## Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>pyrvt</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>1.1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>1.2</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Installation</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>Using pyRVT</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>Frequently Asked Questions</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>Developer Interface</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>5.1</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>5.2</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>5.3</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>5.4</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>History</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>6.1</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>6.2</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>6.3</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>6.4</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>6.5</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>6.6</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>6.7</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>6.8</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>6.9</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>6.10</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>6.11</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>6.12</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>6.13</td>
<td>22</td>
</tr>
<tr>
<td>7</td>
<td>Contributing</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>7.1</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>7.2</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>7.3</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>7.4</td>
<td>25</td>
</tr>
</tbody>
</table>
Contents:
A Python library and command-line application for using random vibration theory to transform between acceleration Fourier amplitude spectrum and acceleration response spectrum.

- Free software: MIT license

Information on the installation and usage can be found in the documentation.

### 1.1 Features

Peak factor implementations:

- Vanmarcke (1975)
- Davenport (1964)
- Der Kiureghian (1985)
- Toro & McGuire (1987)
- Cartwright & Longuet-Higgins (1956)
- Liu & Pezeshk (1999)
1.2 Citation

Please cite this software using the following DOI:
Prior to using pyrvt, Python and a number of packages need to be installed. In addition to Python, the following packages need to be installed:

Required:
- **numpy** – fast vector operations
- **numba** – compiled functions for increased speed
- **scipy** – indefinite integration
- **pyexcel** – reading and writing XLS and XLSX files
- **pytest** – required for the unit tests

Optional:
- **matplotlib** – used for plotting
- **xlrd/xlwt** – reading and writing xls files
- **openpyxl** – reading xlsx files

Install Python dependencies is best accomplished with a package manager. On Windows or OS-X, I recommend using Miniconda3. On Linux, the package manager is preferred.

Miniconda has installers for Windows 32-bit, Windows 64-bit, and OS-X. While pyrvt has support Python27, Python3 is preferred.

After the installer is finished, install the required dependencies by opening a terminal. On Windows, this is best accomplished with Windows Key + r, enter cmd. Next enter the following command:

```
conda install --yes setuptools numpy scipy matplotlib numba openpyxl xlrd xlwt pip
```

On Windows, the text can copied and pasted if “Quick Edit” mode is enabled. To enable this feature, right click on the icon in the upper left portion of the window, and select “Properties”, and then check the “Quick Edit Mode” check box within the “Edit Options” group. Copy the text, and then paste it by click the right mouse button.

Now that the dependencies have been installed, install or upgrade pyrvt using pip:
You should now have `pyrvt` completely installed. Next, read about *using pyrvt*. 
Using pyRVT

pyRVT is used by executing `rvt_operator` with a number of arguments. These arguments can be found by running `rvt_operator`, which will produce the following output:

C:\Users\arkottke\Documents\>pyrvt --help
usage: pyrvt [-h] -i INPUT [-o OUTPUT] [-d DAMPING] [-f] [-m METHOD]
   {psa2fa,fa2psa}

Compute response or Fourier amplitude spectra using RVT.

positional arguments:
   {psa2fa,fa2psa} Operation to be performed. [psa2fa] converts from
   pseudo-spectral acceleration to Fourier amplitude. [fa2psa] converts from Fourier amplitude to pseudo-
   spectral acceleration.

optional arguments:
   -h, --help show this help message and exit
   -i INPUT, --input INPUT
   Path containing the input file(s). Supported file
   types are csv, xls, and xlsx -- provided the required
   packages have been installed. A single file or glob
   can be specified. An example of a glob would be
   "input/*/sa.xls" for all files within directory
   "input" ending in "_sa.xls".
   -o OUTPUT, --output OUTPUT
   Path where the output files should be created. If this
directory does not exist it will be created. Default: ./output
   -d DAMPING, --damping DAMPING
   Oscillator damping in decimal. Default: 0.05.
   -f, --fixed-spacing Fixed spacing of the oscillator period of 0.01 to 10
   sec log-spaced with 100 points. Target SA values will
   be interpolated if needed
   -m METHOD, --method METHOD

(continues on next page)
Specify the peak factor calculation method. Possible options are: [BJ84] Boore and Joyner (1984), [BT12] Boore and Thompson (2012), [DK85] Der Kiureghian (1985), [LP99] Liu and Pezeshk (1999), [TM87] Toro and McGuire (1987), and [V75] Vanmarcke (1975). If the BT12 method is used, then the magnitude, distance and region must be provided by the input files. If no value is provided, then 'V75' is used as the default.

For example, to compute the Fourier amplitude spectra that were compatible with target response spectrum the following command could be used:

```bash
$> rvt_operator psa2fa -i examples\example_targetSa.csv
```

The required format for the events is best understood by looking at one of the example event files. The name of the input file should include an underscore (i.e., ‘_’) during the creation of the output everything to the left of the last underscore is repeated.
1. How long does this take?

The depends on the frequency spacing and the number of events that are being considered. If a large number of periods and/or events are specified, it might take several minutes or longer.
This part of the documentation covers all the interfaces of pyRVT.

## 5.1 Motions

Random vibration theory (RVT) based motions.

**class** `pyrvt.motions.CompatibleRvtMotion` *(osc_freqs, osc_accels_target, duration=None, osc_damping=0.05, event_kwds=None, window_len=None, peak_calculator=None, calc_kwds=None)*

Response spectrum compatible RVT motion.

A `CompatibleRvtMotion` object is used to compute a Fourier amplitude spectrum that is compatible with a target response spectrum.

**class** `pyrvt.motions.RvtMotion` *(freqs=None, fourier_amps=None, duration=None, peak_calculator=None, calc_kwds=None)*

Random vibration theory motion.

**Parameters**

- `freqs` *(array_like, optional)* – Frequency array (Hz).
- `fourier_amps` *(array_like, optional)* – Absolute value of acceleration Fourier amplitudes.
- `duration` *(float, optional)* – Ground motion duration (sec).
- `peak_calculator` *(Calculator, optional)* – Peak calculator to use. If `None`, then the default peak calculator is used. The peak calculator may either be specified by a `Calculator` object, or by the initials of the calculator using `peak_calculator()`.
- `calc_kwds` *(dict, optional)* – Keywords to be passed during the creation the peak calculator. These keywords are only required for some peak calculators.
calc_attenuation(min_freq, max_freq=None)
Compute the site attenuation ($\kappa$) based on a log-linear fit.

Parameters

- min_freq (float) – minimum frequency of the fit (Hz).
- max_freq (float, optional) – maximum frequency of the fit. If None, then the maximum frequency range is used.

Returns

- atten (float) – attenuation parameter.
- r_sqr (float) – squared correlation coefficient of the fit ($R^2$). See scipy.stats.linregress().
- freqs (numpy.ndarray) – selected frequencies
- fitted (numpy.ndarray) – fitted values

Notes

This function computes the site attenuation defined by Anderson & Hough (1984, [AH84]) as:

$$a(f) = A_0 \exp(-\pi \kappa f)(f > f_E)$$

for a single Fourier amplitude spectrum

calc_osc_accels(osc_freqs, osc_damping=0.05, trans_func=[]) Pseudo-acceleration spectral response of an oscillator.

Parameters

- osc_freq (float) – Frequency of the oscillator (Hz).
- osc_damping (float) – Fractional damping of the oscillator (dec). For example, 0.05 for a damping ratio of 5%.
- trans_func (array_like, optional) – Transfer function to be applied to motion prior calculation of the oscillator response.

Returns

spec_accels – Peak pseudo-spectral acceleration of the oscillator

Return type
numpy.ndarray

calc_peak(transfer_func=None, **kwds) Compute the peak response.

Parameters

- transfer_func (array_like, optional) – Transfer function to apply to the motion. If None, then no transfer function is applied.

Returns

peak – Calculated peak

Return type
float

duration
Duration of the ground motion for RVT analysis.

fourier_amps Acceleration Fourier amplitude values (g·sec).

frefs Frequency values (Hz).
class `pyrvt.motions.SourceTheoryMotion`(*magnitude*, *distance*, *region*, *stress_drop=None*,
*depth=8*, *peak_calculator=None*, *calc_kwds=None*)

Single-corner source theory model.

The single-corner source theory model uses default parameters from Campbell (2003, [Cam03]).

calc_duration()
Compute the duration by combination of source and path.

*Returns* duration – Computed duration

*Return type* float

calc_fourier_amps(*freqs=None*)
Compute the acceleration Fourier amplitudes for a frequency range.

*Parameters*

• *freqs* *(array_like, optional)* – Frequency range. If no frequency range
  is specified then `log_spaced_values(0.05, 200.)()` is used.

*Returns* fourier_amps – acceleration Fourier amplitudes

*Return type* np.ndarray

`pyrvt.motions.calc_geometric_spreading(dist, params)`
Geometric spreading defined by piece-wise linear model.

*Parameters*

• *dist* *(float)* – Closest distance to the rupture surface (km).

• *params* *(List[(float, Optional[float])])* – List of (slope, limit) tuples that
define the attenuation. For an infinite distance use None. For example, [(1, None)] would
provide for 1/R geometric spreading to an infinite distance.

*Returns* coeff – Geometric spreading coefficient.

*Return type* float

`pyrvt.motions.calc_sdof_tf(freqs, osc_freq, osc_damping)`
Single-degree-of-freedom transfer function.

When applied on the acceleration Fourier amplitude spectrum, it provides the pseudo-spectral acceleration.

*Parameters*

• *freqs* *(array_like)* – Frequencies at which the transfer function should be calculated
  (Hz).

• *osc_freq* *(float)* – Frequency of the oscillator (Hz).

• *osc_damping* *(float)* – Fractional damping of the oscillator (decimal).

*Returns* Complex valued transfer function.

*Return type* numpy.ndarray

`pyrvt.motions.calc_stress_drop(magnitude)`
Stress drop using Atkinson & Boore (2011, [AB11]) model.

*Parameters* magnitude *(float)* – Moment magnitude of the stress drop.

*Returns* stress_drop – Stress drop (bars).

*Return type* float

`pyrvt.motions.log_spaced_values(lower, upper, per_decade=512)`
Generate values with constant log-spacing.
Parameters

- **lower** (*float*) – lower end of the range.
- **upper** (*float*) – upper end of the range.
- **per_decade** (*int, optional*) – number of points per decade. Default is 512 points per decade.

Returns values – Log-spaced values.

Return type **numpy.ndarray**

```python
pyrvt.motions.sort_increasing(*args)
```

Sort arrays such that they are increasing.

Parameters **args** (*array_like*) – arrays to be re-ordered.

Returns tuple containing sorted **numpy.ndarray**

Return type tuple

Raises **NotImplementedError** – If first array is not monotonic.

### 5.2 Peak Calculators

Peak factor models.

Published peak factor models, which compute the expected peak ground motion. A specific model may include oscillator duration correction.

```python
class pyrvt.peak_calculators.BooreJoyner1984(**kwargs)
```


RVT calculation based on the peak factor definition by Cartwright & Longuet-Higgins (1956, [CLH56]) and along with the root-mean-squared duration correction proposed by Boore & Joyner (1984, [BJ84]).

This RVT calculation is used by SMSIM and is described in Boore (2003, [Boo03]).

```python
__init__(**kwargs)
```

Initialize the class.

```python
class pyrvt.peak_calculators.BooreThompson(region, mag, dist, ref, **kwargs)
```

Abstract class for the Boore & Thompson duration correction.

The duration ratio is defined by Equation (10) in [BT12]. Magnitude and distance is interpolated using **scipy.interpolate.LinearNDInterpolator** on the natural log of the distance.

Parameters

- **region** (*str*) – Region for which the parameters were developed. Valid options are: ‘wna’ for Western North America (active tectonic), and ‘cena’ for Central and Eastern North America (stable tectonic).
- **mag** (*float*) – Magnitude of the event.
- **dist** (*float*) – Distance of the event in (km).
- **ref** (*str*) – Reference for coefficients, either: ‘bt12’ or ‘bt15’

```python
__init__(region, mag, dist, ref, **kwargs)
```

Initialize the class.
class pyrvt.peak_calculators.BooreThompson2012(region, mag, dist, **kwargs)

Peak calculation based on the peak factor definition by Cartwright & Longuet-Higgins (1956, [CLH56] along with the root-mean-squared duration correction proposed by Boore & Thompson (2012, [BT12]).

Parameters

• region (str) – Region for which the parameters were developed. Valid options are: ‘wna’ for Western North America (active tectonic), and ‘cena’ for Central and Eastern North America (stable tectonic).
• mag (float) – Magnitude of the event.
• dist (float) – Distance of the event in (km).

__init__(region, mag, dist, **kwargs)
Initialize the class.

class pyrvt.peak_calculators.BooreThompson2015(region, mag, dist, **kwargs)

Peak calculation based on the peak factor definition by Vanmarcke (1975, [Van75]) along with the root-mean-squared duration correction proposed by Boore & Thompson (2015, [BT15]).

Parameters

• region (str) – Region for which the parameters were developed. Valid options are: ‘wna’ for Western North America (active tectonic), and ‘cena’ for Central and Eastern North America (stable tectonic).
• mag (float) – Magnitude of the event.
• dist (float) – Distance of the event in (km).

__init__(region, mag, dist, **kwargs)
Initialize the class.

class pyrvt.peak_calculators.Calculator(**kwds)
Base class used for all peak calculator classes.

__call__(duration, freqs, fourier_amps, **kwargs)
Compute the peak response.

Parameters

• duration (float) – Duration of the stationary portion of the ground motion. Typically defined as the duration between the 5% and 75% normalized Arias intensity (sec).
• freqs (array_like) – Frequency of the Fourier amplitude spectrum (Hz).
• fourier_amps (array_like) – Amplitude of the Fourier amplitude spectrum with a single degree of freedom oscillator already applied if being used. Units are not important.

Returns

• max_resp (float) – expected maximum response.
• peak_factor (float) – associated peak factor.

__init__(**kwds)
Initialize the object.

5.2. Peak Calculators
__weakref__
    list of weak references to the object (if defined)

abbrev
    Abbreviated name of the calculator.

classmethod limited_num_zero_crossings(num_zero_crossings)
    Limit the number of zero crossing to a static limit.

min_zero_crossings
    Minimum number of zero crossings.

name
    Name of the calculator.

class pyrvt.peak_calculators.CartwrightLonguetHiggins1956(**kwargs)
    Cartwright and Longuet-Higgins (1956) peak factor.
    
    RVT calculation based on the peak factor definition by Cartwright and Longuet-Higgins (1956, [CLH56]) using
    the integral provided by Boore (2003, [Boo03]).

__init__(**kwargs)
    Initialize the class.

class pyrvt.peak_calculators.Davenport1964(**kwargs)
    Davenport (1964) peak factor.
    
    RVT calculation using the asymptotic solution proposed by Davenport (1964, [Dav64]).

__init__(**kwargs)
    Initialize the class.

classmethod asymtotic_approx(zero_crossings)
    Compute the peak factor from the asymptotic approximation.

    Parameters
    zero_crossings (float) – Number of zero crossing.

    Returns approx_peak_factor – Calculated peak factor.

    Return type float

class pyrvt.peak_calculators.DerKiureghian1985(**kwargs)
    Der Kiureghian (1985) peak factor.
    
    RVT calculation using peak factor derived by Davenport (1964, [Dav64]) with limits suggested by Igusa & Der
    Kiureghian (1985, [IDK85]).

__init__(**kwargs)
    Initialize the class.

class pyrvt.peak_calculators.LiuPezeshk1999(**kwargs)
    
    RVT calculation based on the peak factor definition by Cartwright & Longuet-Higgins (1956, [CLH56]) along
    with the root-mean-squared duration correction proposed by Liu & Pezeshk (1999, [LP99]).

__init__(**kwargs)
    Initialize the class.

class pyrvt.peak_calculators.SquaredSpectrum(freqs, fourier_amps)
    Squared Fourier amplitude spectrum.
    
    Used to store calculated spectral moments during calculations.

    Parameters
**freqs** (*array_like*) – Frequency of the Fourier amplitude spectrum (Hz)

**fourier_amps** (*array_like*) – Amplitude of the Fourier amplitude spectrum.

__init__(freqs, fourier_amps)
Initialize self. See help(type(self)) for accurate signature.

__weakref__
list of weak references to the object (if defined)

moment(*num*)
Compute the spectral moments.

The spectral moment is computed using the squared Fourier amplitude spectrum.

**Returns**
moment – Computed spectral moments.

**Return type**
float

class pyrvt.peak_calculators.ToroMcGuire1987(**kwargs)

Peak factor equation using asymptotic solution proposed by Davenport (1964, [Dav64]) with modifications proposed by Toro & McGuire (1987, [TM87]).

__init__(**kwargs)
Initialize the class.

class pyrvt.peak_calculators.Vanmarcke1975(use_nonstationarity_factor=True, **kwargs)
Vanmarcke (1975) peak factor.

The Vanmarcke (1975, [Van75]) peak factor, which includes the effects of clumping. The peak factor equation is from Equation (2) in Der Kiureghian (1980, [Kiu80]), which is based on Equation (29) in [Van75].

The cumulative density function (CDF) of the peak is defined as:

\[
F_x(x) = \left[1 - \exp\left(-x^2/2\right)\right]\exp\left[-N_z \frac{1 - \exp\left(-\sqrt{\pi/2}\delta_c x\right)}{\exp(x^2/2) - 1}\right]
\]

where \(N_z\) is the number of zero crossings, \(\delta_c\) is the effective bandwidth (\(\delta^{1.2}\)).

Typically, the expected value of the peak factor is calculated by integrating over the probability density function (i.e., \(f_x(x) = \frac{d}{dx} F_x(x)\)):

\[
E[x] = \int_0^\infty x f_x(x) dx
\]

However, because of the properties of \(F_x(x)\), specifically that it has non-zero probabilities for only positive values, \(E[x]\) can be computed directly from \(F_x(x)\).

\[
E[x] = \int_0^\infty 1 - F_x(x) dx.
\]

This is based on the following sources\(^1\) and\(^2\).

Parameters

use_nonstationarity_factor (bool) – If the non-stationarity factor should be applied.

__init__(use_nonstationarity_factor=True, **kwargs)
Initialize the class.

---

\(^1\) [http://en.wikipedia.org/wiki/Expected_value#Formulas_for_special_cases](http://en.wikipedia.org/wiki/Expected_value#Formulas_for_special_cases)

\(^2\) [http://stats.stackexchange.com/a/13377/48461](http://stats.stackexchange.com/a/13377/48461)
classmethod nonstationarity_factor(\(osc\_damping, osc\_freq, duration\))

Compute nonstationarity factor to modify duration.

Parameters

\- osc\_damping(float) – Oscillator damping (decimal).
\- osc\_freq(float) – Oscillator frequency (Hz).
\- duration(float) – Duration of the stationary portion of the ground motion

Returns Nonstationarity factor.

Return type float

class pyrvt.peak_calculators.WangRathje2018(\(region, mag, dist, **kwargs\))


Peak calculation based on the peak factor definition by Vanmarcke (1975, [Van75]) along with correction for oscillator duration and site amplification as described in Wang & Rathje (2018, [rathje18]).

\_\_init\_\_(\(region, mag, dist, **kwargs\))

Initialize the class.

pyrvt.peak_calculators.get_peak_calculator(\(method, calc\_kwds\))

Select a peak calculator based on a string.

Parameters

\- method(str) – Name of the peak calculation method
\- calc\_kwds(dict) – Keywords passed to the calculator

Returns calc – Calculator

Return type calculator

pyrvt.peak_calculators.get_region(\(region\))

Return the region naming used in this package.

Parameters region(str) – Regional synonym.

Returns region – Region either ‘cena’ or ‘wna’.

Return type str

5.3 Tools

Tools for reading/writing of files and performing operations.

pyrvt.tools.calc_compatible_spectra(\(method, periods, events, damping=0.05\))

Compute the response spectrum compatible motions.

Parameters

\- method(str) – RVT peak factor method, see get_peak_calculator().
\- periods(array_like) – Periods of the oscillator response shared across all events.
\- events(List[dict]) – All events to consider. See Note.
\- damping(float, optional) – Fractional damping of the oscillator (decimal). Default value of 0.05 for a damping ratio of 5%.

Returns Frequency of the computed Fourier amplitude spectra.
Return type: `numpy.ndarray`

Note: Each event dictionary should have the following keys:

- **psa**: `numpy.ndarray` – pseudo-spectral accelerations. This is the target for the `CompatibleRvtMotion`
- **duration**: float, optional – duration of the ground motion
- **magnitude**: float, optional – earthquake magnitude
- **distance**: float, optional – earthquake distance (km)
- **region**: str, optional – earthquake source region, see `get_region()` If no duration is provided one is estimated from the magnitude, distance, and region.

The `events` dictionary is modified by this function and adds the following keys:

- **duration**: float – duration of the ground motion if one was not specified
- **fa**: `numpy.ndarray` – Fourier amplitude spectra in units of g*sec
- **psa_calc**: `numpy.ndarray` – Pseudo-spectral acceleration calculated from `fa`. This will differ slightly from `psa_target`.

---

`pyrvt.tools.operation_fa2psa(src, dst, damping, method='LP99', fixed_spacing=True, verbose=True)`

Compute the Fourier amplitude spectrum from a accel. response spectrum.

**Parameters**

- **src**(str) – Source for the Fourier amplitudes. This can be a filename or pattern used in `glob.glob()`.
- **dst**(str) – Destination directory for the output PSA. The directory is created if it does not exist.
- **damping**(float) – Fractional damping of the oscillator (decimal).
- **method**(str) – RVT peak factor method, see `get_peak_calculator()`.
- **fixed_spacing**(bool, optional) – If True, then the periods are interpolated to 301 points equally space in log-space from 0.01 to 10.

`pyrvt.tools.operation_psa2fa(src, dst, damping, method='LP99', fixed_spacing=True, verbose=True)`

Compute the accel. response spectrum from a Fourier amplitude spectrum.

**Parameters**

- **src**(str) – Source for the pseudo-spectral accelerations (PSA). This can be a filename or pattern used in `glob.glob()`.
- **dst**(str) – Destination directory for the output PSA. The directory is created if it does not exist.
- **damping**(float) – Fractional damping of the oscillator (decimal).
- **method**(str) – RVT peak factor method, see `get_peak_calculator()`.
- **fixed_spacing**(bool, optional) – If True, then the periods are interpolated to 301 points equally space in log-space from 0.01 to 10.
- **verbose**(bool, optional) – Print status of calculation.
pyrvt.tools.read_events(fpath, response_type)
Read data from the file an Excel work book.

Parameters

- **fpath** (str or pathlib.Path) – Filename of the input file.
- **response_type** (str) – Type of response. Valid options are: ‘psa’ for pseudo-spectral acceleration, or ‘fa’ for Fourier amplitude.

Returns

- **ext** (str) – Extension of input file
- **reference** (numpy.ndarray) – Reference of the response. This is either period (sec) for response_type ‘psa’ or frequency (Hz) for response_type ‘fa’
- **events** (List[dict]) – List of events read from the file. See Note in calc_compatible_spectra() for more information on structure of the dictionaries.

pyrvt.tools.write_events(fname, reference, reference_label, response_type, response_label, events)
Write the events to a file.

Parameters

- **fname** (str) – Save the events to this file. The directory is created if needed.
- **reference** (array_like) – Periods of the oscillator response shared across all events.
- **reference_label** (str) – Label of the reference (e.g., ‘Frequency (Hz)’).
- **response_type** (str) – Type of response. Valid options: *psa* for pseudo-spectral acceleration, or *fa* for Fourier amplitude.
- **response_label** (str) – Label of the response type (e.g., ‘Fourier Ampl. (g/sec)’)
- **events** (List[dict]) – Events to write to file. See Note in compute_compatible_spectra() for more information.

Raises

NotImplementedError: – If extension is not supported

## 5.4 Runner

The command line interface for calling pyRVT.

See *Using pyRVT* for more details.

pyrvt.runner.main()
Perform the command line operations.
CHAPTER 6

History

6.1 0.7.0 (2018-10-20)
• Added Boore & Thompson (2015)
• Added Wang & Rathje (2018)
• Re-factored peak factor calculators

6.2 0.6.3 (2017-10-10)
• Add scripts for profiling
• Add numba to the requirements

6.3 0.6.2 (2017-05-07)
• Fixed PEP8 issues in docstrings.

6.4 0.6.0 (2017-05-05)
• Removed pyprind require and progress bars.

6.5 0.6.0 (2017-05-05)
• Improved performance with multiprocessing and numba.
• Changed CLI name from rvt_operator to pyrvt
6.6 0.5.8 (2016-11-16)

- Fixed: osc_freqs typo in tools.py
- Added: test cases for tools.py

6.7 0.5.7 (2016-07-14)

- Fixed version numbering.

6.8 0.5.6 (2016-07-14)

- Fixed manifest to include data directory.

6.9 0.5.5 (2016-07-11)

- Updated Travis build scripts.

6.10 0.5.4 (2016-07-09)

- Fixed building the documentation on RTFD

6.11 0.5.1 (2016-07-06)

- Fixed Travis deployment

6.12 0.5.0 (2016-07-06)

- Added ability to pass transfer function to PSA calculation.
- Removed extraneous files.
- Cleaned up documentation to use NumPy style.
- Fixed PEP8 issues.

6.13 0.1.0 (2016-03-04)

- First release on PyPI.
Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given. You can contribute in many ways:

### 7.1 Types of Contributions

#### 7.1.1 Report Bugs

Report bugs at [https://github.com/arkottke/pyrvt/issues](https://github.com/arkottke/pyrvt/issues).

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

#### 7.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” is open to whoever wants to implement it.

#### 7.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “feature” is open to whoever wants to implement it.
7.1.4 Write Documentation

pyRVT could always use more documentation, whether as part of the official pyRVT docs, in docstrings, or even on the web in blog posts, articles, and such. Docstrings should be formatted using the NumPy conventions.

7.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/arkottke/pyrvt/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

7.2 Get Started!

Ready to contribute? Here’s how to set up pyRVT for local development.

1. Fork the pyRVT repo on GitHub.
2. Clone your fork locally:
   
   ```
   $ git clone git@github.com:your_name_here/pyrvt.git
   ```
3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

   ```
   $ mkvirtualenv pyrvt
   $ cd pyrvt/
   $ pip install -r requirements_dev.txt
   $ python setup.py develop
   ```

   Note that it is easier to install matplotlib, numpy and scipy dependencies via your package manager.
4. Create a branch for local development:

   ```
   $ git checkout -b name-of-your-bugfix-or-feature
   ```

   Now you can make your changes locally.
5. When you’re done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

   ```
   $ flake8 pyrvt tests
   $ python setup.py test
   $ tox
   ```

   To get flake8 and tox, just pip install them into your virtualenv. If you have make installed, you can check the format and tests with:

   ```
   $ make lint
   $ make test
   $ make test-all
   ```
The documentation can be built with:

```
$ make docs
```

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

### 7.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.

2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.

3. The pull request should work for Python 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/arkottke/pyrvt/pull_requests and make sure that the tests pass for all supported Python versions.

### 7.4 Tips

To run a subset of tests:

```
$ py.test tests
```

Or if make is installed::

```
$ make tests
```
CHAPTER 8

Credits

8.1 Development Lead

- Albert Kottke <albert.kottke@gmail.com>

8.2 Contributors

None yet. Why not be the first?
References
CHAPTER 10

Indices and tables

- genindex
- modindex
- search
Bibliography


Python Module Index

p
  pyrvt, 11
  pyrvt.motions, 11
  pyrvt.peak_calculators, 14
  pyrvt.runner, 20
  pyrvt.tools, 18
Symbols

__call__() (pyrvt.peak_calculators.Calculator method), 15
__init__() (pyrvt.peak_calculators.BooreJoyner1984 method), 14
__init__() (pyrvt.peak_calculators.BooreThompson method), 14
__init__() (pyrvt.peak_calculators.BooreThompson2012 method), 15
__init__() (pyrvt.peak_calculators.BooreThompson2015 method), 15
__init__() (pyrvt.peak_calculators.Calculator method), 15
__init__() (pyrvt.peak_calculators.CartwrightLonguetHiggins1956 method), 16
__init__() (pyrvt.peak_calculators.Davenport1964 method), 16
__init__() (pyrvt.peak_calculators.DerKiureghian1985 method), 16
__init__() (pyrvt.peak_calculators.LiuPezeshk1999 method), 16
__init__() (pyrvt.peak_calculators.SquaredSpectrum method), 17
__init__() (pyrvt.peak_calculators.ToroMcGuire1987 method), 17
__init__() (pyrvt.peak_calculators.Vanmarcke1975 method), 17
__init__() (pyrvt.peak_calculators.WangRathje2018 method), 18
__weakref__(pyrvt.peak_calculators.BooreThompson attribute), 14
__weakref__(pyrvt.peak_calculators.Calculator attribute), 15
__weakref__(pyrvt.peak_calculators.SquaredSpectrum attribute), 17

asym_totic_approx() (pyrvt.peak_calculators.Davenport1964 class method), 16

B
BooreJoyner1984 (class in pyrvt.peak_calculators), 14
BooreThompson (class in pyrvt.peak_calculators), 14
BooreThompson2012 (class in pyrvt.peak_calculators), 15
BooreThompson2015 (class in pyrvt.peak_calculators), 15

C
calc_attenuation() (pyrvt.motions.RvtMotion method), 11
calc_compatible_spectra() (in module pyrvt.tools), 18
calc_duration() (pyrvt.motions.SourceTheoryMotion method), 13
calc_fourier_amps() (pyrvt.motions.SourceTheoryMotion method), 13
calc_geometric_spreading() (in module pyrvt.motions), 13
calc_osc_accels() (pyrvt.motions.RvtMotion method), 12
calc_peak() (pyrvt.motions.RvtMotion method), 12
calc_sdo_f_tf() (in module pyrvt.motions), 13
calc_stress_drop() (in module pyrvt.motions), 13
Calculator (class in pyrvt.peak_calculators), 15
CartwrightLonguetHiggins1956 (class in pyrvt.peak_calculators), 16
CompatibleRvtMotion (class in pyrvt.motions), 11

D
Davenport1964 (class in pyrvt.peak_calculators), 16
DerKiureghian1985 (class in pyrvt.peak_calculators), 16

Abbrev (pyrvt.peak_calculators.Calculator attribute), 16
duration (pyrvt.motions.RvtMotion attribute), 12

F
fourier_amps (pyrvt.motions.RvtMotion attribute), 12
freqs (pyrvt.motions.RvtMotion attribute), 12

G
get_peak_calculator() (in module pyrvt.peak_calculators), 18
get_region() (in module pyrvt.peak_calculators), 18

L
limited_num_zero_crossings()
   (pyrvt.peak_calculators.Calculator class method), 16
LiuPezeshk1999 (class in pyrvt.peak_calculators), 16
log_spaced_values() (in module pyrvt.motions), 13

M
main() (in module pyrvt.runner), 20
min_zero_crossings
   (pyrvt.peak_calculators.Calculator attribute), 16
moment() (pyrvt.peak_calculators.SquaredSpectrum method), 17

N
name (pyrvt.peak_calculators.Calculator attribute), 16
nonstationarity_factor()
   (pyrvt.peak_calculators.Vanmarcke1975 class method), 17

O
operation_fa2psa() (in module pyrvt.tools), 19
operation_psa2fa() (in module pyrvt.tools), 19

P
pyrvt (module), 11
pyrvt.motions (module), 11
pyrvt.peak_calculators (module), 14
pyrvt.runner (module), 20
pyrvt.tools (module), 18

R
read_events() (in module pyrvt.tools), 19
RvtMotion (class in pyrvt.motions), 11

S
sort_increasing() (in module pyrvt.motions), 14
SourceTheoryMotion (class in pyrvt.motions), 12

SquaredSpectrum (class in pyrvt.peak_calculators), 16

T
ToroMcGuire1987 (class in pyrvt.peak_calculators), 17

V
Vanmarcke1975 (class in pyrvt.peak_calculators), 17
W
WangRathje2018 (class in pyrvt.peak_calculators), 18
write_events() (in module pyrvt.tools), 20