*str*) – processed data to plot.
- **network** (*str*) – network code.
- **station** (*str*) – station code.
- **location** (*str*) – location code.
- **channel** (*str*) – channel code.
- **component** (*str*) – component code.

Example

```
>>> block.plot("noise_removed", network="BK", channel="HH*")
```

pyblockseis.block.Block.reconstruct

Block.**reconstruct**(*tag*)
Reconstruct time series

A function that performs the inverse continuous wavelet transform to reconstruct the time series from the wavelet coefficients.

Parameters **tag** – time series to reconstruct.

pyblockseis.block.Block.run

Block.**run**()
Function to run the CWT-based thresholding operations

Apply the nonlinear thresholding operations given the values in the `params` attribute and reconstruct the time series.

pyblockseis.block.Block.write

Block.**write**(*tag*, *output=None*, *filename=None*, *format='npz'*, *network=None*, *station=None*, *location=None*, *channel=None*, *component=None*)

Write the processed data to file

Function to save the waveforms or wavelet transforms to a single uncompressed NumPy .npz format. Additional formats for the waveforms are supported through ObsPy, such as binary SAC. See `obspy.core.stream.Stream.write` for the supported waveform data formats.

Parameters

- **tag** (*str*) – input or processed dataset to save.
- **output** (*str*) – type of output to save, `"waveforms"` for time series data, or `"cwt"` for the continuous wavelet transform.
- **filename** (*str*) – name of the file to write, optional.
- **format** (*str*) – output file format, default is `"npz"`. For waveform data see `write` for additional supported formats.
- **network** (*str*) – network code.
- **station** (*str*) – station code.
- **location** (*str*) – location code.
- **channel** (*str*) – channel code.
- **component** (*str*) – component code.

Attributes

event	Returns an ObsPy event object containing information that describes the seismic event
tags	Returns a list of available tags in the dataset after applying the wavelet thresholding operations

pyblockseis.block.Block.event

property Block.**event**

Returns an ObsPy event object containing information that describes the seismic event

pyblockseis.block.Block.tags

property Block.**tags**
Returns a list of available tags in the dataset after applying the wavelet thresholding operations

2.1.1.3 pyblockseis.block.Parameter

class Parameter(**args*, ***kwargs*)
Bases: object

A container of parameters required to run *Block*

The Parameter class determines the appropriate processes to be applied to the seismograms.

Parameters

- **wave_type** (*str*) – wavelet filter type, options are "morlet", "shannon", "mhat", "hhat". Default is "morlet".
- **nvoices** (*int*) – number of voices, or the sampling of CWT in scale. Higher number of voices give finer resolution. Default is 16.
- **bandpass_blocking** (*bool*) – Default value True will apply a band rejection filter where wavelet coefficients are modified over a scale bandpass.
- **scale_min** (*float*) – minimum time scale for bandpass blocking. Default is 1.
- **scale_max** (*float*) – maximum time scale for bandpass blocking. Default is 200.
- **block_threshhold** – percent amplitude adjustment to the wavelet coefficients within scale_min and scale_max. For example a threshold of 5% means the wavelet coffi-cients in the band will be multipled by 0.05. Default is 0.
- **estimate_noise** (*bool*) – flag to compute the noise model, default is True.
- **noise_starttime** (*float*) – noise start time, default is 0.
- **noise_endtime** (*float*) – noise end time, default is 60.
- **noise_threshold** (*str*) – type of noise thresholding to be applied, the options are "hard" for hard thresholding and "soft" for soft thresholding. Default is None.
- **signal_threshold** (*str*) – type of signal thresholding to be appied, the options are "hard" for hard thresholding, and "soft" for soft thresholding. Default is None.
- **nsigma_method** (*str, int, float*) – method to determine the number of standard deviations for block thresholding. "donoho" for Donoho's Threshold criterion and "ECDF" for empirical cumulative probability distribution method. You can also specify the number of standard deviations by entering a number. None ECDF method assumes Gaussian statistic. The default method "ECDF" is recommended.
- **snr_detection** (*bool*) – Flag to apply the SNR detection method, default is False. If True it will be applied before hard thresholding.
- **snr_lowerbound** (*float*) – Noise level percent lower bound. Default is 1.0.

Methods

keys	Returns a list of object attributes.
update	

pyblockseis.block.Parameter.keys

Parameter.**keys**()
Returns a list of object attributes.

pyblockseis.block.Parameter.update

Parameter.**update**(*adict={}*)

2.1.2 pyblockseis.waveforms

Routines for handling waveform dataå

Classes

Waveforms	Main class for waveform data

2.1.2.1 pyblockseis.waveforms.Waveforms

class Waveforms(*tag_options*)
Bases: object

Main class for waveform data

Parameters **tag_options** (*list*) – a list of available tags, this defines the list of keys for the data attribute.

Additional Attributes

station_name [list of str] stations names.

data [dict of ObsPy Stream objects] waveform data, refer to `obspy.core.stream.Stream.select` for details on how to query the traces.

Methods

add_waveform	Function to add seismic waveform data

pyblockseis.waveforms.Waveforms.add_waveform

Waveforms.**add_waveform**(*wfs*, *tag*)
Function to add seismic waveform data

Parameters

- **wfs** (Stream, Trace, str, ...) – waveform files.
- **tag** (*str*) – data type.

2.1.3 pyblockseis.wavelet

Routines for handling the wavelet transform

Functions

blockbandpass	Apply a band reject filter to modify the wavelet coefficients over a scale bandpass
cwt	Continuous wavelet transform using the wavelet function
inverse_cwt	Inverse continuous wavelet tranform
wfiltfn	Wavelet transform function of the wavelet filter in fourier domain
wfilth	Fast fourier transform of the wavelet function

2.1.3.1 pyblockseis.wavelet.blockbandpass

blockbandpass(*Wx*, *scales*, *scale_min*, *scale_max*, *block_threshold*)
Apply a band reject filter to modify the wavelet coefficients over a scale bandpass

Parameters

- **Wx** (numpy.ndarray) – wavelet transform of shape (len(scales), len(time_series))
- **scale_min** (*float*) – minimum time scale for bandpass blocking.
- **scale_max** (*float*) – maximum time scale for bandpass blocking.
- **block_threshhold** – percent amplitude adjustment to the wavelet coefficients within scale_min and scale_max.

Returns modified wavelet transform of shape (len(scales), len(time_series)).

Return type numpy.ndarray

2.1.3.2 pyblockseis.wavelet.cwt

cwt (*time_series*, *wave_type*, *nvoices*, *dt*)
Continuous wavelet transform using the wavelet function

Parameters

- **time_series** (list or numpy.ndarray) – input time series data.
- **wave_type** (*str*) – wavelet function.
- **nvoices** (*int*) – sampling of CWT in scale.
- **dt** (*float*) – sampling period.

Returns the wavelet transform of shape (scales, time_series), and the length vector containing the associated scales.

Return type (numpy.ndarray, numpy.ndarray)

2.1.3.3 pyblockseis.wavelet.inverse_cwt

inverse_cwt (*Wx*, *wave_type*, *nvoices*)
Inverse continuous wavelet tranform

Reconstructs the original signal from the wavelet transform.

Parameters

- **Wx** (numpy.ndarray) – wavelet transform of shape (len(scales), len(time_series))
- **wave_type** (*str*) – wavelet function.
- **nvoices** (*int*) – sampling of CWT in scale.

Returns time series data.

Return type numpy.ndarray

2.1.3.4 pyblockseis.wavelet.wfiltfn

wfiltfn (*xi*, *wave_type*)
Wavelet transform function of the wavelet filter in fourier domain

Parameters

- **xi** (numpy.ndarray) – sampled time series.
- **wave_type** (*str*) – wavelet function.

Returns mother wavelet function.

Return type numpy.ndarray

2.1.3.5 pyblockseis.wavelet.wfilth

wfilth(*wave_type*, *N*, *a*)
Fast fourier transform of the wavelet function

Outputs the FFT of a given `wave_type` of length N at scale a.

Parameters

- **wave_type** (*str*) – wavelet function.
- **N** (*int*) – number of samples to calculate.
- **a** (*float*) – wavelet scale.

Returns wavelet sampling in frequency domain

Return type `numpy.ndarray`

Classes

Wavelet	One dimensional continues wavelet transform of a time series
WaveletCollection	A collection of wavelet objects

2.1.3.6 pyblockseis.wavelet.Wavelet

class Wavelet(*coefs=None*, *scales=None*, *headers=None*, ***kwargs*)
Bases: `object`

One dimensional continues wavelet transform of a time series

Main class for a single wavelet transform including the station headers, wavelet coefficients and scales. Noise model calculated from the function *noise_model* is passed through additional `kwargs`.

Parameters

- **coefs** (`numpy.ndarray`) – Continuous wavelet transform of a time series, the first axis corresponds to the scales and the second axis corresponds to the length of the time series.
- **scales** (`numpy.ndarray`) – Wavelet scales
- **headers** (dict, `Stats`) – header information of the data.
- **M** (`numpy.ndarray`) – mean of noise model.
- **S** (`numpy.ndarray`) – standard deviation of noise model.
- **P** (`numpy.ndarray`) – threshold of the noise signal.

Attributes

stats [Stats] header of the wavelet transform, including station info.

scales [numpy.ndarray] wavelet scales.

coefs [numpy.ndarray] wavelet coefficients.

noise_model [AttribDict] noise model

Methods

get_id	Returns the station ID

pyblockseis.wavelet.Wavelet.get_id

Wavelet.**get_id**()
Returns the station ID

Returns station ID containing network, station, location and channel codes.

Return type str

2.1.3.7 pyblockseis.wavelet.WaveletCollection

class WaveletCollection(*wavelets=None*)
Bases: object

A collection of wavelet objects

Main class that contains wavelet transforms for multiple time series.

Parameters **wavelets** (*Wavelet*, list) – wavelet transform(s).

Methods

count	
select	Query wavelets

pyblockseis.wavelet.WaveletCollection.count

WaveletCollection.**count**()

pyblockseis.wavelet.WaveletCollection.select

WaveletCollection.**select**(*network=None*, *station=None*, *location=None*, *channel=None*, *component=None*)

Query wavelets

Select wavelet transforms that matches the given station criteria.

Parameters

- **tag** (*str*) – processed data to plot.
- **network** (*str*) – network code.
- **station** (*str*) – station code.
- **location** (*str*) – location code.
- **channel** (*str*) – channel code.
- **component** (*str*) – component code.

2.1.4 pyblockseis.threshold

Non-linear thresholding operations using the continuous wavelet transform

Functions

SNR_detect	Apply SNR detection to CWT
ecdf	Empirical cumulative probability distribution
noise_model	Calculates noise model and threshold function.
noise_thresholding	Apply hard/soft thresholding to the noise (removing noise).
signal_thresholding	Apply hard/soft thresholding to the signal (removing signal).

2.1.4.1 pyblockseis.threshold.SNR_detect

SNR_detect(*Wx*, *M*, *newbeg*, *newend*, *snr_lowerbound*)

Apply SNR detection to CWT

Parameters

- **Wx** (numpy.ndarray) – wavelet transform of shape (len(scales), len(time_series))
- **M** (numpy.ndarray) – mean of noise model.
- **snr_lowerbound** (*float*) – noise level lower bound in percent.

Returns the updated mean and standard deviation of the noise model.

Return type (numpy.ndarray, numpy.ndarray)

2.1.4.2 pyblockseis.threshold.ecdf

ecdf(*x*)
Empirical cumulative probability distribution

Function returns the empirical cumulative distribution function of the input array.

Parameters **x** (numpy.ndarray) – a sample

Returns a 2-D array where the first and second axes correspond to the empirical cumulative distribution function evaluated at the sorted sample points.

Return type numpy.ndarray

2.1.4.3 pyblockseis.threshold.noise_model

noise_model(*Wx*, *delta*, *noise_starttime*, *noise_endtime*, *nsigma_method*, *nlbound*, *detection=False*)
Calculates noise model and threshold function.

Parameters

- **Wx** (numpy.ndarray) – wavelet transform of shape (len(scales), len(time_series))
- **delta** (*float*) – sampling interval.
- **noise_starttime** (*float*) – noise start time.
- **noise_endtime** (*float*) – noise end time.
- **nsigma_method** (*str*) – block thresholding method.
- **nlbound** (*float*) – noise level lower bound in percent.
- **nsigma_gauss** (*float*) – umber of std for block threshold using Gaussian statistic.
- **detection** (*bool*) – If `True` it will be applied before hard thresholding. Default is `False`.

Returns the mean, standard deviation of the noise model, and threshold of the noise signal.

Return type (numpy.ndarray, numpy.ndarray, numpy.ndarray)

2.1.4.4 pyblockseis.threshold.noise_thresholding

noise_thresholding(*Wx*, *noise_threshold*, *P*)
Apply hard/soft thresholding to the noise (removing noise).

Parameters

- **Wx** (numpy.ndarray) – wavelet transform of shape (len(scales), len(time_series))
- **noise_threshold** (*str*) – “soft” or “hard” noise thresholding.
- **P** (numpy.ndarray) – threshold of the noise signal.

Return Wx_new the new wavelet transform.

Rtype Wx_new numpy.ndarray

2.1.4.5 pyblockseis.threshold.signal_thresholding

signal_thresholding(*Wx*, *signal_threshold*, *P*)

Apply hard/soft thresholding to the signal (removing signal).

Parameters

- **Wx** (numpy.ndarray) – wavelet transform of shape (len(scales), len(time_series))
- **signal_threshold** (*str*) – "soft" or "hard" noise thresholding.
- **P** (numpy.ndarray) – threshold of the noise signal.

Return Wx_new the new wavelet transform.

Rtype Wx_new numpy.ndarray

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

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

PYTHON MODULE INDEX

b

pyblockseis.block, 5

t

pyblockseis.threshold, 16

w

pyblockseis.waveforms, 11
pyblockseis.wavelet, 12

INDEX

A

add_waveform() (*Waveforms method*), 12

B

Block (*class in pyblockseis.block*), 6
blockbandpass() (*in module pyblockseis.wavelet*), 12

C

count() (*WaveletCollection method*), 15
cwt() (*in module pyblockseis.wavelet*), 13

E

ecdf() (*in module pyblockseis.threshold*), 17
event() (*Block property*), 9

G

get_id() (*Wavelet method*), 15
get_station_list() (*Block method*), 7
get_waveforms() (*Block method*), 7
get_wavelets() (*Block method*), 8

I

inverse_cwt() (*in module pyblockseis.wavelet*), 13

K

keys() (*Parameter method*), 11

M

module
    pyblockseis.block, 5
    pyblockseis.threshold, 16
    pyblockseis.waveforms, 11
    pyblockseis.wavelet, 12

N

noise_model() (*in module pyblockseis.threshold*), 17
noise_thresholding() (*in module pyblockseis.threshold*), 17

P

Parameter (*class in pyblockseis.block*), 10
plot() (*Block method*), 8
pyblockseis.block
    module, 5
pyblockseis.threshold
    module, 16
pyblockseis.waveforms
    module, 11
pyblockseis.wavelet
    module, 12

R

read() (*in module pyblockseis.block*), 5
reconstruct() (*Block method*), 8
run() (*Block method*), 8

S

select() (*WaveletCollection method*), 16
signal_thresholding() (*in module pyblockseis.threshold*), 18
SNR_detect() (*in module pyblockseis.threshold*), 16

T

tags() (*Block property*), 10

U

update() (*Parameter method*), 11

W

Waveforms (*class in pyblockseis.waveforms*), 11
Wavelet (*class in pyblockseis.wavelet*), 14
WaveletCollection (*class in pyblockseis.wavelet*), 15
wfiltfn() (*in module pyblockseis.wavelet*), 13
wfilth() (*in module pyblockseis.wavelet*), 14
write() (*Block method*), 9