SourceSpec Documentation

Release 1.8+248.g4e9c69b

Claudio Satriano

Jun 24, 2026

START

Earthquake source parameters from P- or S-wave displacement spectra

Release
1.8+248.g4e9c69b

Date
Jun 24, 2026

Project Page
GitHub

SourceSpec is a collection of command line tools to compute earthquake source parameters (seismic moment, corner frequency, radiated energy, source size, static stress drop, apparent stress) from the inversion of P-wave and S-wave displacement spectra recorded at one or more seismic stations. SourceSpec also computes attenuation parameters (t-star, quality factor) and, as a bonus, local magnitude.

See Madariaga [2011] for a primer on earthquake source parameters and scaling laws.

Go to section *Theoretical Background* to get more information on how the code works.

SourceSpec is written in Python and requires a working Python environment to run (see *Installation*). However, since SourceSpec is based on command line, you don't have to know how to code in Python to use it.

The SourceSpec package is made of several command line tools:

- `source_spec`: Compute earthquake source parameters from the inversion of P- or S-wave spectra.
- `source_model`: Direct modelling of P- or S-wave spectra, based on user-defined earthquake source parameters.
- `source_residuals`: Compute station residuals from `source_spec` output.
- `clipping_detection`: Test the clipping detection algorithm.
- `plot_sourcepars`: 1D or 2D plot of source parameters from a sqlite parameter file.

CHAPTER

ONE

GETTING STARTED

This section should get you up and running with SourceSpec in a few minutes. If you need more details, please refer to *Theoretical Background*.

1.1 For the impatient

Note

Note that the default config parameters are suited for a M<5 earthquake recorded within ~100 km. Adjust `win_length`, `noise_pre_time`, the frequency bands (`bp_freqmin_*`, `bp_freqmax_*`, `freq1_*`, `freq2_*`) and the bounds on `fc` and `t_star`, according to your problem.

1.1.1 Use case: miniSEED + StationXML + QuakeML

If you have seismic recordings in miniSEED format (e.g., `traces.mseed`), metadata in StationXML format (e.g., `station.xml`) and event information in QuakeML format (e.g., `event.xml`), then:

1. Generate a config file via `source_spec -S`;
2. Edit the config file variable `station_metadata` to point to `station.xml` file;
3. Run `source_spec -t traces.mseed -q event.xml`.

1.1.2 Use case: SAC + PAZ + SourceSpec Event File

If you have seismic recordings in SAC format (e.g., in a directory named `sac_data`), metadata as SAC polezero (PAZ) (e.g., in a directory named `paz`) and event information in any format, then:

1. Generate a config file via `source_spec -S`;
2. Edit the config file variable `station_metadata` to point to the `paz` directory;
3. Generate a sample *SourceSpec Event File* using `source_spec -y`; this will create a file named `ssp_event.yaml`;
4. Edit the file `ssp_event.yaml` with your event information;
5. Run `source_spec -t sac_data -H ssp_event.yaml`.

1.2 Command line arguments

After successfully installed SourceSpec (see *Installation*), you can get help on the command line arguments used by each code by typing from your terminal:

```
source_spec -h
```

(or `source_model -h`, or `source_residuals -h`).

`source_spec` and `source_model` require you to provide the path to seismic traces via the `--trace_path` command line argument (see *Input File Formats*).

Information on the seismic event can be stored in the trace header (SAC format), or provided through a QuakeML file (`--qmlfile`) or, alternatively (`--hypocenter`), through a *SourceSpec Event File*, a HYPO71 file, or a HYPOINVERSE-2000 file. See *Input File Formats* for more information on the supported file formats.

1.3 Configuration file

`source_spec` and `source_model` require a configuration file. The default file name is `source_spec.conf`, other file names can be specified via the `--configfile` command line argument.

You can generate a sample configuration file through:

```
source_spec -S
```

Take your time to go through the generated configuration file (named `source_spec.conf`): the comments within the file will guide you on how to set up the different parameters.

More details are in section *Configuration File*.

CHAPTER

TWO

THEORETICAL BACKGROUND

2.1 Overview

`source_spec` inverts the P- or S-wave displacement amplitude spectra from station recordings of a single event.

For each station, the code computes P- or S-wave displacement amplitude spectra for each component x (e.g., Z, N, E), within a predefined time window.

$$S_x^{p|s}(f) = \left| \int_{t_0^{p|s}}^{t_1^{p|s}} s_x(t) e^{-i2\pi f t} dt \right|$$

where the exponent $p|s$ means that we are considering either P- or S-waves, $t_0^{p|s}$ and $t_1^{p|s}$ are the start and end times of the P- or S-wave time window, $s_x(t)$ is the displacement time series for component x, and f is the frequency.

Note that the Fourier amplitude spectrum of ground displacement has the dimensions of displacement ($s_x(t)$) multiplied by time (dt).

The same thing is done for a noise time window: noise spectrum is used to compute spectral signal-to-noise ratio (and possibly reject low SNR spectra) and, optionally, to weight the spectral inversion.

Fig. 1: Example three-component trace plot (in velocity), showing noise and S-wave windows.

The component spectra are combined through the root sum of squares (e.g., Z, N, E):

$$S^{p|s}(f) = \sqrt{\left(S_z^{p|s}(f)\right)^2 + \left(S_n^{p|s}(f)\right)^2 + \left(S_e^{p|s}(f)\right)^2}$$

(This is actually done later in the code, after converting the spectra to magnitude units, see below.)

Fig. 2: Example displacement spectrum for noise and S-wave, including inversion results.

2.2 Spectral model

The Fourier amplitude spectrum of the P- or S-wave displacement in far field can be modelled as the product of a source term [Brune, 1970] and a propagation term (geometric and anelastic attenuation of body waves):

$$S^{p|s}(f) = \frac{1}{\mathcal{G}(r)} \times \frac{FR_{\Theta\Phi}}{4\pi\rho_h^{1/2}\rho_r^{1/2}c_h^{5/2}c_r^{1/2}} \times M_O \times \frac{1}{1 + \left(\frac{f}{f_c^{p|s}}\right)^2} \times e^{-\pi f t^*}$$

where:

- $\mathcal{G}(r)$ is the geometrical spreading coefficient (see below) and r is the hypocentral distance;

- F is the free surface amplification factor (generally assumed to be 2);
- $R_{\Theta\Phi}$ is the radiation pattern coefficient for P- or S-waves (average or computed from focal mechanism, if available);
- ρ_h and ρ_r are the medium densities at the hypocenter and at the receiver, respectively;
- c_h and c_r are the P- or S-wave velocities at the hypocenter and at the receiver, respectively;
- M_O is the seismic moment;
- f is the frequency;
- $f_c^{p|s}$ is the corner frequency for P- or S-waves;
- t^* is an attenuation parameter which includes anelastic path attenuation (quality factor) and station-specific effects.

2.2.1 Geometrical spreading

The geometrical spreading coefficient $\mathcal{G}(r)$ can be defined, for local and regional distances, in one of the following ways (see the `geom_spred_model` option in *Configuration File*):

- $\mathcal{G}(r) = r^n$: n can be any positive number. $n = 1$ (default value) is the theoretical value for a body wave in a homogeneous full-space; $n = 0.5$ is the theoretical value for a surface wave in a homogeneous half-space.
- Following Boatwright *et al.* [2002] (eq. 8), to account for the mixture of body waves, Lg waves and surface waves at regional distances ($r < 200km$), a two-part geometrical spreading coefficient:
  - body wave spreading ($\mathcal{G}(r) = r$) for hypocentral distances below a cutoff distance r_0;
  - frequency dependent spreading for hypocentral distances above the cutoff distance r_0.

More precisely, the expression derived from Boatwright *et al.* [2002] is:

$$\mathcal{G}(r) = \begin{cases} r & r \leq r_0 \\ r_0 (r/r_0)^{\gamma(f)} & r > r_0 \end{cases}$$

with

$$\gamma(f) = \begin{cases} 0.5 & f \leq 0.20Hz \\ 0.5 + 2\log_{10}(5f) & 0.20 < f < 0.25Hz \\ 0.7 & f \geq 0.25Hz \end{cases}$$

Note that here we use the square root of eq. 8 in Boatwright *et al.* [2002], since we correct the spectral amplitude and not the energy.

For teleseismic distances (see the option `geom_spread_min_teleseismic_distance` in *Configuration File*), the geometrical spreading coefficient is defined as in Okal [1992] (eq. 4):

$$\mathcal{G}(\Delta) = \frac{a}{g(\Delta)}$$

where Δ is the great circle distance between the source and the receiver, a is the Earth radius and $g(\Delta)$ is defined as:

$$g(\Delta) = \left(\frac{\rho_h c_h}{\rho_r c_r} \frac{\sin i_h}{\sin \Delta} \frac{1}{\cos i_r} \left| \frac{di_h}{d\Delta} \right| \right)^{1/2}$$

where ρ_h and ρ_r are the medium densities at the hypocenter and at the receiver, respectively, c_h and c_r are the P- or S-wave velocities at the hypocenter and at the receiver, respectively, i_h and i_r are the takeoff angle (hypocenter) and the incidence angle (receiver), respectively, and $\frac{di_h}{d\Delta}$ is the variation of the takeoff angle within a ray tube of width Δ (see Okal [1992] for details).

2.3 Building spectra

In `source_spec`, the observed spectrum of component x (vertical or horizontal), $S_x^{p|s}(f)$ is converted into moment magnitude units M_w.

The first step is to multiply the spectrum for the geometrical spreading coefficient and convert it to seismic moment units:

$$M_x^{p|s}(f) \equiv \mathcal{G}(r) \times \frac{4\pi\rho_h^{1/2}\rho_r^{1/2}c_h^{5/2}c_r^{1/2}}{FR_{\Theta\Phi}} \times S_x^{p|s}(f)$$

Then the spectrum is converted in units of magnitude:

$$Y_x^{p|s}(f) \equiv \frac{2}{3} \times \left(\log_{10} M_x^{p|s}(f) - 9.1\right)$$

And the final data vector $Y^{p|s}(f)$ is obtained by combining the three components (e.g., Z, N, E) through the root sum of squares:

$$Y^{p|s}(f) = \sqrt{\left(Y_z^{p|s}(f)\right)^2 + \left(Y_n^{p|s}(f)\right)^2 + \left(Y_e^{p|s}(f)\right)^2}$$

The data vector is compared to the theoretical model $M_{theo}^{p|s}(f)$ which incorporates the Brune's spectrum of the seismic moment and the inelastic attenuation. This model is also converted in magnitude units:

$$Y^{p|s}(f) = \frac{2}{3}\left[\log_{10}\left(M_{theo}^{p|s}(f)\right) - 9.1\right] = \frac{2}{3}\left[\log_{10}\left(M_O \times \frac{1}{1+\left(\frac{f}{f_c^{p|s}}\right)^2} \times e^{-\pi f t^*}\right) - 9.1\right] =$$

$$= \frac{2}{3}\left(\log_{10} M_0 - 9.1\right) + \frac{2}{3}\left[\log_{10}\left(\frac{1}{1+\left(\frac{f}{f_c^{p|s}}\right)^2}\right) + \log_{10}\left(e^{-\pi f t^*}\right)\right]$$

Finally coming to the following model used for the inversion:

$$Y^{p|s}(f) = M_w + \frac{2}{3}\left[-\log_{10}\left(1+\left(\frac{f}{f_c^{p|s}}\right)^2\right) - \pi\, f t^* \log_{10} e\right]$$

where $M_w \equiv \frac{2}{3}(\log_{10} M_0 - 9.1)$.

2.4 Inverted parameters

The parameters determined from the spectral inversion are M_w, $f_c^{p|s}$ and t^*.

As an option, an attenuation model (fixed value or frequency-dependent quality factor) can be specified through the `Q_model` parameter in *Configuration File*. In this case, t^* is not inverted for, but computed from the attenuation model, as:

$$t^*(r, f) = \frac{tt^{p|s}(r)}{Q^{p|s}(f)}$$

where $tt^{p|s}(r)$ is the P- or S-wave travel time from source to station and $Q^{p|s}(f)$ is the (frequency-dependent) P- or S-wave quality factor.

The inversion is performed in moment magnitude M_w units (logarithmic amplitude). More details on the inversion method can be found in *Spectral Inversion, Quality, and Uncertainty*.

2.5 Other computed parameters

Starting from the inverted parameters M_0 (M_w), $f_c^{p|s}$, t^* and following the equations in Madariaga [2011] and Lancieri *et al.* [2012], other quantities are computed for each station:

- the static stress drop $\Delta\sigma$
- the source radius a
- the radiated energy E_r
- the apparent stress σ_a
- the quality factor Q_0 of P- or S-waves

As a bonus, local magnitude M_l can be computed as well.

Event summaries (mean, weighted mean, percentiles) are computed from single station estimates. For mean and weighted mean estimation, outliers are rejected based on the interquartile range rule.

2.5.1 Source radius

The source radius is computed, assuming a circular rupture model, from the corner frequency $f_c^{p|s}$ (Kaneko and Shearer [2014], equation 2):

$$a = k^{p|s} \frac{\beta_h}{f_c^{p|s}}$$

where β_h is the shear wave speed at the hypocenter (in m/s), $f_c^{p|s}$ is the corner frequency (in Hz) estimated from the spectral inversion of P or S waves and $k^{p|s}$ is a constant which depends on the source model.

Brune [1970] provides an expression for k^s in the case of a static circular crack (equation 31 in Madariaga [2011]):

$$k^s_{Brune} = 0.3724$$

Kaneko and Shearer [2014] compiled a table including their own values for k^p and k^s as well as values obtained from other authors. The values are given as a function of the rupture velocity V_r of a dynamic circular crack (or a static crack, when V_r is infinite):

Table 1: Table 1 of Kaneko and Shearer [2014]

V_r/β_h	$k^p_{K\&S}$	$k^s_{K\&S}$	k^p_{Mada}	k^s_{Mada}	k^s_{Brune}	$k^p_{S\&H}$	$k^s_{S\&H}$
Infinite					0.3724		
0.9	0.38	0.26	0.32	0.21		0.42	0.29
0.8	0.35	0.26				0.39	0.28
0.7	0.32	0.26				0.36	0.27
0.6	0.30	0.25				0.34	0.27
0.5	0.28	0.22				0.31	0.24

Where "K&S" stands for Kaneko and Shearer [2014], "Mada" for Madariaga [1976], and "S&H" for Sato and Hirasawa [1973].

2.5.2 Static stress drop

The static stress drop $\Delta\sigma$ is computed under the assumption of a circular rupture of radius a, as discussed in Madariaga [2011] (equation 27):

$$\Delta\sigma = \frac{7}{16} \frac{M_0}{a^3}$$

where M_0 is the seismic moment (in $N \cdot m$) and a is the source radius (in m).

2.5.3 Radiated energy

The computation of the radiated energy E_r starts with the integral of the squared velocity amplitude spectrum: $[\dot{S}^{p|s}(f)]^2 = [2\pi f S^{p|s}(f)]^2$.

Following Boatwright *et al.* [2002] (equation 1) and Lancieri *et al.* [2012] (equation 3), the P- or S-wave radiated energy is computed as:

$$\tilde{\tilde{E}}_r^{p|s} = 8\pi \mathcal{G}^2(r) C^2 \rho_r c_r \int_{f_{min}}^{f_{max}} e^{2\pi f t^*} [\dot{S}^{p|s}(f)]^2 df$$

where $\mathcal{G}^2(r)$ is the squared geometrical spreading coefficient (see above), C is a constant discussed below, ρ_r and c_r are, respectively, the density and P- or S-wave velocity at the receiver (their product is the seismic impedance), f_{min} and f_{max} are the minimum and maximum frequency used to compute the energy (see *Configuration File* for details on the `Er_freq_range` parameter), and the exponential term in the integrand is the squared correction for anelastic attenuation. The double tilde on top of $\tilde{\tilde{E}}_r^{p|s}$ means that the radiated energy needs to be further corrected for noise and finite bandwidth (see below).

The constant C is defined in Boatwright *et al.* [2002] (equation 2) as:

$$C = \frac{\langle R_{\Theta\Phi} \rangle}{R_{\Theta\Phi} F}$$

where $\langle R_{\Theta\Phi} \rangle$ is the root mean square P- or S-wave radiation pattern computed on the focal sphere, $R_{\Theta\Phi}$ is the radiation pattern coefficient for the given station, and F is the free surface amplification factor. If a focal mechanism is not available, then it is assumed $R_{\Theta\Phi} = \langle R_{\Theta\Phi} \rangle$ and, hence, $C = 1/F$. This assumption means that we rely on the averaging between measurements of radiated energy at different stations, instead of precise measurements at a single station.

Noise correction

To account for low frequency noise, below the corner frequency, under the hypothesis that energy is additive and that noise is stationary, we compute a noise-corrected energy as:

$$\tilde{E}_r^{p|s} = \tilde{\tilde{E}}_r^{p|s} - \tilde{\tilde{E}}_r^{noise}$$

where the first term is the radiated energy computed from the P- or S-wave spectrum and the second term is the radiated energy computed from the noise spectrum. If the above difference is negative, then the measure is rejected, since the noise is too large compared to the signal.

Finite bandwidth correction

Next step is to correct the radiated energy for the missing energy above f_{max}, not taken into account in the integral of the squared velocity amplitude spectrum (finite bandwidth correction). Following Lancieri *et al.* [2012] (equation 4), and Di Bona and Rovelli [1988], the noise-corrected radiated energy is divided by the theoretical ratio R between the estimated radiated energy and the true radiated energy, defined as:

$$R = \frac{2}{\pi} \left[\arctan(f_{max}/f_c^{p|s}) - \frac{f_{max}/f_c^{p|s}}{1 + (f_{max}/f_c^{p|s})^2} \right]$$

where f_{max} is the maximum frequency used to compute the energy integral and $f_c^{p|s}$ is the P- or S-wave corner frequency.

The values of R range between 0 (for $f_{max}/f_c^{p|s} \to 0$) and 1 (for $f_{max}/f_c^{p|s} \to \infty$).

The corrected radiated energy for P- or S-waves is then:

$$E_r^{p|s} = \frac{\tilde{E}_r^{p|s}}{R}$$

Energy partition

The final step is to account for the partition of energy between P and S waves. Following Boatwright and Choy [1986] (equations 8 and 15) the ratio between the radiated energy measured from S-waves and the radiated energy measured from P-waves is:

$$\frac{E_r^s}{E_r^p} = 15.6$$

The final estimate of the radiated energy is then:

$$E_r = (1 + 15.6)\, E_r^p$$

or

$$E_r = \left(1 + \frac{1}{15.6}\right) E_r^s$$

depending on whether the radiated energy is computed from P or S waves.

2.5.4 Apparent stress

The apparent stress σ_a is computed as (Madariaga [2011], eq. 18):

$$\sigma_a = \mu_h \frac{E_r}{M_0}$$

where μ_h is the shear modulus (or rigidity, in Pa) near the hypocenter, E_r is the radiated energy (in $N \cdot m$), and M_0 is the seismic moment (in $N \cdot m$).

The value of μ_h is computed from the shear wave velocity (β_h) and the density (ρ_h) at the hypocenter, using the following expression:

$$\mu_h = \rho_h \beta_h^2$$

2.5.5 Quality factor

The retrieved attenuation parameter t^* is converted to the P- or S-wave quality factor $Q_0^{p|s}$ using the following expression:

$$Q_0^{p|s} = \frac{tt^{p|s}(r)}{t^*}$$

where $tt^{p|s}(r)$ is the P- or S-wave travel time from source to station and r is the hypocentral distance.

> **Note**
>
> If a prior attenuation model is provided (see the `Q_model` parameter in *Configuration File*), then t^* is not inverted and $Q_0^{p|s}$ is not computed.

2.6 Station Residuals

Station-specific effects can be determined by running `source_spec` on several events and computing the average of station residuals between observed and inverted spectra. These averages are obtained through the command `source_residuals`; the resulting residuals file can be used for a second run of `source_spec` (see the `residuals_filepath` option in *Configuration File*).

CHAPTER

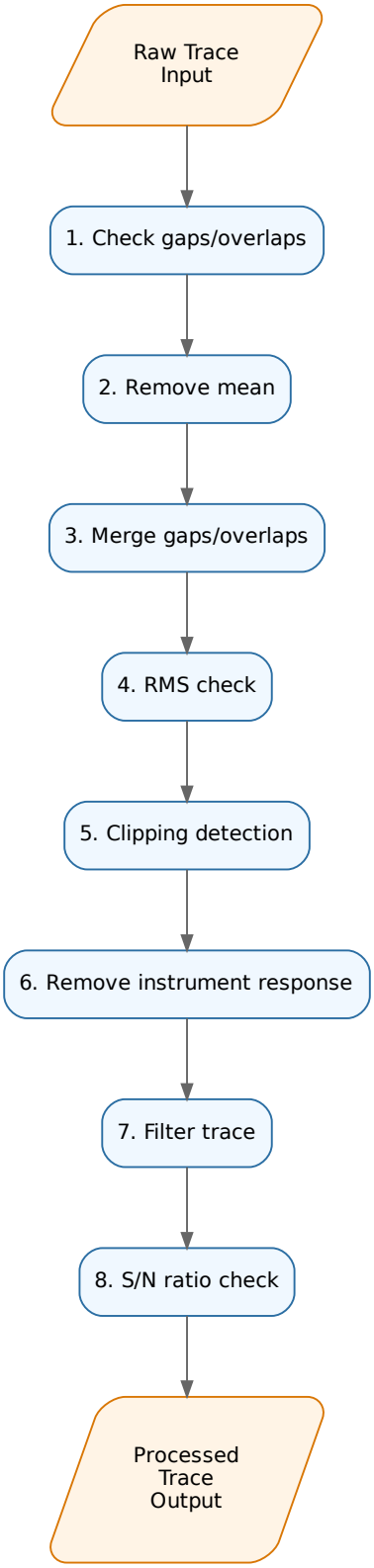
THREE

SIGNAL PROCESSING

This page summarizes the main signal and spectral processing steps used to construct amplitude spectra for inversion in SourceSpec.

The diagrams provide an overview of the processing pipelines. For more details, see the detailed steps below each diagram.

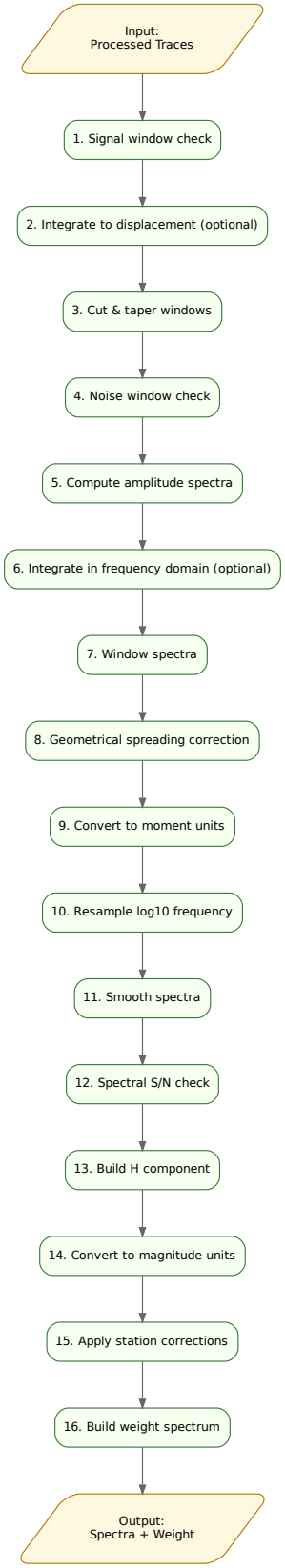
3.1 Trace Processing



1. Traces are checked for gaps and overlaps. Traces with cumulative gap or overlap duration larger than `gap_max` or `overlap_max`, respectively, are skipped.
2. The trace mean is removed (only if the configuration parameter `trace_units` is set to `auto`, i.e., if the trace has not been preprocessed outside SourceSpec).
3. Gaps and overlaps are merged.
4. Traces with RMS smaller than `rmsmin` are skipped.
5. Traces are optionally checked for clipping (see *Clipping Detection*).
6. Instrumental response is removed and trace transformed in its physical units (e.g., velocity, acceleration).
7. Trace is filtered.
8. Signal to noise ratio is measured as the ratio between signal RMS in the P- or S-window (depending on the `wave_type` parameter) and the RMS of the noise window. Traces with signal to noise ratio smaller than `sn_min` are skipped.

See the source code of *ssp_process_traces.process_traces()* for implementation details.

3.2 Spectral Processing



1. A check is made to see if the signal window contains enough data. Traces with no data or more than 25% of zero values are skipped. This is typically due to a wrong specification of signal window (e.g., wrong P or S arrivals).
2. Traces are integrated to displacement in time domain, if `time_domain_int` is `True`.
3. Signal and noise windows are cut and tapered.
4. The noise window is checked. If the ratio between noise RMS and signal RMS is smaller than 1e-6, the noise is considered not significant and the trace is skipped.
5. The amplitude spectra of the signal and noise windows are computed, using `numpy.fft.rfft()`. If `spectral_win_length` is not `None`, the signal is zero-padded to this length before computing the Fast Fourier Transform.
6. Spectra are integrated to displacement in frequency domain, if `time_domain_int` is `False`.
7. Amplitude spectra are windowed (see config parameters `freq1_broadb`, `freq2_broadb` and similar).
8. Geometrical spreading is corrected (see *Geometrical spreading*).
9. Amplitude spectra are converted to seismic moment units (see *Building spectra*).
10. Amplitude spectra are resampled in log_{10} frequency spacing.
11. The resampled spectra are smoothed. The smoothing window width is defined in frequency decades (config parameter `spectral_smooth_width_decades`)
12. The spectral signal to noise ratio is checked. Amplitude spectra with signal to noise ratio smaller than `spectral_sn_min` are skipped.
13. The "H" component is built based on one or more spectral components, depending on the `wave_type` and `ignore_vertical` config parameters:
    - if `wave_type` is `P` or `S`:
      - if `ignore_vertical` is `False`, the two horizontals and the vertical components are combined;
      - if `ignore_vertical` is `True`, only the two horizontals components are combined;
    - if `wave_type` is `SV`:
      - if `ignore_vertical` is `False`, the radial and the vertical component are combined;
      - if `ignore_vertical` is `True`, only the radial component is used;
    - if `wave_type` is `SH`:
      - only the transverse component is used;

    Spectra are combined through the root sum of squares (see *Overview*).
14. All the amplitude spectra are converted to moment magnitude units (see *Building spectra*).
15. Station corrections are applied, if requested (see *Station Residuals*).
16. The weight spectrum is built, depending on the config option `weighting`.

See the source code of *`ssp_build_spectra.build_spectra()`* for implementation details.

CHAPTER

FOUR

CLIPPING DETECTION

SourceSpec can optionally check for clipping in the traces. Two algorithms are available for this purpose:

1. *"Clipping Score" algorithm* (default): compute a trace clipping score based on the shape of the kernel density estimation.
2. *"Clipping Peaks" algorithm*: check if trace is clipped, based on the number of peaks in the kernel density estimation;

Both algorithms are based on the kernel density estimation of the trace amplitude values, using the `scipy.stats.gaussian_kde` class. See the documentation of this class for more details on the kernel density.

The clipping detection algorithm is selected through the `clipping_detection_algorithm` parameter in the *Configuration File*. The algorithm options are set via the configuration file.

Debug plots cand be activated by setting the `clipping_debug_plot` parameter in the *Configuration File* to `True`.

A command line script (`clipping_detection`) is available to test the clipping detection algorithm on a set of traces. Run:

```
$ clipping_detection --help
```

for details on the script usage.

4.1 "Clipping Score" algorithm

The "Clipping Score" algorithm computes a trace clipping score based on the shape of the kernel density estimation of the trace amplitude values.

The algorithm is based on the following steps:

1. The trace is detrended and demeaned. Optionally, the trace baseline can be removed (if the configuration parameter `remove_baseline` is set to `True`)
2. A kernel density estimation is computed on the trace amplitude values.
3. Two weighted kernel density functions are computed:
   - a full weighted kernel density, where the kernel density is weighted by the distance from the zero mean amplitude value, using a 8th order power function between 1 and 100.
   - a weighted kernel density without the central peak, where the kernel density is weighted by the distance from the zero mean amplitude value, using a 8th order power function between 0 and 100.

   In both cases, the weight gives more importance to samples far from the zero mean value. In the second case, the central peak is ignored.

4. The score, ranging from 0 to 100, is the sum of the squared weighted kernel density without the central peak, normalized by the sum of the squared full weighted kernel density. The score is 0 if there is no additional peak beyond the central peak.
5. The trace is considered clipped if the score is above the `clipping_score_threshold` config parameter.

See the `clipping_detection.clipping_score()` function for more details.

Fig. 1: Example debug plot for the "Clipping Score" algorithm. The score is the sum of the squared weighted kernel density without the central peak (shaded area), normalized by the sum of the squared full weighted kernel density (green curve). The largest the peaks far from the central peak, the higher the score.

4.2 "Clipping Peaks" algorithm

The "Clipping Peaks" algorithm checks if a trace is clipped, based on the number of peaks in the kernel density estimation of the trace amplitude values.

The algorithm is based on the following steps:

1. The trace is demeaned.
2. A kernel density estimation is computed on the trace amplitude values.
3. The kernel density estimation is weighted by the distance from the zero mean amplitude value, using a parabolic function, between 1 and 5.
4. Peaks are detected in the weighted kernel density estimation. The sensitivity of the peak detection algorithm is controlled by the `clipping_peaks_sensitivity` parameter, based on which a minimum prominence threshold is set.
5. The trace is considered clipped if there is at least one peak in the amplitude range corresponding to the `clipping_peaks_percentile` parameter.

See the *`clipping_detection.clipping_peaks()`* function for more details.

Fig. 2: Example debug plot for the "Clipping Peaks" algorithm. If there is at least one peak in the amplitude range corresponding to the `clipping_peaks_percentile` (yellow areas), the trace is considered clipped.

CHAPTER

FIVE

SPECTRAL INVERSION, QUALITY, AND UNCERTAINTY

5.1 Inversion algorithms

The spectral model in SourceSpec is defined by three main parameters: M_w (moment magnitude), $f_c^{p|s}$ (corner frequency for P or S waves), and t^* (attenuation).

Note

If a prior attenuation model is provided (see the `Q_model` parameter in *Configuration File*), then t^* is not inverted and instead derived from the attenuation model (see *Theoretical Background*).

See the *Theoretical Background* section for details on the spectral model.

Different algorithms can be used to invert for these parameters. The algorithm is chosen with the `inv_algorithm` option in the configuration file (see *Configuration File*). The inversion is performed in moment magnitude M_w units (logarithmic amplitude).

Available inversion algorithms:

- **TNC**: Truncated Newton algorithm (with bounds).
- **LM**: Levenberg–Marquardt algorithm. If bounds are provided, the Trust Region Reflective algorithm is used instead.
- **BH**: Basin-hopping algorithm.
- **GS**: Grid search.
- **IS**: Importance sampling of the misfit grid, using a k-d tree.

5.2 Inversion weighting

The inversion can be weighted using different schemes, selected with the `weighting` option in the configuration file (see *Configuration File*).

Weights are normalized between 0 and 1 and are shown in the output figures (if plotting is enabled).

Available weighting schemes:

- **'noise'**: Applies weights based on spectral signal-to-noise ratio.
- **'frequency'**: Applies a constant weight for $f \leq f_{\text{weight}}$; for $f > f_{\text{weight}}$, a weight of 1 is applied. Controlled by the parameters `f_weight` and `weight`.

- **'inv_frequency'**: Computes weights as $1/(f - f_0 + 0.25)^{0.25}$ for $f \leq f_1$. Weights are set to 0 for $f < f_0$ or $f > f_1$. Here, `f_0` and `f_1` are the first and last frequencies where the signal-to-noise ratio exceeds 3 (or the spectrum bounds if no noise window is available).
- **'no_weight'**: No weighting (all weights = 1).

5.3 Spectrum quality

The quality of each spectrum, for each component x (e.g., Z, N, E), is evaluated from the average spectral signal-to-noise ratio ($SSNR_x$):

$$SSNR_x = \frac{1}{f_{max} - f_{min}} \int_{f_{min}}^{f_{max}} \frac{S_x(f)}{N_x(f)}\, df$$

where $S_x(f)$ and $N_x(f)$ are the signal and noise amplitude spectra for component x, respectively, and f_{min} and f_{max} define the frequency range over which the SSNR is computed (either the full frequency band or a user-defined range, see `spectral_sn_freq_range` in *Configuration File*).

A minimum threshold for $SSNR_x$ can be set with `spectral_sn_min` (see *Configuration File*). Component spectra below this threshold are excluded from the inversion.

From $SSNR_x$, two quality parameters are derived:

1. **SSNR mean**: Mean of $SSNR_x$ across the station components.
2. **SSNR max**: Maximum of $SSNR_x$ across the station components.

These values are reported in the output YAML file and in the SQLite database (if enabled). See *Output File Formats* for details.

5.4 Quality of spectral fit

5.4.1 Normalized root mean square error (RMSN)

The fit between observed and synthetic spectra is evaluated with the normalized root mean square error (RMSN):

$$RMSN = \frac{RMS}{\sigma_w}$$

where RMS is the root mean square error between observed and synthetic spectra, and σ_w is the weighted standard deviation of the observed spectrum.

The RMS is given by:

$$\sqrt{\frac{1}{n} \sum_{i=1}^{n} \left(S_{obs}(f_i) - S_{syn}(f_i)\right)^2}$$

with $S_{obs}(f_i)$ and $S_{syn}(f_i)$ the observed and synthetic amplitudes at frequency f_i, and n the number of frequency samples.

The weighted standard deviation σ_w is:

$$\sqrt{\frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} w_i \left(S_{obs}(f_i) - \overline{S_{obs}}\right)^2}$$

where w_i is the weight for each frequency sample.

RMSN values are reported in the output YAML file and SQLite database (if enabled).

5.4.2 Quality of fit

From the RMSN, a fit quality parameter Q_{fit} is defined:

$$Q_{fit} = 100 \times e^{-RMSN}$$

Q_{fit} ranges from 0 to 100, with higher values indicating better fits. It is reported in the output YAML file and SQLite database (if enabled).

The minimum acceptable fit quality can be set with `pi_quality_of_fit_min` (see *Configuration File*). Fits below this threshold are excluded from the final results.

5.5 Spectral parameters: single spectra and event summaries

For each inverted spectrum, SourceSpec computes the following parameters (see *Theoretical Background* for details):

- **Moment magnitude** (M_w): obtained directly from inversion.
- **Corner frequency** (f_c): obtained directly from inversion.
- **Attenuation parameter** (t^*): obtained directly from inversion (or derived from prior attenuation model, see note above).
- **Radiated energy** (E_r): measured directly from the spectrum.
- **Seismic moment** (M_0): derived from M_w.
- **Source radius** (a): derived from M_0 and f_c.
- **Static stress drop** ($\Delta\sigma$): derived from M_0 and a.
- **Apparent stress** (σ_a): derived from E_r and M_0.
- **Quality factor** (Q): derived from t^* (or fixed from prior attenuation model, see note above).

For each spectrum, both the parameter estimates and their uncertainties (from the chosen inversion algorithm) are stored in the output YAML file and in the SQLite database (if enabled). See *Output File Formats* for details.

In addition, **event-level summary values** and associated uncertainties are computed across spectra using the mean, weighted mean, or user-specified percentiles. Outliers are rejected using the interquartile range (IQR) method (see the parameter `nIQR` in the configuration file).

A boxplot of the distribution of spectral parameters across stations is produced if plotting is enabled.

5.6 General quality of the inversion

The overall reliability of the inversion is characterized by several summary parameters:

- **Number of spectra inverted**: The number of spectra that passed quality checks and contributed to the inversion.
- **Primary azimuthal gap**: the largest gap in station azimuthal coverage; measures the overall quality of the distribution.
- **Secondary azimuthal gap**: the largest gap that would remain if any one station were removed; measures the robustness of the coverage.
- **Mean RMSN**: The average normalized root mean square error (RMSN) across all inverted spectra (see the RMSN definition above).
- **Mean fit quality**: The average spectral fit quality (Q_{fit}) across all inverted spectra (see the definition of Q_{fit} above).

- **Spectral dispersion (RMSN)**: The normalized RMSN quantifying how much the observed spectra deviate from the event summary spectrum (see details below).
- **Spectral dispersion score**: A quality score derived from the spectral dispersion RMSN (see details below).

All these parameters are written to the output YAML file and the SQLite database (if enabled).

5.6.1 Spectral dispersion (RMSN)

The spectral dispersion RMSN quantifies how much individual spectra deviate from the event summary spectrum. It is defined as:

$$RMSN_{disp} = \frac{RMS_{disp}}{IPR_{80}}$$

Here, RMS_{disp} is the root mean square error between all observed spectra and the event summary spectrum:

$$RMS_{disp} = \sqrt{\frac{1}{mn} \sum_{j=1}^{n} \sum_{i=1}^{m} \left(S_{obs,j}(f_i) - S_{event}(f_i)\right)^2}$$

where:

- $S_{obs,j}(f_i)$ = observed amplitude of spectrum j at frequency f_i,
- $S_{event}(f_i)$ = amplitude of the event summary spectrum at frequency f_i,
- m = number of frequency samples per spectrum,
- n = number of spectra.

The denominator, IPR_{80}, is the 80% inter-percentile range of all data points in the observed spectra, i.e., the range between the 10th and 90th percentiles.

5.6.2 Spectral dispersion score

This score is derived from the spectral dispersion RMSN ($RMSN_{disp}$) and ranges from 0 to 100, with higher values indicating better overall fit among spectra.

It is defined as:

$$Score_{disp} = 100 \times e^{-RMSN_{disp}}$$

CHAPTER

SIX

CONFIGURATION FILE

Configuration file (default name: `source_spec.conf`) is a plain text file with keys and values in the form `key = value`. Comment lines start with #.

Here is the default config file, generated through `source_spec -S`:

```
# Config file for source_spec

# GENERAL PARAMETERS --------
# All the fields are optional.
# The filled in fields will be written to output files.
# Author information
author_name = None
author_email = None
# Agency information
agency_full_name = None
agency_short_name = None
agency_url = None
# the logo can be a local file (it will be copied to the output dir)
# or a URL
agency_logo = None
# -------- GENERAL PARAMETERS

# TRACE AND METADATA PARAMETERS --------
# Channel naming for mis-oriented channels (vertical, horiz1, horiz2):
# Example:
#   mis_oriented_channels = Z,1,2
mis_oriented_channels = None

# Option to specify non standard instrument codes (e.g., "L" for accelerometer)
instrument_code_acceleration = None
instrument_code_velocity = None

# For more complex network.station.location.channel (SCNL) naming scenarios,
# you can provide a file, in json format, with traceid (SCNL) mapping
traceid_mapping_file = None

# List of traceids to ignore.
# Use network.station.location.channel; wildcards are accepted
# Example:
#   ignore_traceids = FR.CIEL.*.*, AM.RA0D3.00.*
ignore_traceids = None
```

(continues on next page)

(continued from previous page)

```
# List of traceids to use.
# Use network.station.location.channel; wildcards are accepted
# Example:
#   use_traceids = FR.CIEL.*.*, AM.RA0D3.00.*
use_traceids = None

# Epicentral distance ranges (km) to select stations to be processed.
# Use a list of alternating min/max values, ex.:
#   to only use stations between 0 and 100 km:
#       epi_dist_ranges = 0, 100
#   to avoid teleseismic distances between 14° (1300 km) and 29° (3200 km)
#   where the P-wave undergoes travel time triplications:
#       epi_dist_ranges = 0, 1300, 3200, 999999
# Leave it to None to use all stations.
epi_dist_ranges = None

# Directory or single file name containing station metadata
# (instrument response and station coordinates).
# Note: this parameter can be overridden by the command line option
#       with the same name.
# Station metadata files can be in one of the following formats:
#   StationXML, dataless SEED, SEED RESP, PAZ (SAC polezero format)
# Notes:
# 1. SourceSpec will not enter in subdirectories of the given directory
#    (only one level allowed)
# 2. Traceid for PAZ files is specified through their name.
#    The traceid (network.station.location.channel) must be in the last four
#    fields (separated by a dot ".") before the file suffix (which can be
#    ".paz", ".pz", or no suffix).
#    Example:
#      PREFIX.NET.STA.LOC.CHAN.paz
#    or (no prefix):
#      NET.STA.LOC.CHAN.paz
#    or (no prefix and no suffix):
#      NET.STA.LOC.CHAN
# 3. If no traceid is specified through the PAZ file name, then it is assumed
#    that this is a generic PAZ, valid for all the stations that do not have
#    a specific PAZ. Use "trace_units" below to specify the units of the
#    generic PAZ.
# 4. SEED RESP and PAZ files do not contain station coordinates, which
#    should therefore be in the trace header (traces in SAC format)
station_metadata = None

# It is also possible to provide a constant sensitivity (i.e., flat instrument
# response curve) as a numerical value or a combination of SAC header fields
# (in this case, traces must be in SAC format).
# This parameter overrides the response curve computed from station_metadata.
# Leave it to None to compute instrument response from station_metadata.
# Examples:
#  sensitivity = 1
#  sensitivity = 1e3
```

(continues on next page)

(continued from previous page)

```
#  sensitivity = resp0
#  sensitivity = resp1*resp2
#  sensitivity = user3/user2
sensitivity = None

# SQLite database file for storing output parameters (optional):
database_file = None

# Correct_instrumental_response (optional, default=True):
correct_instrumental_response = True

# Trace units.
# Leave it to 'auto' to let the code decide, based on instrument type.
# Manually set it to 'disp', 'vel' or 'acc' if you have already preprocessed
# the traces.
trace_units = auto
# -------- TRACE AND METADATA PARAMETERS


# EARTH MODEL PARAMETERS --------
# General Earth model parameters.
# The parameters are defined in terms of vp (km/s), vs (km/s)
# and density (rho, kg/m3).
# These values can be provided either as single values or as layered models
# (list of values). In the latter case, the top depths of each layer
# (in km, positive down) must be provided as well.
#
# Earth model parameters are used to compute theoretical P and S
# arrival times as well as coefficients for converting station displacements
# to seismic moment.
#
# If any of the parameters below is set to None, values from the global
# Earth model "iasp91" are used.
#
# Example of layered model:
#   vp = 5.30, 5.60, 6.20, 6.90, 7.40, 7.70, 7.90, 8.10, 8.30
#   vs = 3.01, 3.18, 3.52, 3.92, 4.20, 4.37, 4.49, 4.60, 4.72
#   rho = 2520, 2610, 2780, 2970, 3120, 3200, 3260, 3320, 3370
#   layer_top_depths = 0.0, 4.0, 9.0, 14.0, 19.0, 24.0, 33.0, 49.0, 66.0
vp = 5.5
vs = 3.2
rho = 2500
layer_top_depths = None
# Source-specific parameters.
# You can specify different Earth model parameters near the source.
# These values can be provided either as single values or as layered models
# (list of values). In the latter case, the top depths of each layer
# (in km, positive down) must be provided as well.
# If any of the parameters below is set to None, then general Earth model
# values (defined above) are used.
vp_source = None
vs_source = None
```

(continues on next page)

(continued from previous page)

```
rho_source = None
layer_top_depths_source = None
# Station-specific parameters.
# You can specify different Earth model parameters near the stations.
# If any of the parameters below is set to None, then general Earth model
# values (defined above) are used.
# Note:
#   Station-specific parameters cannot be layered models.
vp_stations = None
vs_stations = None
rho_stations = None
# NonLinLoc models.
# As an alternative, NonLinLoc valocity and travel time models can be
# specified.
# NLL_time_dir: directory containing NonLinLoc travel time grids.
# Theoretical travel times will be read from these grids, instead of being
# computed from the vp and vs values defined above.
# Note:
#   Reading NonLinLoc grids takes time. For simple 1D models, you
#   can speed up considerably the process using a generic station
#   named "DEFAULT". The coordinates of this default station are not important,
#   since they will be superseded by each station's coordinates.
NLL_time_dir = None
# NLL_model_dir: directory containing a NonLinLoc velocity model.
# This model will be used to read vp, vs values near the source and stations,
# instead of using the values defined above.
# Important note:
#   NonLinLoc velocity models do not contain density values. Therefore,
#   when using a NonLinLoc velocity model, density values must be provided
#   through the parameters defined above (rho, rho_source, rho_stations).
NLL_model_dir = None
# -------- EARTH MODEL PARAMETERS


# TIME WINDOW PARAMETERS --------
# Arrival tolerances (in seconds) to accept a manual P or S pick
p_arrival_tolerance = 4.0
s_arrival_tolerance = 4.0

# Refine theoretical P and S arrival using a simple autopicker
# based on the analysis of the smoothed envelope of the trace.
# Note: manual picks are left unchanged
refine_theoretical_arrivals = False
# Minimum frequency (Hz) for the autopicker.
# Trace will be high-pass filtered at this frequency.
autopick_freqmin = 1.0
# Plot a debug figure for each trace with the results of the autopicker
# Note: the figures are always shown, even if "plot_show" is False (see below)
autopick_debug_plot = False

# Start time (in seconds) of the noise window, respect to the P window
# If None, it will be set to the length of the signal (P or S) window plus
```

(continues on next page)

(continued from previous page)

```
# the value of "signal_pre_time" (see below)
noise_pre_time = None

# Start time (in seconds) of the signal window, respect to the P or S arrival
# times (see "wave_type" below)
signal_pre_time = 1.0

# Length (in seconds) for both noise and signal windows.
# Notes:
# 1. If the weighting mode is 'noise' (see INVERSION PARAMETERS below)
#    and the trace starts too late to include the full noise window,
#    the noise window will be shortened to fit the available data,
#    and the signal window will be truncated to match the shortened
#    noise window length.
# 2. If the weighting mode is 'inv_frequency' (see INVERSION PARAMETERS below)
#    and the trace starts too late to include the full noise window,
#    the noise window will be zero-padded to match the signal window length.
# 3. If 'force_noise_zero_padding' below is set to True, the noise window will
#    always be zero-padded to match the signal window length, regardless
#    of the weighting mode.
win_length = 5.0
# Force zero-padding of the noise window to match the signal window length
# Useful when the dataset comprises a mixture of traces with sufficient
# noise length and traces with too short or missing noise windows
# (e.g., triggered records)
force_noise_zero_padding = False
# Variable window length factor (fraction of travel time):
#   win_length = max(win_length, variable_win_length_factor * travel_time)
# Set to None to disable variable window length.
variable_win_length_factor = None
# Minimum allowable window length (in seconds) for both signal and noise
# windows. This parameter acts as a lower bound when the window length is
# determined automatically, either through the 'noise' weighting mode or
# when the 'variable_win_length_factor' parameter is set (see above).
# Traces having a window length smaller than this value will be skipped.
win_length_min = None
# -------- TIME WINDOW PARAMETERS


# SPECTRUM PARAMETERS --------
# Wave type to analyse: 'P', 'S', 'SH' or 'SV'
# If 'SH' or 'SV' are selected, traces are rotated in the radial-transverse
# system. Transverse component is used for 'SH', radial component (and
# optionally the vertical component, see 'ignore_vertical' below) is used
# for 'SV'
wave_type = S

# Integrate in time domain (default: integration in spectral domain)
time_domain_int = False

# Ignore vertical components when building S or SV spectra
# Note: this option has no effect when 'wave_type' is 'P' (the vertical
```

(continues on next page)

(continued from previous page)

```
# component is not ignored) and when 'wave_type' is 'SH' (the vertical
# component is not needed)
ignore_vertical = False

# Taper half width: between 0 (no taper) and 0.5
taper_halfwidth = 0.05

# Spectral window length (seconds)
# Signal is tapered, and then zero padded to
# this window length, so that the spectral
# sampling is fixed to 1/spectral_win_length.
# Comment out (or set to None) to use
# signal window as spectral window length.
spectral_win_length = None

# Spectral smoothing window width in frequency decades
# (i.e., log10 frequency scale).
# Example:
#  spectral_smooth_width_decades=1 means a width of 1 decade
#  (generally, too large, producing a spectrum which is too smooth).
#  spectrum(f0) is smoothed using values between f1 and f2, so that
#  log10(f1)=log10(f0)-0.5 and log10(f2)=log10(f0)+0.5
#    i.e.,
#  f1=f0/(10^0.5) and f2=f0*(10^0.5)
#    or,
#  f2/f1=10 (1 decade width)
# Default value of 0.2 is generally a good choice
spectral_smooth_width_decades = 0.2

# Residuals file path
# An HDF5 file with the mean residuals per station, used for station
# correction. This file is generally created using the command
# "source_residuals" on a previous SourceSpec run.
residuals_filepath = None

# Remove the signal baseline after instrument correction and before filtering
remove_baseline = False

# Band-pass frequencies (Hz) for accelerometers, velocimeters
# and displacement sensors.
# Use bp_freqmin_STATION and bp_freqmax_STATION to provide
# filter frequencies for a specific STATION code.
# TODO: calculate from sampling rate?
bp_freqmin_acc    = 1.0
bp_freqmax_acc    = 50.0
bp_freqmin_shortp = 1.0
bp_freqmax_shortp = 40.0
bp_freqmin_broadb = 0.5
bp_freqmax_broadb = 40.0
bp_freqmin_disp   = 0.5
bp_freqmax_disp   = 40.0
```

(continues on next page)

(continued from previous page)

```
# Spectral windowing frequencies (Hz) for accelerometers, velocimeters
# and displacement sensors.
# (spectra will be cut between these two frequencies)
# Use freq1_STATION and freq2_STATION to provide
# windowing frequencies for a specific STATION code.
freq1_acc     = 1.0
freq2_acc     = 30.0
freq1_shortp  = 1.0
freq2_shortp  = 30.0
freq1_broadb  = 0.5
freq2_broadb  = 30.0
freq1_disp    = 0.5
freq2_disp    = 30.0

# Save the spectra to an HDF5 file in the output directory
save_spectra = False
# -------- SPECTRUM PARAMETERS


# SIGNAL/NOISE PARAMETERS --------
# Minimum rms (in trace units before instrument corrections)
# to consider a trace as noise
rmsmin = 0

# Time domain S/N ratio min
# Note: in many cases, especially for small earthquakes recorded at broadband
# sensors, it is better to leave this parameter to 0 (or 1) and use the
# spectral S/N ratio instead (see "spectral_sn_min" below). This beacause time
# domain S/N ratio can be close to 1 due to strong microseismic noise.
# Band-limited noise is naturally handled by the spectral S/N ratio.
sn_min = 0

# Clipping detection algorithm
# Options:
#  - 'none': no clipping detection
#  - 'clipping_score': compute a clipping score for each trace, based on the
#    shape of the kernel density estimation of the trace amplitude values.
#    A high clipping score will be obtained for traces with a high number of
#    samples whose amplitude is close to the trace highest or lowest
#    amplitude values. Clipping scores for each trace are printed on the
#    terminal and in the log file.
#    Note: if "remove_baseline" is True (see above), clipping scores are
#    computed on the baseline-corrected signal.
#  - 'clipping_peaks': count the number of peaks in the kernel density
#    estimation of the trace amplitude values. The trace is considered clipped
#    if at least one peak is found within the trace highest or lowest amplitude
#    values. Kernel density peaks for each trace are printed on the terminal
#    and in the log file.
clipping_detection_algorithm = clipping_score
# Minimum ratio between the overall sensitivity of the instrument and the
# maximum amplitude of the trace (both in counts), to actually perform the
# clipping check.
```

(continues on next page)

(continued from previous page)

```
# Traces with max_amp < (sensitivity / clipping_min_amplitude_ratio) will not
# be checked for clipping.
# It must be a strictly positive value. Default is None, meaning no amplitude
# check is performed.
clipping_min_amplitude_ratio = None
# Plot a debug figure for each trace with the results of the clipping algorithm
# Note: the figures are always shown, even if "plot_show" is False (see below)
clipping_debug_plot = False
# Threshold for the 'clipping_score' algorithm (between 0 and 100).
# A value of 100 means no clipping detection.
# This parameter is ignored if "clipping_detection_algorithm" is not set to
# 'clipping_score'.
clipping_score_threshold = 10
# Sensitivity for the 'clipping_peaks' algorithm (between 1 and 5).
# Higher values mean more peaks are detected.
# This parameter is ignored if "clipping_detection_algorithm" is not set to
# 'clipping_peaks'.
clipping_peaks_sensitivity = 3
# Trace amplitude percentile for the 'clipping_peaks' algorithm (between 0
# and 100). Example:
#   clipping_peaks_percentile = 10
# means that the 10% highest and lowest values of the trace amplitude will be
# checked for clipping.
# A value of 0 means that no clipping check will be performed.
# This parameter is ignored if "clipping_detection_algorithm" is not set to
# 'clipping_peaks'.
clipping_peaks_percentile = 10

# Maximum gap length for the whole trace, in seconds
gap_max = None
# Maximum overlap length for the whole trace, in seconds
overlap_max = None

# Minimum average spectral S/N ratio, below which a spectrum will be skipped
# This parameter should be preferred to the time domain S/N ratio
# ("sn_min", above) since it better handles band-limited noise.
spectral_sn_min = 0
# Frequency range (Hz) to compute the average spectral S/N ratio
# (comment out or use None to indicate the whole frequency range)
# Example:
#  spectral_sn_freq_range = 0.1, 2
spectral_sn_freq_range = None
# -------- SIGNAL/NOISE PARAMETERS


# SPECTRAL MODEL PARAMETERS --------
# Free-surface amplification factor
# Can be specified as either:
#   - A single positive float value applied to all stations, or
#   - A list of station code patterns with corresponding positive
#     float factors. Wildcards '*' (e.g., NET1.STA1.*) are allowed;
#     a lone '*' matches all remaining stations.
```

(continues on next page)

(continued from previous page)

```
# Examples:
#   free_surface_amplification = 2.0
#   free_surface_amplification = NET1.STA1.*: 1.0, NET2.*: 1.5, *: 2.0
free_surface_amplification = 2.0
# Geometrical spreading correction of wave amplitude.
# Spectra will be multiplied by this value to correct for the lost amplitude.
# Possible options are:
#    'r_power_n':   "r" to the power of "n" (r).
#                   You must provide the value of the exponent "n"
#                   (see "geom_spread_n_exponent" below).
#    'r_power_n_segmented': piecewise continuous r  function,
#                   cf. Atkinson & Boore (1995), Boore (2003)
#                   You must provide the values of the exponent "n"
#                   (see "geom_spread_n_exponents" below), as well as
#                   the distances separating the segments
#                   (see "geom_spread_n_disntaces" below)
#    'boatwright':  "r" (body waves) geometrical spreading for hypocentral
#                   distances below a cutoff distance; frequency-dependent
#                   geometrical spreading above the cutoff distance (Boatwright
#                   et al., 2002). You must provide the cutoff distance (see
#                   "geom_spread_cutoff_distance" below). This coefficient can
#                   be a valid choice for regional distances (up to 200 km),
#                   where S-waves, Lg waves and surface waves are mixed.
geom_spread_model = r_power_n
# Exponent "n" for the "r_power_n" geometrical spreading coefficient (positive
# float). Examples:
#   geom_spread_n_exponent = 1 (default, body wave in a homogeneous full-space)
#   geom_spread_n_exponent = 0.5 (surface wave in a homogeneous half-space)
geom_spread_n_exponent = 1
# Exponents and hypocentral distances (in km) defining different segments
# for the "r_power_n_segmented" geometrical spreading function.
# The number of exponents must be equal to the number of distances.
# Distances correspond to the start of each segment.
# Note that no geometrical spreading correction is considered
# below the smallest distance. This distance is generally set to 1 km.
# Example:
#   geom_spread_n_exponents = 1., 0., 0.5
#   geom_spread_n_distances = 1, 70, 130
geom_spread_n_exponents = None
geom_spread_n_distances = None
# Geometrical spreading cutoff hypocentral distance, in km, for the
# "boatwright" model:
geom_spread_cutoff_distance = 50
# Minimum epicentral distance (in km) to use a teleseismic geometrical
# spreading model. Above this distance, the model from Okal (1992) for body
# waves spreading in a spherically symmetric Earth will be used.
# Set to None to never use the teleseismic geometrical spreading model.
# Note that this model might not be appropriate for very deep events.
geom_spread_min_teleseismic_distance = None
# P-wave average radiation pattern coefficient:
rpp = 0.52
# S-wave average radiation pattern coefficient:
```

(continues on next page)

```
rps = 0.62
# Radiation pattern coefficient from focal mechanism, if available.
#   Note: radiation pattern is computed for the first arriving phase and might
#   not be correct for windows involving multiple phase arrivals (e.g.,
#   Lg waves, surface waves at regional distances, depth phases at teleseismic
#   distances)
rp_from_focal_mechanism = False
# Lower bound on radiation pattern coefficient from focal mechanism
# in order to avoid overestimated station magnitudes close to nodal planes
rp_lower_bound = 0.01
# "kp" and "ks" coefficients to compute source radius a from the P-wave
# corner frequency fc_p or the S-wave corner frequency fc_s and the shear
# wave speed beta ("vs_source"):
#
#   a = kp * beta / fc_p
#   a = ks * beta / fc_s
#
# (Madariaga, 2009; Kaneko and Shearer, 2014)
#
# The default value for S-waves is "ks = 0.3724", obtained by Brune (1970)
# for a static circular crack.
# Other values are discussed in Kaneko and Shearer (2014) for a dynamic
# circular crack, as a function of the ratio Vr/beta, where Vr is the rupture
# speed:
#
#   Vr/beta  kp(K&S)   ks(K&S)   kp(Mada)   ks(Mada)   kp(S&H)   ks(S&H)
#   0.9      0.38      0.26      0.32       0.21       0.42      0.29
#   0.8      0.35      0.26                            0.39      0.28
#   0.7      0.32      0.26                            0.36      0.27
#   0.6      0.30      0.25                            0.34      0.27
#   0.5      0.28      0.22                            0.31      0.24
#
#   K&S: Kaneko and Shearer (2014)
#   Mada: Madariaga (1976)
#   S&H: Sato and Hirasawa (1973)
kp = 0.38
ks = 0.3724
# -------- SPECTRAL MODEL PARAMETERS


# INVERSION PARAMETERS --------
# Weighting type: 'noise', 'frequency', 'inv_frequency' or 'no_weight'
#   'noise':         spectral signal/noise ratio weighting
#   'frequency':     a constant weight is applied for f<=f_weight
#                      a weight of 1 is used for f>f_weight
#                      (see "f_weight" and "weight" below)
#   'inv_frequency': weight is computed as 1/(f-f0+0.25)**0.25 for f<=f1,
#                      weight is 0 for f<f0 and f>f1.
#                      f0 and f1 are the first and last frequencies where
#                      spectral signal/noise ratio is above 3, or the first and
#                      last frequencies of the entire spectrum if no noise window
#                      is available
#   'no_weight':     no weighting
```

(continues on next page)

(continued from previous page)

```
weighting = noise
# Parameters for 'frequency' weighting (ignored for the other weighting types):
#   weight for f<=f_weight (Hz)
#   1      for f> f_weight (Hz)
f_weight = 7.
weight = 10.

# Inversion algorithm:
# TNC: truncated Newton algorithm (with bounds)
# LM: Levenberg-Marquardt algorithm
# (warning: Trust Region Reflective algorithm will be used instead if
#  bounds are provided)
# BH: basin-hopping algorithm
# GS: grid search
# IS: importance sampling of misfit grid, using k-d tree
inv_algorithm = TNC

# Mw initial value and bounds.
# Set to True to use the magnitude (or scalar moment) from event file as
# initial Mw value for the inversion, instead of computing it from the average
# of the spectral plateau.
# If the event file does not contain a magnitude value or a scalar moment,
# then this parameter is ignored
Mw_0_from_event_file = False
# Allowed variability for Mw in the inversion
# (expressed as a fraction of Mw_0, between 0 and 1).
# This parameter is interpreted differently, depending on whether
# Mw_0_from_event_file is True or False:
#   - If Mw_0_from_event_file is True, then Mw_variability is interpreted as
#     the allowed variability around the Mw value provided in the event file.
#   - If Mw_0_from_event_file is False, then the Mw bounds are defined as
#       Mw_min = min(Mw(f))*(1-Mw_0_variability)
#       Mw_max = max(Mw(f))*(1+Mw_0_variability),
#     where Mw(f) is the low frequency spectral plateau in magnitude units.
#     If noise weighting is used, frequencies for which
#     S/N(f) < 0.5*max(S/N(f)) will be ignored, where S/N(f) is the spectral
#     signal to noise ratio.
Mw_0_variability = 0.1

# Bounds for fc (Hz)
# Specify bounds as a list, ex.:
#   fc_min_max = 0.1, 40
# Note:
#    If not specified, fc bounds will be autoset to fc0/10 and fc0*10, i.e. two
#    decades around fc0. The value of fc0 is set as the first maximum of
#    spectral S/N (noise weighting), or at "f_weight" (frequency weighting),
#    or at frequency where weight is 30% below the maximum (inverse-frequency
#    weighting) or at half of the frequency window (no weighting)
fc_min_max = None

# Initial value and bounds for t_star (seconds)
t_star_0 = 0.045
```

(continues on next page)

(continued from previous page)

```
# Try to invert for t_star_0.
# If False, then the fixed t_star_0 defined above will be used.
# If the inverted t_star_0 is non-positive, then fixed t_star_0 will be used
invert_t_star_0 = False
# Allowed variability around inverted t_star_0 in the inversion
# (expressed as a fraction of t_star_0, between 0 and 1).
# If the inverted t_star_0 is non-positive, then t_star_min_max is used
# (see below).
t_star_0_variability = 0.1
# t_star_min_max does not supersede t_star_0_variability
t_star_min_max = 0.001, 0.25
# optional : Qo bounds (converted into t_star bounds in the code).
# (comment out or use None to indicate no bound)
# Note: if you want to explore negative t_star values, you have to specify
# -Qo_min, Qo_min. This because t_star is proportional to 1/Qo.
# Example, for searching only positive t_star values:
#   Qo_min_max = 10, 1000
# If you want to search also negative t_star values:
#   Qo_min_max = -10, 10
Qo_min_max = None
# Prior knowledge of the attenuation model for your region can be specified
# through Q_model as either a fixed value or a frequency-dependent function.
# Use 'f' to represent frequency in the expression.
# The frequency dependence is commonly expressed as:
#   Q_model = Qo * (f/fo)**alpha
# where:
#   - Qo is the quality factor at the reference frequency fo
#   - fo is the reference frequency (typically 1 Hz), to keep Q_model unitless
#   - alpha is the frequency exponent
# Any functional form can be used, provided Q_model remains unitless.
# When Q_model is specified, t_star will not be inverted and all t_star-related
# inversion parameters above will be ignored.
# Examples:
#   Fixed value:
#       Q_model = 500
#   Frequency-dependent (assuming fo=1 Hz):
#       Q_model = 529 * f**0.42
Q_model = None
# -------- INVERSION PARAMETERS

# POST-INVERSION CONSTRAINTS --------
# Post-inversion constraints are applied AFTER the inversion process,
# on a per-station basis. They help filter out unreliable or physically
# implausible solutions without overly constraining the inversion itself.
#
# Rationale:
# - Inversion often benefits from being more permissive during fitting.
# - Instead of forcing strict bounds during the inversion (which can
#   prevent convergence), it can be more effective to reject "bad"
#   solutions afterwards using post-inversion checks.
#
# Corner frequency (fc) bounds, in Hz
```

(continues on next page)

(continued from previous page)

```
pi_fc_min_max = None
# Corner frequency weight constraint:
# Minimum acceptable weight value near fc. Choose a value between 0 and 1.
# A weight lower than pi_fc_weight_min indicates that fc lies in a poorly
# constrained part of the spectrum and the solution should be rejected.
pi_fc_weight_min = 0
# t_star (attenuation parameter) bounds, in seconds
pi_t_star_min_max = None
# Static stress drop (SSD) bounds, in MPa
pi_ssd_min_max = None
# Minimum acceptable spectral quality of fit, in percentage
# (100% = perfect fit).
# Reject solutions where the quality of fit is too low.
pi_quality_of_fit_min = None
# -------- POST-INVERSION CONSTRAINTS


# RADIATED-ENERGY PARAMETERS --------
# Minimum and maximum frequency (Hz) to measure radiated energy Er
# Examples:
#   Set min and max frequency to the "noise limits"
#   (i.e. the frequency range where spectral signal/noise ratio is above 3):
#       Er_freq_range = noise, noise
#   Use the whole spectrum:
#       Er_freq_range = None
#           or
#       Er_freq_range = None, None
#   Use the lowest possible frequency, and set the max frequency
#   to the "noise limit":
#       Er_freq_range = None, noise
#   Use frequencies between 1 and 10 Hz
#       Er_freq_range = 1, 10
#   Use frequencies between 1 and the "noise limit"
#       Er_freq_range = 1, noise
#
# The finite-band correction of Di Bona & Rovelli (1988) will be applied
# to account for the missing energy above the maximum frequency.
Er_freq_range = None, None
# -------- RADIATED-ENERGY PARAMETERS


# LOCAL MAGNITUDE PARAMETERS --------
compute_local_magnitude = False
# Local magnitude parameters:
#   ml = log10(A) + a * log10(R/100) + b * (R-100) + c
# where A is the maximum W-A amplitude (in mm)
# and R is the hypocentral distance (in km)
# Default values (for California) are:
#   a = 1., b = 0.00301, c = 3.
a = 1.
b = 0.00301
c = 3.
```

(continues on next page)

(continued from previous page)

```
# Band-pass filtering frequencies (Hz) for local magnitude
ml_bp_freqmin = 0.1
ml_bp_freqmax = 20.0
# -------- LOCAL MAGNITUDE PARAMETERS


# SUMMARY STATISTICS PARAMETERS --------
# For each spectral parameter, SourceSpec computes three different summary
# estimates (from station estimates), using the following statistics:
#  - mean
#  - weighted_mean
#  - percentiles
# All the three summary estimates are stored in the YAML and SQLite output,
# but only a reference one is used for map plots, QuakeML and HYPO output,
# as well as for the "Event Summary" section in HTML report and for computing
# station spectral residuals.
# Use the parameter "reference_statistics" to specify the reference summary
# statistics that will be used in the cases described above.
reference_statistics = weighted_mean
# Number of sigmas (standard deviations) for average and weighted average
# uncertainty estimation
n_sigma = 1
# Percentage levels to compute lower, mid and upper percentiles
#   Example: to mimic a Gaussian distribution (one-sigma, 68.2% confidence):
#       lower_percentage = 15.9
#       mid_percentage = 50
#       upper_percentage = 84.1
# Note: the confidence level is upper_percentage - lower_percentage
lower_percentage = 15.9
mid_percentage = 50
upper_percentage = 84.1
# Reject outliers before computing means (standard and weighted),
# using the IQR method.
# IQR is the interquartile range Q3-Q1, where Q1 is the 25% percentile
# and Q3 is the 75% percentile.
# Values that are smaller than (Q1 - nIQR*IQR) or larger than (Q3 + nIQR*IQR)
# will be rejected as outliers.
# Set nIQR to None to disable outlier rejection.
# Note: this parameter also controls the position of "whiskers" on the source
# parameter box plots.
nIQR = 1.5
# -------- SUMMARY STATISTICS PARAMETERS


# PLOT PARAMETERS --------
# Show interactive plots (slower)
plot_show = False
# Save plots to disk
plot_save = True
# Save trace and spectrum plots as soon as they are ready.
# This uses less memory but slows down the code.
plot_save_asap = False
```

(continues on next page)

(continued from previous page)

```
# Plot file format: 'png', 'pdf', 'pdf_multipage' or 'svg'
plot_save_format = png
# Plots an extra synthetic spectrum with no attenuation
plot_spectra_no_attenuation = False
# Plots an extra synthetic spectrum with no fc
plot_spectra_no_fc = False
# Max number of rows in plots
plot_spectra_maxrows = 3
plot_traces_maxrows = 3
# Plot ignored traces (clipped or low S/N)
plot_traces_ignored = True
# Plot ignored spectra (low S/N)
plot_spectra_ignored = True
# Plot station map
plot_station_map = False
# Map style (for regional maps)
#   Options: 'hillshade', 'hillshade_dark', 'ocean', 'satellite',
#            'stamen_terrain', 'no_basemap', 'geotiff'
#   All basemap are from Esri, except 'stamen_terrain' which is from Stamen.
#   Use 'geotiff' to plot a GeoTIFF file as a basemap
#   (see "plot_map_geotiff_filepath" below).
#   Notes:
#     1. The map style is only used for regional maps.
#        At teleseismic distances, the global map will alyaws use the
#        Natural Earth basemap.
#     2. For the 'stamen_terrain' basemap, you need a (free) API key from
#        Stadia Maps, see https://stadiamaps.com
plot_map_style = no_basemap
# API key for the 'stamen_terrain' basemap
#   Note: for privacy reasons, this parameter is not transcripted to the
#   output config file.
plot_map_api_key = None
# Alternatively, use a GeoTIFF file as a basemap
plot_map_geotiff_filepath = None
# Plot GeoTIFF in grayscale
plot_map_geotiff_grayscale = False
# GeoTIFF attribution text
plot_map_geotiff_attribution = None
# Plot station names on map
plot_station_names_on_map = False
# Text size for station names
plot_station_text_size = 8
# Coastline resolution
# Use None to let the code autoset the coastline resolution.
# Otherwise choose one of:
#   'full', 'high', 'intermediate', 'low', 'crude', 'no_coastline'
plot_coastline_resolution = None
# Zoom level for map tiles
# Use None to let the code autoset the zoom level
# Otherwise choose an integer between 1 (minimum zoom) and 18 (maximum zoom)
# Note: for zoom levels larger than 11, some map tiles could be missing
plot_map_tiles_zoom_level = None
```

(continues on next page)

(continued from previous page)

```
# -------- PLOT PARAMETERS


# HTML REPORT --------
# Generate an HTML page summarizing the results of this run
# Note: "plot_save_format" (above) must be "png" or "svg"
html_report = False
# Link to event page. If set, the event ID on the HTML page will be a link to
# the event page. You can use the following placeholders:
# - $EVENTID: the event ID
# - $YEAR: the year of the event
# - $MONTH: the month of the event
# - $DAY: the day of the event
# Examples:
#   event_url = https://earthquake.usgs.gov/earthquakes/eventpage/$EVENTID/executive
#   event_url = http://bbnet.gein.noa.gr/Events/$YEAR/$MONTH/$EVENTID_info.html
event_url = None
# -------- HTML REPORT

# QUAKEML INPUT PARAMETERS --------
# Parameters for QuakeML input.
# Set "qml_event_description" to True, if you want to obtain the event name
# from the QuakeML event "description" tag
qml_event_description = False
# If "qml_event_description" is True, then the following parameter can be used
# to define a regular expression to extract the event name from the QuakeML
# event "description" tag.
# Examples:
#   - For QuakeML produced by https://api.franceseisme.fr, we want to keep
#     only the string "near of CITY NAME":
#        qml_event_description_regex = 'near of .+'
# Leave to None to use the full description as event name.
qml_event_description_regex = None
# -------- QUAKEML INPUT PARAMETERS


# QUAKEML OUTPUT PARAMETERS ----------------
# Parameters for QuakeML output.
#
# A QuakeML file will be generated only if QuakeML is used for input.
# The output file will be based on the input file, with additional information
# on seismic moment, Mw and source parameters computed by SourceSpec.
# Note: if you don't understand the parameters below, then probably you
# don't need QuakeML output and you can leave all the parameters to their
# default value

# Set SourceSpec Mw as preferred
set_preferred_magnitude = False
# Base for all the object ids (smi)
smi_base = "smi:local"
# String to strip from the Origin id when constructing the
# Magnitude and stationMagnitude ids.
```

(continues on next page)

(continued from previous page)

```
smi_strip_from_origin_id = ""
# Template for the Magnitude object id (smi).
# Use $SMI_BASE to indicate smi_base defined above
# Use $ORIGIN_ID to indicate the id of the associated Origin.
smi_magnitude_template = "$SMI_BASE/Magnitude/Origin/$ORIGIN_ID#sourcespec"
# Template for the stationMagnitude object id (smi).
# Use $SMI_BASE to indicate smi_base defined above
# Use $ORIGIN_ID to indicate the id of the associated Origin.
# Use $SMI_MAGNITUDE_TEMPLATE to reuse the template for Magnitude object
# Use $WAVEFORM_ID to indicate the id of the associated waveform.
smi_station_magnitude_template = "$SMI_MAGNITUDE_TEMPLATE#$WAVEFORM_ID"
# Template for the MomentTensor object id (smi) which is used to store
# the scalar moment value.
# Use $SMI_BASE to indicate smi_base defined above
# Use $ORIGIN_ID to indicate the id of the associated Origin.
smi_moment_tensor_template = "$SMI_BASE/MomentTensor/Origin/$ORIGIN_ID#sourcespec"
# Template for the FocalMechanism object id (smi) which is used to store
# the scalar moment value.
# Use $SMI_BASE to indicate smi_base defined above
# Use $ORIGIN_ID to indicate the id of the associated Origin.
smi_focal_mechanism_template = "$SMI_BASE/FocalMechanism/Origin/$ORIGIN_ID#sourcespec"
# -----------------QUAKEML OUTPUT PARAMETERS
```

CHAPTER

SEVEN

INPUT FILE FORMATS

7.1 Trace formats

SourceSpec can read all the trace formats supported by ObsPy.

Two very common choices are:

- miniSEED
- SAC

The SAC format can carry additional information in its header, like event location and origin time, phase picks, instrument sensitivity.

Input trace files can be provided –through the `-t` option– as a list of files, as a directory containing the files, or as a TAR(GZ) or ZIP archive containing the files.

7.2 Event formats

SourceSpec can read event information (event ID, location, origin time) in the following formats:

- *SourceSpec Event File*: this file can contain additional event information, such as magnitude, moment tensor or focal mechanism
- QuakeML: this file can contain additional event information, such as magnitude, moment tensor or focal mechanism. If phase picks are available, they will be read as well
- HYPO71
- HYPOINVERSE-2000: if phase picks are available, they will be read as well

Event information can also be stored in the SAC file header (header fields: `EVLA`, `EVLO`, `EVDP`, `O`, `KEVNM`).

Note

Any depth value provided in the event file or in the SAC header can be overridden using the command line option `--depth`.

7.3 Phase pick formats

Phase picks for P and S waves can be read from one of the following formats:

- QuakeML
- HYPO71

- HYPOINVERSE-2000

Phase picks can also be stored in the SAC file header, using the header fields `A` and `T0` through `T9`. A pick label can be specified (header fields `KA` and `KT0` through `KT9`) to identify the pick; the pick label can be a standard 4-characters SAC label (e.g., `"IPU0"`, `" S 1"`) or a label starting with `"P"` or `"S"` (lowercase or uppercase, e.g., `"P"`, `"pP"`, `"Pg"`, `"S"`, `"Sn"`). Picks with labels that cannot be parsed by SourceSpec will be ignored. If no label is specified, then SourceSpec will assume that `A` is the P-pick and `T0` is the S-pick.

7.4 Station metadata formats

Station metadata (coordinates, instrumental response) can be provided in one of the following formats:

- StationXML
- Dataless SEED
- SEED RESP
- SAC polezero (PAZ)

Note that SEED RESP and PAZ formats do not contain station coordinates, which should therefore be in the trace header (traces in SAC format).

The station metadata file name or file directory is provided in the configuration file through the parameter `station_metadata`.

Alternatively, instrument sensitivity can be provided in the SAC header or as a constant in the configuration file. In both cases, use the configuration parameter `sensitivity`.

CHAPTER

EIGHT

SOURCESPEC EVENT FILE

The SourceSpec Event File format is a custom format, based on YAML, which allows specifying the source properties for one ore more given events. The following properties can be specified:

- `event_id` (**mandatory**): the event ID.

  > **Note**
  >
  > This field must be preceded by a dash (`-`).

- `name` (*optional*): the event name
- `hypocenter` (**mandatory**): hypocentral location and origin time
- `magnitude` (*optional*): the event magnitude (used when `Mw_0_from_event_file` in the *Configuration File* is set to `True`)

  > **Note**
  >
  > If a scalar moment or a moment tensor is given, a Mw magnitude will be (re)computed from it.

- `scalar moment` (*optional*): event scalar moment (used when `Mw_0_from_event_file` in the *Configuration File* is set to `True`)

  > **Note**
  >
  > If a moment tensor is given, the scalar moment will be (re)computed from it.

- `focal_mechanism` (*optional*): event focal mechanism (used for computing radiation pattern when the option `rp_from_focal_mechanism` in the *Configuration File* is set to `True`).

  > **Note**
  >
  > If a moment tensor is given, the focal mechanism will be (re)computed from it.

- `moment_tensor` (*optional*): event moment tensor (used for computing focal planes and radiation pattern when the option `rp_from_focal_mechanism` in the *Configuration File* is set to `True`; also used for computing moment magnitude, to be used when `Mw_0_from_event_file` is set to `True`)

See the sample SourceSpec Event File below for more details on the format.

8.1 Sample SourceSpec Event File

A sample SourceSpec Event File can be obtained through the command:

```
source_spec --samplesspevent
```

or

```
source_spec -y
```

The content of the sample SourceSpec Event File is shown below:

```
# SourceSpec Event File
#
# One or more events can be defined in this file.
# Each event must have a unique "event_id".
# Each event description starts with a dash (-) and is followed by a mandatory
# "event_id" field. The "hypocenter" field is mandatory, all the other fields
# are optional.
# Optional fields can be empty or commented out.
# The indentation of the fields must be respected.
#
# For a minimal working file, replace the placeholders between "<" and ">" with
# the appropriate values.
#
# This is a YAML file. For more information on the format, see
# https://en.wikipedia.org/wiki/YAML

# Mandatory event_id, preceded by a dash (-)
- event_id: <EVENT_ID>
  # Optional event name
  name:
  # Mandatory hypocenter information
  hypocenter:
    longitude:
      value: <LONGITUDE>
      # currently, only decimal degrees are supported
      units: deg
    latitude:
      value: <LATITUDE>
      # currently, only decimal degrees are supported
      units: deg
    depth:
      value: <DEPTH>
      # units can be one of the following: km, m
      units: km
    origin_time: <ORIGIN_TIME>
  # Optional magnitude value.
  # If a scalar moment or a moment tensor is given, a Mw magnitude will be
  # (re)computed from it
  magnitude:
    value:
    # magnitude type is a free string, e.g. Mw, mb, Ms, etc.
    mag_type:
```

(continues on next page)

(continued from previous page)

```
  # Optional scalar moment.
  # If a moment tensor is given, the scalar moment will be (re)computed
  # from it
  scalar_moment:
    value:
    # units can be one of the following: N-m, dyne-cm
    units:
  # Optional focal mechanism, in terms of strike, dip and rake (in degrees)
  # of one of the two focal planes.
  # If a moment tensor is given, the focal mechanism will be (re)computed
  # from it
  focal_mechanism:
    # currently, only decimal degrees are supported
    units: deg
    strike:
    dip:
    rake:
  # Optional moment tensor, in up-south-east convention (USE)
  moment_tensor:
    # units can be one of the following: N-m, dyne-cm
    units:
    # moment tensor components, in moment units defined above
    m_rr:
    m_tt:
    m_pp:
    m_rt:
    m_rp:
    m_tp:

# You can specify as many events as you want, as long as they have unique
# event_id's
# - event_id: <EVENT_ID2>
#   hypocenter:
#     longitude:
#       value: <LONGITUDE2>
#       units: deg
#     latitude:
#       value: <LATITUDE2>
#       units: deg
#     depth:
#       value: <DEPTH2>
#       units: km
#     origin_time: <ORIGIN_TIME2>
# etc...
```

CHAPTER

NINE

OUTPUT FILE FORMATS

The SourceSpec main code, `source_spec` will produce the following output files (`EVID` is replaced by the actual event ID):

- `EVID.ssp.yaml`: YAML file containing the estimated spectral parameters (summary values and per station values)
- `EVID.ssp.out` (*deprecated*): text file containing the estimated spectral parameters (summary values and per station values)
- `EVID.ssp.log`: log file in text format (including the command line arguments, for reproducibility)
- `EVID.ssp.conf`: the input config file (for reproducibility)
- `EVID.residuals.hdf5`: station residuals in *HDF5 File Format*
- `EVID.spectra.hdf5`: (optional) spectra in *HDF5 File Format*
- `EVID.ssp.h`: hypocenter file in HYPO71 format with the estimated moment magnitude (only if an input HYPO71 file is provided)
- `EVID.xml`: updated QuakeML file with the results of the SourceSpec inversion (only if an input QuakeML file is provided)

The following plots will be created, in png, pdf or svg format:

- `EVID.traces.png[.pdf,.svg]`: trace plots
- `EVID.ssp.png[.pdf,.svg]`: spectral plots
- `EVID.sspweight.png[.pdf,.svg]`: spectral weight plots
- `EVID.boxplot.png[.pdf,.svg]`: box plots for the earthquake source parameters retrieved at each station
- Misfit plots, when using "grid search" or "importance sampling" for the spectral inversion

As an option, station maps can be created (requires Cartopy):

- `EVID.map_mag.png[.pdf,.svg]`: station map with symbols colored by estimated moment magnitude
- `EVID.map_fc.png[.pdf,.svg]`: station map with symbols colored by estimated corner frequency

As an option, the retrieved source parameters (per station and average) can be appended to a SQLite database, whose path is defined in the configuration file.

Finally, always as an option, `source_spec` can generate a report in HTML format.

CHAPTER

TEN

SPECTRAL FILE FORMATS

Spectra produced by SourceSpec can be optionally saved in a HDF5 file named `EVID.spectra.hdf5` (where `EVID` is the event ID; see the `save_spectra` option in *Configuration File*).

Additionally, spectral residuals are always saved to a file named `EVID.residuals.hdf5`, and average residuals produced by `source_residuals` are saved to a file named `residual_mean.hdf5`.

All these files use an HDF5 based format, detailed in the following section. They can be read using the function *spectrum.read_spectra()*, which returns a *spectrum.SpectrumStream* object, consisting of *spectrum.Spectrum* objects.

Example code:

```
>>> from sourcespec.spectrum import read_spectra
>>> specst = read_spectra('38471103.spectra.hdf5')
>>> print(specst)
SpectrumStream with 439 Spectrum objects:
CI.CCA..HHE | 200 samples, 0.2-39.9 Hz | 0.2 Hz sample interval | 59 samples logspaced,␣
↪0.20-41.70 Hz | 0.04 log10([Hz]) sample interval logspaced
CI.CCA..HHH | 200 samples, 0.2-39.9 Hz | 0.2 Hz sample interval | 59 samples logspaced,␣
↪0.20-41.70 Hz | 0.04 log10([Hz]) sample interval logspaced
CI.CCA..HHN | 200 samples, 0.2-39.9 Hz | 0.2 Hz sample interval | 59 samples logspaced,␣
↪0.20-41.70 Hz | 0.04 log10([Hz]) sample interval logspaced
CI.CCA..HHS | 200 samples, 0.2-39.9 Hz | 0.2 Hz sample interval | 59 samples logspaced,␣
↪0.20-41.70 Hz | 0.04 log10([Hz]) sample interval logspaced
CI.CCA..HHZ | 200 samples, 0.2-39.9 Hz | 0.2 Hz sample interval | 59 samples logspaced,␣
↪0.20-41.70 Hz | 0.04 log10([Hz]) sample interval logspaced
CI.CCA..HHh | 200 samples, 0.2-39.9 Hz | 0.2 Hz sample interval | 59 samples logspaced,␣
↪0.20-41.70 Hz | 0.04 log10([Hz]) sample interval logspaced
CI.CCA..HHs | 200 samples, 0.2-39.9 Hz | 0.2 Hz sample interval | 59 samples logspaced,␣
↪0.20-41.70 Hz | 0.04 log10([Hz]) sample interval logspaced
...
>>> print(specst[0])
Spectrum CI.CCA..HHE | 200 samples, 0.2-39.9 Hz | 0.2 Hz sample interval | 59 samples␣
↪logspaced, 0.20-41.70 Hz | 0.04 log10([Hz]) sample interval logspaced
>>> print(specst[0].stats)
{'delta': 0.1996007984031936,
 'npts': 200,
 'delta_logspaced': 0.040000000000000036,
 'npts_logspaced': 59,
 'station': 'CCA',
 'network': 'CI',
 'location': '',
```

(continues on next page)

(continued from previous page)

```
 'channel': 'HHE',
 'azimuth': 198.9047069007039,
 'coeff': 1282817000215832.2,
 'coords': {'elevation': 0.71,
 'latitude': 35.15251922607422,
 'longitude': -118.01648712158203},
 ...
```

Additionally, *spectrum.SpectrumStream* and *spectrum.Spectrum* objects can be saved to a text file format, with one `Spectrum` per file. The details of this format are explained below.

Here is an example of how to read an HDF5 file and save it to a text file:

```
>>> from sourcespec.spectrum import read_spectra
>>> specst = read_spectra('38471103.spectra.hdf5')
>>> specst.write('38471103.spectra.txt', format='TEXT')
>>> from os import listdir
>>> print('\n'.join(sorted(listdir('.'))))
...
38471103.spectra_0000.txt
38471103.spectra_0001.txt
38471103.spectra_0002.txt
38471103.spectra_0003.txt
38471103.spectra_0004.txt
38471103.spectra_0005.txt
38471103.spectra_0006.txt
38471103.spectra_0007.txt
38471103.spectra_0008.txt
38471103.spectra_0009.txt
38471103.spectra_0010.txt
...
>>> specst0 = read_spectra('38471103.spectra_0000.txt', format='TEXT')
>>> print(specst0)
SpectrumStream with 1 Spectrum objects:
CI.CCA..HHE | 200 samples, 0.2-39.9 Hz | 0.2 Hz sample interval | 59 samples logspaced,␣
↪0.20-41.70 Hz | 0.04 log10([Hz]) sample interval logspaced
```

10.1 HDF5 File Format

In the HDF5 file format, all the spectra are stored in a group named `spectra`. This will allow for storing additional data types in the future. Within the `spectra` group, each *spectrum.Spectrum* object is stored in a group named `spectrum_NNNNN_NET.STA.LOC.CHAN`, where `NNNN` is the index of the spectrum in the original *spectrum.SpectrumStream* object. For each group, metadata is stored in the attributes section, and data is stored into 6 datasets, as illustrated below:

```
HDF5 ──> attributes
  └── spectra
      ├── spectrum_NNNNN_NET.STA.LOC.CHAN ──> attributes
      |   ├── data
      |   ├── data_logspaced
      |   ├── data_mag
      |   ├── data_mag_logspaced
```

(continues on next page)

(continued from previous page)

```
|   ├── freq
|   └── freq_logspaced
├── spectrum_NNNNN_NET.STA.LOC.CHAN --> attributes
...
```

The mandatory metadata fields are:

- `network`: the network code;
- `station`: the station code;
- `location`: the location code;
- `channel`: the channel code;
- `delta`: the sample interval for linearly spaced frequencies;
- `npts`: the number of samples for linearly spaced frequencies and data;
- `delta_logspaced`: the sample interval for logspaced frequencies (set to 1 if logspaced frequencies are not used);
- `npts_logspaced`: the number of samples for logspaced frequencies and data (set to 0 if logspaced frequencies are not used).

Other metadata fields might be present. Dictionary-like metadata fields are stored as YAML strings.

The 6 datasets are:

- `data` (mandatory): spectral amplitude;
- `data_logspaced` (optional): spectral amplitude for logspaced frequencies;
- `data_mag` (optional): spectral amplitude in magnitude units;
- `data_mag_logspaced` (optional): spectral amplitude in magnitude units for logspaced frequencies;
- `freq` (mandatory): linearly spaced frequencies;
- `freq_logspaced` (optional): logspaced frequencies.

Example code for reading a SourceSpec HDF5 file using `h5py`:

```
>>> import h5py
>>> fp = h5py.File('38471103.spectra.hdf5', 'r')
>>> spectra = fp['spectra']
>>> print('\n'.join(spectra.keys()))
spectrum_00000_CI.CCA..HHE
spectrum_00001_CI.CCA..HHH
spectrum_00002_CI.CCA..HHN
spectrum_00003_CI.CCA..HHS
...
>>> spec = spectra['spectrum_00000_CI.CCA..HHE']
>>> print(spec.attrs['network'])
CI
>>> print(spec.attrs['station'])
CCA
>>> print(spec.attrs['channel'])
HHE
>>> print(spec.attrs['coords'])
{'elevation': 0.959, 'latitude': 35.34149169921875, 'longitude': -116.87464141845703}
```

(continues on next page)

(continued from previous page)

```
>>> print(spec['freq'][...])
[ 0.1996008   0.3992016   0.5988024   0.79840319  0.99800399  1.19760479
  1.39720559  1.59680639  1.79640719  1.99600798  2.19560878  2.39520958
...
>>> print(spec['data'][...])
[2.49035173e+15 1.37636948e+15 1.44746675e+15 1.63566457e+15
 6.93049162e+14 1.01315194e+15 9.36761128e+14 8.00776096e+14
...
```

10.2 TEXT File Format

The TEXT file format is not used internally by SourceSpec, but it can be useful to convert the HDF5 files to a more human-readable format (see above for an example on reading an HDF5 file and converting it to TEXT format).

The format is structured as follows:

- A header section in YAML format. The header is identified by the two lines `# %BEGIN STATS YAML` and `# %END STATS YAML`. Each line in the header starts with a `#` character, which should be removed when using a YAML parser.
- One or two data sections, each with three columns: `frequency (Hz)`, `data`, `data_mag` (if `data_mag` is not present, it is set to `nan`):
  - linearly spaced data, between `# %BEGIN LINSPACED DATA` and `# %END LINSPACED DATA`;
  - (optional) logspaced data, between `# %BEGIN LOGSPACED DATA` and `# %END LOGSPACED DATA`.

Here's an example TEXT file (with ellipses for brevity):

```
# %SOURCESPEC TEXT SPECTRUM FORMAT 1.0
# %BEGIN STATS YAML
# delta: 0.1996007984031936
# npts: 200
# delta_logspaced: 0.040000000000000036
# npts_logspaced: 59
# station: CCA
# network: CI
# location: ''
# channel: HHE
# azimuth: 198.9047069007039
# coeff: 1282817000215832.2
# coords:
#   elevation: 0.71
#   latitude: 35.15251922607422
#   longitude: -118.01648712158203
...
# %END STATS YAML
# %BEGIN LINSPACED DATA
# frequency(Hz) data data_mag
0.199601 60680538429002.429688 3.122033
0.399202 111489590392894.000000 3.298156
0.598802 109341550136192.765625 3.292523
0.798403 46532921128263.000000 3.045174
0.998004 69772021841544.039062 3.162454
```

(continues on next page)

(continued from previous page)

```
...
# %END LINSPACED DATA
# %BEGIN LOGSPACED DATA
# frequency_logspaced(Hz) data_logspaced data_mag_logspaced
0.199601 60680538429002.437500 3.122033
0.218858 65978522113754.125000 3.146268
0.239973 71686845260565.812500 3.170293
0.263125 77968569379397.437500 3.194613
0.288511 84857989097347.140625 3.219128
...
# %END LOGSPACED DATA
```

CHAPTER

ELEVEN

INSTALLATION

SourceSpec requires at least Python 3.9. All the required dependencies will be downloaded and installed during the setup process.

11.1 Installing the latest release

11.1.1 Using Anaconda

The following command will automatically create an Anaconda environment named `sourcespec`, install the required packages and install the latest version of SourceSpec via `pip`:

```
conda env create --file https://raw.githubusercontent.com/SeismicSource/sourcespec/main/
↪sourcespec_conda_env.yml
```

If you want a different name for your environment, use:

```
conda env create -n YOUR_ENV_NAME --file https://raw.githubusercontent.com/SeismicSource/
↪sourcespec/main/sourcespec_conda_env.yml
```

Activate the environment with:

```
conda activate sourcespec
```

(or `conda activate YOUR_ENV_NAME`)

To keep SourceSpec updated run:

```
pip install --upgrade sourcespec
```

from within your environment.

11.1.2 Using pip and PyPI

The latest release of SourceSpec is available on the Python Package Index.

You can install it easily through `pip`:

```
pip install sourcespec
```

To upgrade from a previously installed version:

```
pip install --upgrade sourcespec
```

11.1.3 From SourceSpec GitHub releases

Download the latest release from the releases page, in `zip` or `tar.gz` format, then:

```
pip install sourcespec-X.Y.zip
```

or

```
pip install sourcespec-X.Y.tar.gz
```

Where, `X.Y` is the version number (e.g., `1.2`). You don't need to uncompress the release files yourself.

11.2 Installing a developer snapshot

If you need a recent feature that is not in the latest release (see the "unreleased" section in *SourceSpec Changelog*), you want to use the more recent development snapshot from the SourceSpec GitHub repository.

11.2.1 Using pip

The easiest way to install the most recent development snapshot is to download and install it through `pip`, using its builtin `git` client:

```
pip install git+https://github.com/SeismicSource/sourcespec.git
```

Run this command again, from times to times, to keep SourceSpec updated with the development version.

11.2.2 Cloning the SourceSpec GitHub repository

If you want to take a look at the source code (and possibly modify it), clone the project using `git`:

```
git clone https://github.com/SeismicSource/sourcespec.git
```

or, using SSH:

```
git clone git@github.com:SeismicSource/sourcespec.git
```

(avoid using the "Download ZIP" option from the green "Code" button, since version number is lost).

Then, go into the `sourcespec` main directory and install the code in "editable mode" by running:

```
pip install -e .
```

You can keep your local SourceSpec repository updated by running `git pull` from times to times. Thanks to `pip`'s "editable mode", you don't need to reinstall SourceSpec after each update.

CHAPTER

TWELVE

SAMPLE RUNS

Several sample runs are available in the sourcespec_testruns repository.

CHAPTER

THIRTEEN

GETTING HELP

13.1 I need help

Join the SourceSpec Discussions and feel free to ask!

13.2 I found a bug

Please open an Issue.

CHAPTER

FOURTEEN

CONTRIBUTING

SourceSpec development happens on GitHub.

I'm very open to contributions: if you have new ideas, please open an Issue. Don't hesitate sending me pull requests with new features and/or bugfixes!

CHAPTER

FIFTEEN

HOW TO CITE

If you used SourceSpec for a scientific paper, please cite it as:

> Satriano, C. (2026). SourceSpec – Earthquake source parameters from P- or S-wave displacement spectra (X.Y). doi: 10.5281/ZENODO.3688587

Please replace `X.Y` with the SourceSpec version number you used.

You can also cite the following abstract presented at the 2016 AGU Fall Meeting:

> Satriano, C., Mejia Uquiche, A. R., & Saurel, J. M. (2016). Spectral estimation of seismic moment, corner frequency and radiated energy for earthquakes in the Lesser Antilles. In AGU Fall Meeting Abstracts (Vol. 2016, pp. S13A-2518), bibcode: 2016AGUFM.S13A2518S

CHAPTER

SIXTEEN

SOURCESPEC API

SourceSpec has a modular structure. Each module corresponds to a specific function or class of functions.

16.1 Main modules

SourceSpec main modules are presented below, following the logical order on which they're used within `source_spec.py`.

16.1.1 source_spec.py

Earthquake source parameters from inversion of P- or S-wave spectra.

copyright
2012 Claudio Satriano <satriano@ipgp.fr>

2013-2014 Claudio Satriano <satriano@ipgp.fr>,
Emanuela Matrullo <matrullo@geologie.ens.fr>, Agnes Chounet <chounet@ipgp.fr>

2015-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

`source_spec.`**`main`**`()`
Main routine for source_spec.

16.1.2 ssp_parse_arguments

Argument parser for sourcespec.

copyright
2021-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

`ssp_parse_arguments.`**`parse_args`**`(`*progname*`)`
Parse command line arguments.

16.1.3 ssp_setup

Setup functions for sourcespec.

copyright
2012 Claudio Satriano <satriano@ipgp.fr>

2013-2014 Claudio Satriano <satriano@ipgp.fr>,
Emanuela Matrullo <matrullo@geologie.ens.fr>, Agnes Chounet <chounet@ipgp.fr>

2015-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

ssp_setup.**configure**(*options*, *progname*, *config_overrides=None*)

Parse command line arguments and read config file.

Parameters

- **options** (*object*) – An object containing command line options
- **progname** (*str*) – The name of the program
- **config_overrides** (*dict*) – A dictionary with parameters that override or extend those defined in the config file

Returns
A `Config` object with both command line and config options.

ssp_setup.**move_outdir**(*config*)

Move outdir to a new dir named from evid (and optional run_id).

ssp_setup.**remove_old_outdir**(*config*)

Try to remove the old outdir.

ssp_setup.**save_config**(*config*)

Save config file to output dir.

ssp_setup.**setup_logging**(*config*, *basename=None*, *progname='source_spec'*)

Set up the logging infrastructure.

This function is typically called twice: the first time without basename and a second time with a basename (typically the eventid). When called the second time, the previous logfile is renamed using the given basename.

ssp_setup.**sigint_handler**(*sig*, *frame*)

Handle SIGINT signal.

ssp_setup.**ssp_exit**(*retval=0*, *abort=False*)

Exit the program.

16.1.4 ssp_read_traces

Read traces in multiple formats of data and metadata.

copyright
2012 Claudio Satriano <satriano@ipgp.fr>

2013-2014 Claudio Satriano <satriano@ipgp.fr>,
Emanuela Matrullo <matrullo@geologie.ens.fr>

2015-2026 Claudio Satriano <satriano@ipgp.fr>,
Sophie Lambotte <sophie.lambotte@unistra.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

ssp_read_traces.**read_traces**(*config*)

Read traces, store waveforms and metadata.

16.1.5 ssp_process_traces

Trace processing for sourcespec.

copyright
2012 Claudio Satriano <satriano@ipgp.fr>

2013-2014 Claudio Satriano <satriano@ipgp.fr>,
Emanuela Matrullo <matrullo@geologie.ens.fr>

2015-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

ssp_process_traces.**filter_trace**(*config*, *trace*)

Filter trace.

ssp_process_traces.**process_traces**(*config*, *st*)

Remove mean, deconvolve and ignore unwanted components.

16.1.6 ssp_build_spectra

Build spectral objects.

copyright
2012 Claudio Satriano <satriano@ipgp.fr>

2013-2014 Claudio Satriano <satriano@ipgp.fr>,
Emanuela Matrullo <matrullo@geologie.ens.fr>, Agnes Chounet <chounet@ipgp.fr>

2015-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

exception ssp_build_spectra.**SpectrumIgnored**(*message*, *reason*)

Spectrum ignored exception

exception ssp_build_spectra.**TraceIgnored**(*message*, *reason*)

Trace ignored exception

ssp_build_spectra.**build_spectra**(*config*, *st*)

Build spectra and the `spec_st` object.

Computes P- or S-wave (displacement) spectra from accelerometers and velocimeters, uncorrected for anelastic attenuation, corrected for instrumental constants, normalized by geometrical spreading.

16.1.7 ssp_plot_traces

Trace plotting routine.

copyright
2015-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

class ssp_plot_traces.**ScalarFormatter**(**args: Any*, ***kwargs: Any*)

A ScalarFormatter with a custom format.

ssp_plot_traces.**plot_traces**(*config*, *st*, *ncols=None*, *block=True*, *suffix=None*)

Plot raw (counts) or processed traces (instrument units and filtered).

Display to screen and/or save to file.

16.1.8 ssp_inversion

Spectral inversion routines for sourcespec.

copyright

2012 Claudio Satriano <satriano@ipgp.fr>

2013-2014 Claudio Satriano <satriano@ipgp.fr>,
Emanuela Matrullo <matrullo@geologie.ens.fr>, Agnes Chounet <chounet@ipgp.fr>

2015-2026 Claudio Satriano <satriano@ipgp.fr>

license

CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

ssp_inversion.**spectral_inversion**(*config*, *spec_st*, *weight_st*)

Inversion of displacement spectra.

16.1.9 ssp_radiated_energy

Compute radiated energy from spectral integration.

copyright

2012 Claudio Satriano <satriano@ipgp.fr>

2013-2014 Claudio Satriano <satriano@ipgp.fr>,
Emanuela Matrullo <matrullo@geologie.ens.fr>, Agnes Chounet <chounet@ipgp.fr>

2015-2026 Claudio Satriano <satriano@ipgp.fr>

license

CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

ssp_radiated_energy.**radiated_energy_and_apparent_stress**(*config*, *spec_st*, *specnoise_st*, *sspec_output*)

Compute radiated energy (in N.m) and apparent stress (in MPa).

Parameters

- **config** – Config object
- **type** (*sspec_output*) – sourcespec.config.Config
- **spec_st** – Stream of spectra
- **type** – obspy.core.stream.Stream
- **specnoise_st** – Stream of noise spectra
- **type** – obspy.core.stream.Stream
- **sspec_output** – Output of spectral inversion
- **type** – sourcespec.ssp_data_types.SourceSpecOutput

16.1.10 ssp_local_magnitude

Local magnitude calculation for sourcespec.

copyright
2012 Claudio Satriano <satriano@ipgp.fr>

2013-2014 Claudio Satriano <satriano@ipgp.fr>,
Emanuela Matrullo <matrullo@geologie.ens.fr>

2015-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

`ssp_local_magnitude.`**`local_magnitude`**`(`*config*, *st*, *proc_st*, *sspec_output*`)`
Compute local magnitude from max absolute W-A amplitude.

16.1.11 ssp_summary_statistics

Post processing of station source parameters.

copyright
2012-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

`ssp_summary_statistics.`**`compute_summary_statistics`**`(`*config*, *sspec_output*, *spec_st*, *weight_st*`)`
Compute summary statistics from station spectral parameters.

16.1.12 ssp_output

Output functions for source_spec.

copyright
2012 Claudio Satriano <satriano@ipgp.fr>

2013-2014 Claudio Satriano <satriano@ipgp.fr>,
Emanuela Matrullo <matrullo@geologie.ens.fr>

2015-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

`ssp_output.`**`save_spectra`**`(`*config*, *spec_st*`)`
Save spectra to file.

`ssp_output.`**`write_output`**`(`*config*, *sspec_output*`)`
Write results into different formats.

16.1.13 ssp_residuals

Spectral residual routine for sourcespec.

copyright

2013-2014 Claudio Satriano <satriano@ipgp.fr>,
Agnes Chounet <chounet@ipgp.fr>

2015-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

ssp_residuals.**spectral_residuals**(*config*, *spec_st*, *sspec_output*)

Compute spectral residuals with respect to an average spectral model. Saves a stream of residuals to disk in HDF5 format.

Parameters

- **config** (Config) – Configuration object
- **spec_st** (SpectrumStream) – Stream of spectra
- **sspec_output** (SourceSpecOutput) – Output of the source spectral parameter estimation

16.1.14 ssp_plot_spectra

Spectral plotting routine.

copyright
2012 Claudio Satriano <satriano@ipgp.fr>

2013-2014 Claudio Satriano <satriano@ipgp.fr>,
Emanuela Matrullo <matrullo@geologie.ens.fr>

2015-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

class ssp_plot_spectra.**PlotParams**

Parameters for plotting spectra.

set_plot_params(*config*, *spec_st*, *specnoise_st*)

Determine the number of plots and axes min and max.

ssp_plot_spectra.**plot_spectra**(*config*, *spec_st*, *specnoise_st=None*, *ncols=None*, *stack_plots=False*, *plot_type='regular'*)

Plot spectra for signal and noise.

Display to screen and/or save to file.

16.1.15 ssp_plot_stacked_spectra

Plot stacked spectra, along with summary inverted spectrum.

copyright
2023-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

ssp_plot_stacked_spectra.**plot_stacked_spectra**(*config*, *spec_st*, *weight_st*, *sspec_output*)

Plot stacked spectra, along with summary inverted spectrum.

16.1.16 ssp_plot_params_stats

Plot parameter statistics.

copyright
2022-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

class ssp_plot_params_stats.**PlotParam**(*name*, *unit*, *color*)

A plot parameter.

ssp_plot_params_stats.**box_plots**(*config*, *sspec_output*)

Show parameter statistics through box plots.

16.1.17 ssp_plot_stations

Station plotting routine.

copyright
2018-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

ssp_plot_stations.**plot_stations**(*config*, *sspec_output*)

Plot station map, color coded by magnitude or fc.

Parameters

- **config** (config.Config) – Configuration object
- **sspec_output** (ssp_data_types.SourceSpecOutput) – SourceSpecOutput object

16.1.18 ssp_html_report

Generate an HTML report for source_spec.

copyright
2021-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

class ssp_html_report.**HTMLtemplates**

Class to hold paths to HTML templates.

ssp_html_report.**html_report**(*config*, *sspec_output*)

Generate an HTML report.

16.2 Other modules

These modules, in alphabetical order, are used by the main modules.

16.2.1 ssp_correction

Spectral station correction calculated from ssp_residuals.

copyright

2013-2014 Claudio Satriano <satriano@ipgp.fr>,
Agnes Chounet <chounet@ipgp.fr>

2015-2026 Claudio Satriano <satriano@ipgp.fr>
Kris Vanneste <kris.vanneste@oma.be>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

ssp_correction.**station_correction**(*spec_st*, *config*)

Correct spectra using station-average residuals.

Residuals are obtained from a previous run.

Parameters

- **spec_st** (SpectrumStream or list of Spectrum) – List of spectra to be corrected. Corrected spectra are appended to the list (component code of uncorrected spectra is renamed to 'h').
- **config** (Config) – Configuration object containing the residuals file path.

16.2.2 ssp_data_types

Classes for spectral inversion routines.

copyright
2017-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

class ssp_data_types.**Bounds**(*config*, *spec*, *initial_values*)

Bounds for bounded spectral inversion.

property bounds

Get bounds for minimize() as sequence of (min, max) pairs.

get_bounds_curve_fit()

Get bounds for curve-fit().

class ssp_data_types.**InitialValues**(*Mw_0=None*, *fc_0=None*, *t_star_0=None*, *invert_t_star=True*)

Initial values for spectral inversion.

get_params0()

Get initial values as a tuple.

class ssp_data_types.**OrderedAttribDict**

An ordered dictionary whose values can be accessed as classattributes.

class ssp_data_types.**SourceSpecOutput**

The output of SourceSpec.

error_array(*key*, *filter_outliers=False*)

Return an array of errors (two columns) for the given key.

find_outliers(*key*, *n*)

Find outliers using the IQR method.

```
    Q1-n*IQR   Q1   median  Q3    Q3+n*IQR
                  |-----:-----|
o      |--------|     :     |--------|    o  o
                  |-----:-----|
outlier         <----------->            outliers
                      IQR
```

If `n` is `None`, then the above check is skipped. `Nan` and `inf` values are also marked as outliers.

mean_nobs()

Return a dictionary of number of observations used for computing mean.

mean_uncertainties()

Return a dictionary of mean uncertainties.

mean_values()

Return a dictionary of mean values.

outlier_array(*key*)

Return an array of outliers for the given key.

percentiles_nobs()

Return a dictionary of number of observations used for computing percentiles.

percentiles_uncertainties()

Return a dictionary of percentile uncertainties.

percentiles_values()

Return a dictionary of percentile values.

reference_summary_parameters()

Return a dictionary of reference summary parameters, each being a SummaryStatistics() object.

reference_uncertainties()

Return a dictionary of reference uncertainties.

reference_values()

Return a dictionary of reference values.

value_array(*key*, *filter_outliers=False*)

Return an array of values for the given key.

weighted_mean_nobs()

Return a dictionary of number of observations used for computing weighted mean.

weighted_mean_uncertainties()

Return a dictionary of weighted mean uncertainties.

weighted_mean_values()

Return a dictionary of weighted mean values.

class ssp_data_types.**SpectralParameter**(*param_id*, *name=None*, *units=None*, *value=None*, *uncertainty=None*, *lower_uncertainty=None*, *upper_uncertainty=None*, *confidence_level=None*, *format_spec=None*)

A spectral parameter measured at one station.

compact_uncertainty()

Return uncertainty in a compact form.

class ssp_data_types.**StationParameters**(*station_id*, *instrument_type=None*, *latitude=None*, *longitude=None*, *hypo_dist_in_km=None*, *epi_dist_in_km=None*, *azimuth=None*, *spectral_snratio_mean=None*, *spectral_snratio_max=None*, *rmsn=None*, *quality_of_fit=None*, *ignored=False*, *ignored_reason=None*)

The parameters describing a given station (e.g., its id and location) and the spectral parameters measured at that station.

Spectral parameters are provided as attributes, using SpectralParameter() objects.

get_spectral_parameters()

Return a dictionary of spectral parameters.

class ssp_data_types.**SummarySpectralParameter**(*param_id*, *name=None*, *units=None*, *format_spec=None*)

A summary spectral parameter comprising one ore more summary statistics.

class ssp_data_types.**SummaryStatistics**(*stat_type*, *value=None*, *uncertainty=None*, *lower_uncertainty=None*, *upper_uncertainty=None*, *confidence_level=None*, *lower_percentage=None*, *mid_percentage=None*, *upper_percentage=None*, *nobs=None*, *message=None*, *format_spec=None*)

A summary statistics (e.g., mean, weighted_mean, percentile), along with its uncertainty.

compact_uncertainty()

Return uncertainty in a compact form.

16.2.3 ssp_event

SourceSpec event class and supporting classes.

copyright
2023-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

class ssp_event.**SSPCoordinate**(*value=None*, *units=None*, ***kwargs*)

SourceSpec coordinate class.

Stores a longitude or latitude value. Currently only degrees are supported.

class ssp_event.**SSPDepth**(*value=None*, *units=None*, ***kwargs*)

SourceSpec depth class.

property value_in_km

Return the depth value in km.

property value_in_m

Return the depth value in m.

class ssp_event.**SSPEvent**(*event_dict=None*)

SourceSpec event class.

from_event_dict(*event_dict*)

Initialize the event object from a event dictionary.

class ssp_event.**SSPFocalMechanism**(*strike=None*, *dip=None*, *rake=None*, *units=None*, ***kwargs*)

SourceSpec focal mechanism class.

Angles are in degrees.

from_moment_tensor(*moment_tensor*)

Initialize the focal mechanism object from a moment tensor.

Parameters
moment_tensor (*SSPMomentTensor*) – moment tensor object

class ssp_event.**SSPHypocenter**(*longitude=None*, *latitude=None*, *depth=None*, *origin_time=None*, ***kwargs*)

SourceSpec hypocenter class.

class ssp_event.**SSPMagnitude**(*value=None*, *mag_type=None*, ***kwargs*)

SourceSpec magnitude class.

from_scalar_moment(*scalar_moment*)

Initialize the magnitude object from a scalar moment.

Parameters
scalar_moment (*SSPScalarMoment*) – scalar moment object

class ssp_event.**SSPMomentTensor**(*units=None*, *m_rr=None*, *m_tt=None*, *m_pp=None*, *m_rt=None*, *m_rp=None*, *m_tp=None*, ***kwargs*)

SourceSpec moment tensor class.

to_N_m()

Convert the moment tensor to N-m, if necessary.

class ssp_event.**SSPScalarMoment**(*value=None*, *units=None*, ***kwargs*)

SourceSpec scalar moment class.

from_moment_tensor(*moment_tensor*)

Initialize the scalar moment object from a moment tensor.

Parameters
moment_tensor (*SSPMomentTensor*) – moment tensor object

to_N_m()

Convert the scalar moment to N-m, if necessary.

16.2.4 ssp_geom_spreading

Functions to compute geometrical spreading coefficients.

copyright
2012-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

ssp_geom_spreading.**geom_spread_boatwright**(*hypo_dist_in_km*, *cutoff_dist_in_km*, *freqs*)

" Geometrical spreading coefficient from Boatwright et al. (2002), eq. 8.

Except that we take the square root of eq. 8, since we correct amplitude and not energy.

Parameters

- **hypo_dist_in_km** (*float*) – Hypocentral distance (km).
- **cutoff_dist_in_km** (*float*) – Cutoff distance (km).
- **freqs** (*numpy.ndarray*) – Frequencies (Hz).

Returns
Geometrical spreading correction (in m)

Return type
numpy.ndarray

ssp_geom_spreading.**geom_spread_r_power_n**(*hypo_dist_in_km*, *exponent*)

r geometrical spreading coefficient.

Parameters

- **hypo_dist_in_km** (*float*) – Hypocentral distance (km).
- **exponent** (*float*) – Exponent.

Returns
Geometrical spreading correction (in m)

Return type
float

ssp_geom_spreading.**geom_spread_r_power_n_segmented**(*hypo_dist_in_km*, *exponents*, *hinge_distances*)

Geometrical spreading function defined as piecewise continuous powerlaw, as defined in Boore (2003), eq. 9

Parameters

- **hypo_dist_in_km** (*float*) – Hypocentral distance (km).
- **exponents** (*numpy.ndarray*) – Exponents for different powerlaw segments
- **hinge_distances** (*numpy.ndarray*) – Distances defining start of powerlaw segments

Returns
Geometrical spreading correction (for distance in m)

Return type
float

ssp_geom_spreading.**geom_spread_teleseismic**(*angular_distance*, *source_depth_in_km*, *station_depth_in_km*, *phase*)

Calculate geometrical spreading coefficient for teleseismic body waves.

Implements eq (4) in Okal (1992) for a spherically symmetric Earth. This equations is derived from the conservation of the kinetic energy flux along a ray tube between the source and the receiver.

Parameters

- **angular_distance** (*float*) – Angular distance (degrees).
- **source_depth_in_km** (*float*) – Source depth (km).
- **station_depth_in_km** (*float*) – Station depth (km).
- **phase** (*str*) – Phase type (`'P'` or `'S'`).

Returns
Geometrical spreading correction (in m)

Return type
float

ssp_geom_spreading.**plot_geom_spread_models**(*source_depth*, *epi_dists*, *models=None*, *plottype='linlog'*, *xaxis='epi_dist'*, *return_fig=False*)

Plot geometrical spreading coefficients for different models.

Parameters

- **source_depth** (*float or list[float] or numpy.ndarray*) – Source depth(s) (km). Can be a scalar, list, or numpy array.
- **epi_dists** (*float or list[float] or numpy.ndarray*) – Epicentral distances (km). Can be a scalar, list, or numpy array.
- **models** (*list[tuple[str, dict or None]]*) – List of tuples where each tuple contains:
  - The model name.
  - A dictionary of model parameters (or `None` if the model has no parameters or to use defaults).

  Valid model names are:
  - `'r_power_n'`: r geometrical spreading.
  - `'r_power_n_segmented'`: Piecewise continuous power law, as defined in Boore (2003), eq. 9.
  - `'boatwright'`: Boatwright et al. (2002).
  - `'teleseismic'`: Teleseismic (Okal, 1992).

  If `None`, defaults to `[('r_power_n', {'exponent': 1})]`.
- **plottype** (*str*) – Type of plot scaling. Options:
  - `'linlin'`
  - `'linlog'`
  - `'loglin'`
  - `'loglog'`
- **xaxis** (*str*) – X-axis type. Options:
  - `'epi_dist'`: Epicentral distance.
  - `'hypo_dist'`: Hypocentral distance.
- **return_fig** (*bool*) – If True, return the figure instead of showing it.

Returns
Matplotlib figure, if `return_fig` is True.

Return type
matplotlib.figure.Figure

Examples

```
>>> import numpy as np
>>> source_depth = [10, 20]
>>> epi_dists = np.arange(10, 1000, 10)
>>> models = [
>>>     ('r_power_n', {'exponent': 1}),
>>>     ('boatwright', {'cutoff_dist_in_km': 100}),
>>>     ('boatwright', {'cutoff_dist_in_km': 200, 'freqs': 0}),
>>>     ('boatwright', {'cutoff_dist_in_km': 200, 'freqs': 10}),
>>>     ('teleseismic', {'phase': 'P'}),
>>>     ('teleseismic', {'phase': 'S'}),
>>> ]
>>> plot_geom_spread_models(
```

(continues on next page)

(continued from previous page)

```
>>>     source_depth, epi_dists, models=models,
>>>     plottype='linlog', xaxis='epi_dist')
```

```
>>> import numpy as np
>>> source_depth = 0
>>> epi_dists = np.arange(1, 1000, 1)
>>> models = [
>>>     ('r_power_n', {'exponent': 1}),
>>>     ('r_power_n_segmented', None),
>>> ]
>>> plot_geom_spread_models(
>>>     source_depth, epi_dists, models=models,
>>>     plottype='loglog', xaxis='hypo_dist')
```

16.2.5 ssp_grid_sampling

A class for sampling a parameter space over a grid.

Sampling can be performed by several approaches. The class provides optimal solutions, uncertainties and plotting methods.

copyright
2022-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

class ssp_grid_sampling.**GridSampling**(*misfit_func*, *bounds*, *nsteps*, *sampling_mode*, *params_name*, *params_unit*)

A class for sampling a parameter space over a grid.

Sampling can be performed by several approaches. The class provides optimal solutions, uncertainties and plotting methods.

property conditional_misfit

Compute 1-D conditional misfit profiles for each parameter.

For each parameter axis, return the misfit values obtained by fixing all other parameters to their optimal values (from `self.min_idx`) and extracting the 1-D profile along that axis. Results are cached in `self._conditional_misfit` so the computation runs only once per sampled misfit.

Returns
A tuple with one 1-D array per parameter axis containing the conditional misfit profile along that axis. Returns `None` when `self.misfit` is not set.

Return type
`tuple` of `numpy.ndarray` or `None`

property conditional_peak_widths

Find width of conditional misfit around its minimum.

grid_search()

Sample the misfit function by simple grid search.

kdtree_search()

Sample the misfit function using kdtree search.

property min_idx

Find the index of the minimum of the misfit function.

property params_err

Return optimal parameters uncertainties.

property params_opt

Return optimal parameters.

plot_conditional_misfit(*config*, *label*)

Plot conditional misfit for each parameter.

plot_misfit_2d(*config*, *plot_par_idx*, *label*)

Plot a 2D conditional misfit map.

property values

Return a meshgrid of parameter values.

This property lazily builds and caches `self._values` as the tuple of N-D coordinate arrays produced by `np.meshgrid(..., indexing='ij')`. The 1-D vectors used to create the mesh are derived from `self.truebounds` and `self.nsteps`; when `sampling_mode` is `'log'` those 1-D vectors are created with `np.logspace`.

Returns

Tuple of N arrays (one per parameter). Each array has shape `(n_0, n_1, ..., n_{N-1})` and gives the coordinate value for that parameter at every grid point (the same structure returned by `np.meshgrid(*values_1d, indexing='ij')`).

Return type

`tuple` of `numpy.ndarray`

property values_1d

Return 1-D grid value arrays for each parameter axis.

Returns

One 1-D array per parameter axis, in the same order used to build `self.values` (the order passed to `np.meshgrid(..., indexing='ij')`). Each array has shape `(n_i,)` and contains the sampled parameter values along that axis. Parameters sampled with `'log'` mode produce values via `np.logspace` and are returned in linear scale.

Return type

`tuple` of `numpy.ndarray`

Note

Calling `np.meshgrid(*self.values_1d, indexing='ij')` reconstructs `self.values` (if the 1-D arrays are unchanged).

ssp_grid_sampling.**find_peak_width**(*x*, *peak_idx*, *rel_height*, *negative=False*)

Find width of a single peak at a given relative height.

rel_height: float parameter between 0 and 1

0 means the base of the curve and 1 the peak value (Note: this is the opposite of scipy.peak_widths)

16.2.6 ssp_pick

SourceSpec pick class.

copyright
2023-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

`class ssp_pick.SSPPick`

A pick object.

16.2.7 ssp_spectral_model

Spectral model and objective function.

copyright
2012 Claudio Satriano <satriano@ipgp.fr>

2013-2014 Claudio Satriano <satriano@ipgp.fr>,
Emanuela Matrullo <matrullo@geologie.ens.fr>, Agnes Chounet <chounet@ipgp.fr>

2015-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

`ssp_spectral_model.callback(_)`

Empty callback function for bounded inversion.

`ssp_spectral_model.objective_func(`*freq*, *ampl*, *weight*, *t_star=None*`)`

Objective function generator for bounded inversion.

Parameters

- **freq** (array-like) – Frequencies (in Hz)
- **ampl** (array-like) – Log spectral amplitudes
- **weight** (array-like) – Weights for each data point
- **t_star** (float or callable, *optional*) – Set a fixed t_star value or function of frequencies; if None, t_star is a parameter to invert for

Returns
Objective function

Return type
callable

`ssp_spectral_model.spectral_model(`*freq*, *Mw*, *fc*, *t_star*, *alpha=1.0*`)`

Spectral model for seismic source spectrum.

This function computes the theoretical spectral model based on the Brune source model with attenuation, expressed in terms of moment magnitude.

$$Y_{data} = M_w + \frac{2}{3}\left[-\log_{10}\left(1+\left(\frac{f}{f_c}\right)^2\right) - \pi\, f^{\alpha} t^*(f) \log_{10} e\right]$$

See *Theoretical Background*

Parameters

- **freq** (float or array-like) – Frequency or array of frequencies (in Hz) at which to compute the spectral model.
- **Mw** (float) – Moment magnitude of the seismic event.
- **fc** (float) – Corner frequency (in Hz) of the source spectrum.
- **t_star** (float or callable) – Attenuation parameter t* (in seconds), equal to travel time divided by quality factor (tt/Q). Can be a constant value or a function of frequency.
- **alpha** (float, *optional*) – Frequency exponent for the attenuation term. Default is 1.0 for standard attenuation model.

Returns
Logarithmic spectral amplitude(s) at the specified frequency/frequencies.

Return type
float or array-like

16.2.8 ssp_qml_output

QuakeML output for source_spec.

copyright
2016-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement, Version 2.1 (http://www.cecill.info/index.en.html)

class ssp_qml_output.**SSPContainerTag**(**args: Any*, ***kwargs: Any*)
Container for nested custom tags.

class ssp_qml_output.**SSPExtra**(**args: Any*, ***kwargs: Any*)
Container for custom tags.

class ssp_qml_output.**SSPTag**(**args: Any*, ***kwargs: Any*)
Custom tag object.

ssp_qml_output.**write_qml**(*config*, *sspec_output*)
Write QuakeML output.

16.2.9 ssp_radiation_pattern

Compute radiation pattern.

copyright
2021-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

ssp_radiation_pattern.**get_radiation_pattern_coefficient**(*stats*, *config*)
Get radiation pattern coefficient.

ssp_radiation_pattern.**radiation_pattern**(*strike*, *dip*, *rake*, *takeoff_angle*, *azimuth*, *wave*)
Body wave radiation pattern.

From Lay-Wallace, page 340.

ssp_radiation_pattern.**toRad**(*angle*)
Convert angle from degrees to radians.

16.2.10 ssp_read_sac_header

Read metadata from SAC file headers.

copyright
2023-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

ssp_read_sac_header.**compute_sensitivity_from_SAC**(*trace*, *config*)

Compute sensitivity from SAC header fields.

ssp_read_sac_header.**get_event_from_SAC**(*trace*)

Get event information from SAC header.

ssp_read_sac_header.**get_instrument_from_SAC**(*trace*)

Get instrument information from SAC header.

ssp_read_sac_header.**get_picks_from_SAC**(*trace*)

Get picks from SAC header.

ssp_read_sac_header.**get_station_coordinates_from_SAC**(*trace*)

Get station coordinates from SAC header.

ssp_read_sac_header.**is_SAC_trace**(*trace*)

Check if a trace is in SAC format.

16.2.11 ssp_read_station_metadata

Read station metadata in StationXML, dataless SEED, SEED RESP, PAZ (SAC polezero format).

copyright
2012-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

class ssp_read_station_metadata.**PAZ**(*file=None*)

Instrument response defined through poles and zeros.

property seedID

Return the seedID.

to_inventory()

Convert PAZ object to an Inventory object.

ssp_read_station_metadata.**read_station_metadata**(*path*)

Read station metadata into an ObsPy `Inventory` object.

Parameters
path (*str*) – path to the station metadata file or directory

Returns
inventory

Return type
Inventory

16.2.12 ssp_sqlite_output

SQLite output for source_spec.

copyright
2013-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

ssp_sqlite_output.**write_sqlite**(*config*, *sspec_output*)

Write SSP output to SQLite database.

Parameters

- **config** (config.Config) – SSP configuration object
- **sspec_output** (ssp_data_types.SourceSpecOutput) – SSP output object

16.2.13 ssp_update_db

Update an existing SourceSpec database from a previous version.

copyright
2013-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

ssp_update_db.**update_db_file**(*db_file*)

Update an existing SourceSpec database from a previous version.

Parameters
db_file (*str*) – SQLite database file

16.2.14 ssp_util

Utility functions for sourcespec.

copyright
2012-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

class ssp_util.**MediumProperties**(*lon*, *lat*, *depth_in_km*, *config*)

Class to retrieve medium properties from config.

Parameters

- **lon** (*float*) – Longitude (degrees).
- **lat** (*float*) – Latitude (degrees).
- **depth_in_km** (*float*) – Depth (km).
- **config** (Config) – Configuration object.

get(*mproperty*, *where*)

Get medium property (P- or S-wave velocity, density) at a given point from NLL grid, config or taup model.

Parameters

- **mproperty** (*str*) – Property to be retrieved ('vp', 'vs' or 'rho').

- **where** (*str*) – Where to retrieve medium property (`'source'` or `'stations'`).

Returns
Property value.

Return type
float

`get_from_config_param_source`(*mproperty*)

Get medium property at the source from config parameter.

Parameters
mproperty (*str*) – Property to be retrieved (`'vp'`, `'vs'` or `'rho'`).

Returns
Property value.

Return type
float

`get_from_config_param_station`(*mproperty*)

Get medium property at the station from config parameter.

Parameters
mproperty (*str*) – Property to be retrieved (`'vp'`, `'vs'` or `'rho'`).

Returns
Property value.

Return type
float

`get_from_taup`(*mproperty*)

Get medium property (P- or S-wave velocity, density) at a given depth from taup model.

Parameters
mproperty (*str*) – Property to be retrieved (`'vp'`, `'vs'` or `'rho'`).

Returns
Property value.

Return type
float

`get_vel_from_NLL`(*wave*)

Get velocity from NLL model.

Parameters
wave (*str*) – Wave type (`'P'` or `'S'`).

Returns
Velocity (km/s).

Return type
float

`to_string`(*mproperty*, *value*)

Return a string with the property name and value.

Parameters

- **mproperty** (*str*) – Property name. Must contain one of the following: `'vp'`, `'vs'`, `'rho'`, `'depth'`

- **value** (*float*) – Property value.

Returns
Property string.

Return type
str

ssp_util.**cosine_taper**(*signal*, *width*, *left_taper=False*)

Apply a cosine taper to the signal.

ssp_util.**mag_to_moment**(*magnitude*)

Convert magnitude to moment.

ssp_util.**moment_to_mag**(*moment*)

Convert moment to magnitude.

ssp_util.**primary_and_secondary_azimuthal_gap**(*azimuth_array*)

Compute primary and secondary azimuthal gap from an array of azimuths.

Parameters
azimuth_array (*list of float or numpy array*) – Array of azimuths (degrees).

Returns
Primary and secondary gap (degrees).

Return type
tuple of two floats

ssp_util.**quality_factor**(*travel_time_in_s*, *t_star_in_s*)

Compute quality factor from travel time and t_star.

ssp_util.**remove_instr_response**(*trace*, *pre_filt=(0.5, 0.6, 40.0, 45.0)*)

Remove instrument response from a trace.

Trace is converted to the sensor units (m for a displacement sensor, m/s for a short period or broadband velocity sensor, m/s**2 for a strong motion sensor).

Parameters

- **trace** (Trace) – Trace to be corrected.
- **pre_filt** (*tuple of four floats*) – Pre-filter frequencies (None means no pre-filtering).

ssp_util.**select_trace**(*stream*, *traceid*, *instrtype*)

Select trace from stream using traceid and instrument type.

ssp_util.**smooth**(*signal*, *window_len=11*, *window='hanning'*)

Smooth the signal using a window with requested size.

ssp_util.**source_radius**(*fc_in_hz*, *vs_in_m_per_s*, *k_coeff=0.3724*)

Compute source radius in meters.

Madariaga (2009), doi:10.1007/978-1-4419-7695-6_22, eq. 31 Kaneko and Shearer (2014), doi:10.1093/gji/ggu030, eq. 2, 15, 16

ssp_util.**spec_minmax**(*amp*, *freq*, *amp_minmax=None*, *freq_minmax=None*)

Get minimum and maximum values of spectral amplitude and frequency.

ssp_util.**static_stress_drop**(*Mo_in_N_m*, *ra_in_m*)

Compute static stress drop in MPa.

Madariaga (2009), doi:10.1007/978-1-4419-7695-6_22, eq. 27

ssp_util.**station_to_event_position**(*trace*)

Compute station position with respect to the event, in terms of hypocentral distance (km), epicentral distance (km), great-circle distance (degrees), azimuth and back-azimuth.

Values are stored in the trace stats dictionary.

ssp_util.**toDeg**(*radians*)

Convert radians to degrees.

ssp_util.**toRad**(*degrees*)

Convert degrees to radians.

ssp_util.**weighted_std**(*values*, *weights*)

Return the weighted standard deviation.

Parameters

- **values** – array of values
- **weights** – array of weights

Returns

weighted standard deviation

16.2.15 ssp_wave_arrival

16.2.16 config

Config class for sourcespec.

copyright

2013-2026 Claudio Satriano <satriano@ipgp.fr>

license

CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

class config.**Config**

Config class for sourcespec.

16.2.17 kdtree

Grid importance sampling using a k-d tree.

copyright

2022-2026 Claudio Satriano <satriano@ipgp.fr>

license

CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

class kdtree.**KDTCell**(*extent*, *calc_pdf*, *min_cell_prob=0*, *ndiv=None*, *maxdiv=None*)

A cell of a k-d tree.

divide(*parts*)

Divide the cell in *parts* parts along every dimension.

class kdtree.**KDTree**(*extent*, *init_parts*, *calc_pdf*, *min_cell_prob=0.0*, *maxdiv=None*)

A k-d tree.

divide()

Find the cell with highest probability and divide it in 2 parts along every dimension

get_pdf(*deltas*)

Calculate the probability density function (PDF) on a grid defined by deltas.

16.2.18 savefig

Save Matplotlib figure. Optimize PNG format using PIL.

copyright
2022-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

savefig.**savefig**(*fig*, *figfile*, *fmt*, *quantize_colors=True*, *use_threading=True*, ***kwargs*)

Save Matplotlib figure. Optimize PNG format using PIL.

Parameters

- **fig** (matplotlib.figure.Figure) – Matplotlib figure to save.
- **figfile** (str) – Output file name.
- **fmt** (str) – Output format (e.g., 'png', 'pdf', 'svg').
- **quantize_colors** (bool, *optional*) – If True (default), reduce the number of colors in the PNG image.
- **use_threading** (bool, *optional*) – If True (default), save the figure in a separate thread.
- ****kwargs** – Additional keyword arguments for *fig.savefig*.

16.2.19 spectrum

Define Spectrum and SpectrumStream classes, similar to ObsPy's Trace and Stream.

Provides the high-level function read_spectra() to read SpectrumStream objects from HDF5 or TEXT files.

copyright

2012-2026 Claudio Satriano <satriano@ipgp.fr>
Kris Vanneste <kris.vanneste@oma.be>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

class spectrum.**AttributeDict**(**args*, ***kwargs*)

A dictionary that allows attribute-style access.

class spectrum.**Spectrum**(*obspy_trace=None*)

A class to handle amplitude spectra.

Parameters
obspy_trace – An ObsPy Trace object to compute the spectrum from.

copy()

Return a copy of the spectrum.

property data

Return the array containing the amplitude spectrum.

property data_logspaced

Return the array containing the amplitude spectrum in logspaced frequencies.

property data_mag

Return the array containing the amplitude spectrum in mangitude units.

property data_mag_logspaced

Return the array containing the amplitude spectrum in logspaced frequencies in magnitude units.

property freq

Return the frequency axis of the spectrum.

property freq_logspaced

Return the logspaced frequency axis of the spectrum.

from_obspy_trace(*trace*)

Compute the spectrum from an ObsPy Trace object.

get_id()

Return the id of the spectrum.

property id

Return the id of the spectrum.

interp_data_logspaced_to_new_freq(*new_freq*, *fill_value=numpy.nan*)

Interpolate the logspaced data to a new frequency axis.

Parameters

- **new_freq** – The new frequency axis.
- **fill_value** – The value to use for extrapolation. Default is np.nan. If 'extrapolate', the data is extrapolated to the new frequency range. See scipy.interpolate.interp1d for more details.

interp_data_to_new_freq(*new_freq*, *fill_value=numpy.nan*)

Interpolate the linear data to a new frequency axis.

Parameters

- **new_freq** – The new frequency axis.
- **fill_value** – The value to use for extrapolation. Default is np.nan. If 'extrapolate', the data is extrapolated to the new frequency range. See scipy.interpolate.interp1d for more details.

make_freq_logspaced(*delta_logspaced=None*)

Create a logspaced frequency axis from the linear frequency axis.

If logspaced data already exist, it is reinterpolated to the new logspaced frequencies.

Parameters

delta_logspaced – The log10([Hz]) sample interval. If None, it is computed from the linear frequency axis.

make_linear_from_logspaced(*which='data_logspaced'*)

Convert the logspaced data to linear data.

Parameters
which – The data to convert. One of 'data_logspaced', 'data_mag_logspaced' or 'both'.

Note
The linear frequency axis must exist.

make_logspaced_from_linear(*which='data'*)

Convert the linear data to logspaced data.

Parameters
which – The data to convert. One of 'data', 'data_mag' or 'both'.

Note
The logspaced frequency axis must exist.

plot(***kwargs*)

Plot the amplitude spectrum.

slice(*fmin*, *fmax*, *nearest_sample=True*, *pad=False*, *fill_value=None*)

Slice the spectrum between fmin and fmax.

Parameters

- **fmin** – Minimum frequency.
- **fmax** – Maximum frequency.
- **nearest_sample** – If True, the slice will include the nearest frequency to fmin and fmax.
- **pad** – If True, the slice will be padded with the value of fill_value until fmin and fmax are included.
- **fill_value** – The value to use for padding.

Note
Only the linear spaced frequencies, data and data_mag are sliced. If the original spectrum contains logspaced frequencies, data, and data_mag, those are not preserved in the sliced spectrum.

write(*filename*, *format='HDF5'*, *append=False*, *hdf5_group=None*)

Write the spectrum to a file.

Parameters

- **filename** – The name of the file to write to.
- **format** – The format to use. One of 'HDF5' or 'TEXT'. Default is 'HDF5'.
- **append** – If True, append the spectrum to an existing file. Only valid for HDF5 format.
- **hdf5_group** – If provided, write directly to this HDF5 group instead of creating/opening a file. Only valid for HDF5 format.

class spectrum.**SpectrumStream**(*iterable=()*, */ (Positional-only parameter separator (PEP 570))*)

A class to handle a collection of amplitude spectra.

append(*spectrum*)

Append a spectrum to the collection.

select(***kwargs*)

Select a subset of the SpectrumStream.

sort(*reverse=False*)

Sort the SpectrumStream in place.

write(*filename*, *format='HDF5'*)

Write the SpectrumStream to a file.

Parameters

- **filename** – The name of the file to write to.
- **format** – The format to use. One of 'HDF5' or 'TEXT'.

spectrum.**read_spectra**(*pathname*, *format='HDF5'*)

Read a SpectrumStream from one ore more files.

Parameters

- **pathname** – The pathname of the files to read. Can contain wildcards.
- **format** – The format to use. One of 'HDF5' or 'TEXT'. Default is 'HDF5'.

Returns

The SpectrumStream object.

spectrum.**signal_fft**(*signal*, *delta*)

Compute the complex Fourier transform of a signal.

Parameters

- **signal** – The signal to transform.
- **delta** – The sampling interval.

Returns

The Fourier transform and the frequency axis.

16.2.20 clipping_detection

Check trace for clipping using kernel density estimation of the trace= amplitude values.

Two methods are available:

1. `clipping_score()`: compute a trace clipping score based on the shape of the kernel density estimation.
2. *`clipping_peaks()`*: check if trace is clipped, based on the number of peaks in the kernel density estimation;

copyright

2023-2026 Claudio Satriano <satriano@ipgp.fr>,
Kris Vanneste <kris.vanneste@oma.be>

license

CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

clipping_detection.**check_min_amplitude**(*trace*, *min_amplitude_ratio*)

Check if trace amplitude is above a minimum threshold computed from the instrument sensitivity.

Parameters

- **trace** (Trace) – Trace to check. Must have an attached inventory.

- **min_amplitude_fraction** (float) – Minimum trace amplitude, as a fraction of the overall sensitivity of the instrument, to check for clipping. Valid values are between 0 and 1.

Returns

- bool – True if trace amplitude is above the minimum threshold, False otherwise.
- **trace_max** (float) – Maximum amplitude of the trace, in counts
- **min_threshold** (float) – Minimum amplitude threshold, in counts

clipping_detection.**clipping_peaks**(*trace*, *sensitivity=3*, *clipping_percentile=10*, *debug=False*)

Check if a trace is clipped, based on the number of peaks in the kernel density estimation of the trace amplitude values.

The algorithm is based on the following steps:

1. The trace is demeaned.
2. A kernel density estimation is computed on the trace amplitude values.
3. The kernel density estimation is weighted by the distance from the zero mean amplitude value, using a parabolic function, between 1 and 5.
4. Peaks are detected in the weighted kernel density estimation. The sensitivity of the peak detection algorithm is controlled by the `sensitivity` parameter, based on which a minimum prominence threshold is set.
5. The trace is considered clipped if there is at least one peak in the amplitude range corresponding to the `clipping_percentile` parameter.

Parameters

- **trace** (obspy.core.trace.Trace) – Trace to check.
- **sensitivity** (int) – Sensitivity level, from 1 (least sensitive) to 5 (most sensitive). (default: 3). The sensitivity level controls the minimum prominence threshold used for peak detection in the kernel density estimation. See the scipy.signal.find_peaks() documentation for more details.
- **clipping_percentile** (float, between 0 and 100) – Percentile of trace amplitude range (expressed as percentage) to check for clipping. Default is 10, which means that the 10% highest and lowest values of the trace amplitude will be checked for clipping. A value of 0 means that no clipping check will be performed.
- **debug** (bool) – If True, plot trace, samples histogram and kernel density.

Returns

- **trace_clipped** (bool) – True if trace is clipped, False otherwise.
- **properties** (dict) – Dictionary with the properties used for the clipping detection. The dictionary contains the following keys:
  - npeaks (int): number of peaks found in the kernel density
  - npeaks_clipped (int): number of peaks found in the kernel density that are considered clipped
  - peaks (numpy.ndarray): list of peaks found in the kernel density
  - prominences (numpy.ndarray): list of prominences of the peaks found in the kernel density

clipping_detection.**compute_clipping_score**(*trace*, *remove_baseline=False*, *debug=False*)

Compute a trace clipping score based on the shape of the kernel density estimation of the trace amplitude values.

The algorithm is based on the following steps:

1. The trace is detrended and demeaned. Optionally, the trace baseline can be removed.
2. A kernel density estimation is computed on the trace amplitude values.
3. Two weighted kernel density functions are computed:
   - a full weighted kernel density, where the kernel density is weighted by the distance from the zero mean amplitude value, using a 8th order power function between 1 and 100.
   - a weighted kernel density without the central peak, where the kernel density is weighted by the distance from the zero mean amplitude value, using a 8th order power function between 0 and 100.

   In both cases, the weight gives more importance to samples far from the zero mean value. In the second case, the central peak is ignored.
4. The score, ranging from 0 to 100, is the sum of the squared weighted kernel density without the central peak, normalized by the sum of the squared full weighted kernel density. The score is 0 if there is no additional peak beyond the central peak.

Parameters

- **trace** (Trace) – Trace to check for clipping.
- **remove_baseline** (bool, *optional*) – If set, remove the trace baseline before computing the score.
- **debug** (bool, *optional*) – If set, plot the trace, the samples histogram, the kernel density (unweighted and weighted), the kernel baseline model, and the misfit. Default is False.

Returns

clipping_score – Clipping score, in percentage.

Return type

float

> **Note**
>
> Distorted traces (e.g., signals with strong baselines) can also get a high clipping score. To avoid this, the trace baseline can be removed.
>
> A debug mode is available to plot the trace, the samples histogram, the kernel density (unweighted and weighted), the kernel baseline model, and the misfit.

clipping_detection.**main**()

Main function

16.2.21 plot_sourcepars

plot_sourcepars.py

1D or 2D plot of source parameters from a sqlite parameter file.

copyright

2023-2026 Claudio Satriano <satriano@ipgp.fr>

license
CeCILL Free Software License Agreement v2.1 (http://www.cecill.info/licences.en.html)

class plot_sourcepars.**Annot**(*xdata*, *ydata*, *labels*, *yformat*, *xformat=None*)

Annotate the plot with the evid, Mw and the value of the parameter.

class plot_sourcepars.**Params**(*args*)

Class to handle the parameters from a sqlite file.

filter(*stamin=None*, *stamax=None*, *magmin=None*, *magmax=None*, *ssdmin=None*, *ssdmax=None*, *sigmaamin=None*, *sigmaamax=None*)

Filter the parameters based on one or more conditions.

plot_Er_mw(*hist=False*, *fit=False*, *nbins=None*, *colorby=None*, *colormap=None*)

Plot the logarithm of the radiated energy vs the moment magnitude.

Parameters

- **hist** (bool) – If True, plot a 2D histogram instead of a scatter plot.
- **fit** (bool) – If True, plot a linear regression of Er vs mw.
- **nbins** (int) – Number of bins for the 2D histogram.
- **colorby** (str) – Color the data points by the given parameter.
- **colormap** (str) – Use this colormap for the colorby parameter instead of the default.

plot_fc_mw(*hist=False*, *fit=False*, *slope=False*, *nbins=None*, *wave_type='S'*, *colorby=None*, *colormap=None*)

Plot the logarithm of the corner frequency vs the moment magnitude.

Parameters

- **hist** (bool) – If True, plot a 2D histogram instead of a scatter plot.
- **fit** (bool) – If True, plot a linear regression of fc vs mw.
- **slope** (bool) – If True, also fit the slope of the linear regression.
- **nbins** (int) – Number of bins for the 2D histogram.
- **wave_type** (str) – One of "P", "S", "SV" or "SH". Default is "S".
- **colorby** (str) – Color the data points by the given parameter.
- **colormap** (str) – Use this colormap for the colorby parameter instead of the default.

plot_gutenberg_richter(*magmin=None*, *magmax=None*, *nbins=None*, *fit=False*)

Plot the Gutenberg-Richter law.

Parameters

- **magmin** (float) – Minimum magnitude for the plot.
- **magmax** (float) – Maximum magnitude for the plot.
- **nbins** (int) – Number of bins for the histogram.
- **fit** (bool) – If True, find the magnitude of completeness and fit the Gutenberg-Richter law to the data.

plot_hist(*param_name*, *nbins=None*, *wave_type='S'*)

Plot a histogram of the given parameter.

Parameters

- **param_name** (str) – Name of the parameter to plot.
- **nbins** (int) – Number of bins for the histogram.
- **wave_type** (str) – One of "P", "S", "SV" or "SH". Default is "S".

plot_mw_time(*colorby=None*, *colormap=None*)

Plot the moment magnitude vs time.

Parameters

- **colorby** (str) – Color the data points by the given parameter.
- **colormap** (str) – Use this colormap for the colorby parameter instead of the default.

plot_ssd_depth(*hist=False*, *fit=False*, *nbins=None*, *colorby=None*, *colormap=None*)

Plot the logarithm of static stress drop vs depth.

Parameters

- **hist** (bool) – If True, plot a 2D histogram instead of a scatter plot.
- **fit** (bool) – If True, plot a linear regression of ssd vs depth.
- **nbins** (int) – Number of bins for the 2D histogram.
- **colorby** (str) – Color the data points by the given parameter.
- **colormap** (str) – Use this colormap for the colorby parameter instead of the default.

plot_ssd_mw(*hist=False*, *fit=False*, *nbins=None*, *colorby=None*, *colormap=None*)

Plot the logarithm of static stress drop vs moment magnitude.

Parameters

- **hist** (bool) – If True, plot a 2D histogram instead of a scatter plot.
- **fit** (bool) – If True, plot a linear regression of ssd vs mw.
- **nbins** (int) – Number of bins for the 2D histogram.
- **colorby** (str) – Color the data points by the given parameter.
- **colormap** (str) – Use this colormap for the colorby parameter instead of the default.

skip_events(*idx*)

Skip events with index idx.

plot_sourcepars.**apparent_stress_curve_Er_mw**(*sigma_a*, *mu*, *mw*)

Constant apparent stress curve in Er vs Mw.

Madariaga (2009), doi:10.1007/978-1-4419-7695-6_22, eq. 33., page 374.

plot_sourcepars.**calc_r2**(*x*, *y*, *yerr*, *a*, *b*)

Coefficient of determination.

plot_sourcepars.**compute_mc**(*magnitudes*, *magn_bin*, *vector_compl*, *pval*)

Compute the magnitude of completeness of a seismic catalog using the method in Taroni 2023 - TSR, https://doi.org/10.1785/0320230017.

This code is translated from the original MATLAB code provided by Matteo Taroni.

Parameters

- **magnitudes** (np.ndarray) – Vector of magnitudes.
- **magn_bin** (float) – Binning of the magnitudes (e.g., 0.01 or 0.1).
- **vector_compl** (np.ndarray) – Vector of completeness values to analyze.
- **pval** (float) – P-value threshold for the complete part of the catalog (e.g., 0.1).

Returns
mc – The magnitude of completeness of the catalog.

Return type
float

plot_sourcepars.**fc_mw_function**(*mw*, *a*, *b*)
Function to fit fc vs Mw.

plot_sourcepars.**fit_gutenberg_richter**(*bin_centers*, *cum_nevs*, *mc*)
Fit the Gutenberg-Richter law to the data points.

Parameters

- **bin_centers** (np.ndarray) – Bin centers.
- **cum_nevs** (np.ndarray) – Cumulative number of events.
- **mc** (float) – Magnitude of completeness.

Returns
a, b – G-R law parameters.

Return type
float

plot_sourcepars.**get_colormap**(*parameter*)
Get a colormap for the parameter.

plot_sourcepars.**get_param_label**(*parameter*)
Get the label for the parameter.

plot_sourcepars.**mag_to_moment**(*mag*, *b=0.5*)
Convert magnitude to moment.

The parameter b is used to change the slope of the fc-Mw stress drop curve.

The standard value of b is 0.5, which corresponds to a self-similar model.

plot_sourcepars.**main**()
Main function.

plot_sourcepars.**moment_to_mag**(*moment*, *b=0.5*)
Convert moment to magnitude.

The parameter b is used to change the slope of the fc-Mw stress drop curve.

The standard value of b is 0.5, which corresponds to a self-similar model.

plot_sourcepars.**parse_args**()
Parse command line arguments.

plot_sourcepars.**query_event_params_into_numpy**(*cursor*, *param*, *param_type*, *query_condition*)
Query a parameter from the Events table and return it as a numpy array.

plot_sourcepars.**run**()

Run the script.

plot_sourcepars.**stress_drop_curve_Er_mw**(*delta_sigma*, *mu*, *mw*)

Constant stress drop curve in Er vs Mw.

Madariaga (2009), doi:10.1007/978-1-4419-7695-6_22, eq. 33., page 374.

plot_sourcepars.**stress_drop_curve_fc_mw**(*delta_sigma*, *vel*, *mw*, *k=0.3724*, *b=-0.5*)

Constant stress drop curve in fc vs Mw.

Obtained by combining the equation for stress drop:

delta_sigma = 7/16 * Mo / a^3

with the equation for source radius:

a = k * vel * (delta_sigma / fc)

where k is a coefficient discussed in Kaneko and Shearer (2014).

For the Brune source model, k=0.3724.

Parameters

- **delta_sigma** (float) – Stress drop in MPa.
- **vel** (float) – P or S-wave velocity in km/s.
- **mw** (float) – Moment magnitude.
- **b** (float, *optional*) – The slope of the stress drop curve. Default is -0.5.
- **k** (float, *optional*) – Coefficient for the source radius. Default is 0.3724 (Brune source model).

Returns

fc – Corner frequency in Hz.

Return type

float

CHAPTER

SEVENTEEN

SOURCESPEC CHANGELOG

Earthquake source parameters from P- or S-wave displacement spectra

17.1 unreleased

Important: this release requires at least Python 3.9.

Note: several new config file parameters have been added and others have been replaced: please run `source_spec -U CONFIG_FILE_NAME` to update your old config file.

Warning: the SQLite database used by this version is not compatible with previous versions. You will need to upgrade your old database manually or using `source_spec -u DATABASE_FILE_NAME`.

17.1.1 Input/output

- Global inversion quality parameters in YAML, SQLite and HTML output:
  - Number of input stations
  - Number of input spectra
  - Number of spectra inverted
  - Azimuthal gap (primary and secondary)
  - Mean normalized RMS (RMSN)
  - Mean quality of fit (in percent)
  - Spectral dispersion (RMSN)
  - Spectral dispersion score (in percent)
- Station quality parameters in YAML and SQLite output:
  - Mean spectral S/N ratio, across the station components
  - Max spectral S/N ratio, across the station components
  - Normalized RMS (RMSN)
  - Quality of fit (in percent)
- The field `dist` in the `Stations` table of the SQLite database has been renamed to `hypo_dist`
- New fields in the `Stations` table of the SQLite database: `epi_dist`, `lon`, `lat`, `instr_type`, `spectral_snratio_mean`, `spectral_snratio_max`, `rmsn`, `quality_of_fit`, `ignored`, `ignored_reason`

- New fields in the `Events` table of the SQLite database: `n_input_stations`, `n_input_spectra` (renamed from `nobs`), `n_spectra_inverted`, `azimuthal_gap_primary`, `azimuthal_gap_secondary`, `rmsn_mean`, `quality_of_fit_mean`, `spectral_dispersion_rmsn`, `spectral_dispersion_score`
- HTML report improvements:
  - Inversion Quality Summary
  - Link to supplementary files and plots in the HTML report
  - Display configuration, log and output files in a modal window with syntax highlighting (only when the HTML report is served by a web server)
- More informative warning messages when signal windows are truncated or noise windows are zero-padded
- Improved writing of HDF5 spectrum files: the file is opened only once to write all the spectra
- New command line option `--depth` to override any event depth provided in the input files

17.1.2 Processing

- New config parameter `force_noise_zero_padding` to force noise window zero-padding regardless of the weighting mode
- New config parameter `win_length_min` to set a minimum window length for the spectral analysis. Useful when signal and noise windows are automatically defined
- New option `r_power_n_segmented` for the `geom_spread_model` config parameter to use a segmented geometrical spreading model with different powers for different distance ranges
- New config parameter `refine_theoretical_arrivals` to refine the theoretical P and S arrival times using a simple autopicker based on the smoothed envelope of the trace
- New config parameter `clipping_min_amplitude_ratio` to set a threshold for trace amplitude below which the trace is not checked for clipping
- Use spectral interpolation to compute and apply station residuals
- Limit spectrum and residual to common frequency range when applying correction before fitting (see #69)
- Possibility of defining a prior attenuation model, either as a constant Q value or as a frequency-dependent Q model from a text file (see #76). In this case, t* is not inverted but derived from the attenuation model.
- Possibility of computing travel times and takeoff angles from a 1D layered velocity model (see #79)
- Possibility of specifying station-specific free-surface amplification factors (see #81 and #82)

17.1.3 Post-Inversion

- New config parameter `pi_fc_weight_min`, which defines the minimum acceptable weight near the inverted fc. This helps flag cases where fc values may be poorly constrained
- New options for `source_residuals`:
  - `--runid` to select a specific run when multiple runs exist for the same event
  - `--exclude` to exclude 1 or more subfolders when parsing residuals files (see #68)
  - `--yrange` to specify a fixed range for the Y axis in the plots (see #68)

17.1.4 Plotting

- New plot: raw traces
- Stacked spectra: color spectral curves according to the weighting function
- Spectral plots: show information on the reason why a fit failed
- Spectral plots: add "ignored" next to the S/N ratio if the spectrum is ignored because of a S/N ratio too low or impossible to compute
- `plot_sourcepars`: possibility of selecting the latest runid for each event
- *plot_sourcepars*: new plot type: static stress drop vs. depth
- `plot_sourcepars`: option to color data points in scatter plot by another parameter
- `plot_sourcepars`: Gutenberg-Richter plot with optional fit of the magnitude of completeness and the a and b values
- *plot_sourcepars*: new plot type: Mw vs. time
- Support for Matplotlib 3.9, 3.10 and 3.11
- Maps: use PlateCarree projection for small maps (map diagonal below 100 km), which speeds up the plotting
- Maps: possibility to use a GeoTIFF file as a basemap
- Use threading to speed up saving plots to disk

17.1.5 Config file

- Consolidated seismic wave velocity and rock density parameters under the "EARTH MODEL PARAMETERS" section of the config file
- `vp_tt` and `vs_tt` renamed to `vp` and `vs`; they can be lists to define a layered model for travel-time and takeoff-angle computations. This is used together with the new `rho` and `layer_top_depths` parameters. These parameters are also used to convert station displacements into seismic moment, unless overridden by `vp_source`/`vp_stations`, `vs_source`/`vs_stations` or `rho_source`/`rho_stations`
- New config parameter: `force_noise_zero_padding`
- New config parameter: `win_length_min`
- Improved documentation for the `win_length` parameter
- New option `r_power_n_segmented` for the `geom_spread_model` config parameter
- New config parameters: `geom_spread_n_exponents`, `geom_spread_n_distances`
- New config parameters: `refine_theoretical_arrivals`, `autopick_freqmin`, `autopick_debug_plot`
- New config parameter `clipping_min_amplitude_ratio`
- New option `geotiff` for `plot_map_style` to use a GeoTIFF file as a basemap
- New config parameters: `plot_map_geotiff_filepath`, `plot_map_geotiff_grayscale`, `plot_map_geotiff_attribution`
- Improved documentation for the `sn_min` and `spectral_sn_min` parameters
- New option `rp_lower_bound` to avoid overcorrection for stations close to a nodal plane when radiation coefficient is computed from focal mechanism
- New config parameter `pi_fc_weight_min`

- Config parameter `pi_misfit_max` renamed to `pi_quality_of_fit_min`: it now defines the minimum acceptable quality of fit in percent (0-100).
- Config parameter `noise_pre_time` now defaults to `None`, which means that it will be autoset to the length of the signal window plus the value of `signal_pre_time`
- New config parameter `Q_model` to define a prior attenuation model (constant or frequency-dependent, see #76)
- Config parameter `free_surface_amplification` now also accepts a list of station code patterns with corresponding amplification factors (see #81 and #82)

17.1.6 Bugfixes

- Fix `np.float64` being printed in logs and YAML output when using NumPy>=2
- Fix for rejected spectra still being plotted in the stacked spectra plot
- Fix for corner case where all the inversion errors are zero
- Fix I/O error when reading PAZ files
- Fix for event ids in SourceSpec event file being only numbers: they are now correctly interpreted as strings
- Fix for ignored picks for certain kind of QuakeML files (like the ones from USGS) where the phase name is not in the <phase_hint> attribute of the <pick> element but in the <phase> attribute of the corresponding <arrival> element
- Stabilize the computation of derivatives in the Okal geometrical spreading model, thus avoiding numerical instabilities
- Speed up map plotting when using coastlines
- Gently skip building spectra for station with very short data windows (less than 10 samples)
- Fix for `sensitivity` config parameter always requiring a SAC file. Now, if `sensitivity` is a numerical value, any file format is accepted
- Improved estimation of `fc_0` (initial corner frequency) when inverse frequency weighting is used (see #67)
- Fix in `source_residuals`: removed extrapolation of individual station residuals beyond their valid frequency range, which was leading to incorrect mean residuals at high frequencies (see #68)
- Fix bug preventing reading travel times from NonLinLoc grid files
- Improved handling of glob patterns in trace ID filtering (`use_traceids` and `ignore_traceids` config parameters)
- Fix for config parameter `noise_pre_time` not being able to be set to `None`
- Fix a bug causing severe slowdown when saving many plots to disk
- Fix for geophone instrument codes (`P`) not being recognized

17.1.7 Requirements

- Python minimum version raised to 3.9
- ObsPy minimum version raised to 1.5
- Support added for Python 3.13 and 3.14

17.2 v1.8 - 2024-04-07

This long overdue release brings many improvements, new features and bugfixes gradually introduced during the last year.

Release highlights:

- New file formats for events and spectra
- New configuration options to better specifify velocity and density models
- Better support for P-wave inversion and teleseismic events
- Radiation pattern correction from focal mechanism in radiated energy computation
- Option for travel time-based signal window length
- More options to control the calculation of source radius and stress drop
- Improved estimation of radiated energy
- New source parameter: apparent stress
- Fix for map tiles not plotted anymore

This release requires at least Python 3.7.

Warning: the SQLite database used by this version is not compatible with previous versions. You will need to upgrade your old database manually or using `source_spec -u DATABASE_FILE_NAME`.

Make sure to read the detailed Changelog below

17.2.1 v1.8: Input/output

- Introducing a new file format for providing event information (hypocentral location, magnitude, focal mechanism, moment tensor): the SourceSpec Event File.
- New HDF5 and TEXT file formats to store spectra
- Station residuals are now saved in an HDF5 spectrum file, instead of a pickle file
- New config file option `save_spectra` to save the spectra to an HDF5 file in the output directory
- Changes in the YAML output file:
  - `bsd` (Brune stress drop) parameter renamed to `ssd` (static stress drop)
  - Store in the `event_info` section the values of vp, vs and rho close to the hypocenter
  - Store in the `inversion_info` section the type of wave used for the inversion (P, S, SV or SH)
- Changes in the SQLite database (warning: these changes break compatibility with previous database versions):
  - `bsd` (Brune stress drop) parameter renamed to `ssd` (static stress drop)
  - Store the `Stations` table information on whether each parameter is an outlier (see #38)
  - Make place in the `Stations` table for station-level errors on radiated energy (even if they are currently not computed)
  - Store in the `Events` table the number of observations used for computing each summary parameter
  - Store in the `Events` table weighted means for radiated energy and local magnitude, even if those means are currently the same as the simple means, since those parameters do not have station-level errors defined
  - New columns for apparent stress in both `Events` and `Stations` tables
  - Store in the `Events` table the values of vp, vs and rho close to the hypocenter

  - Store in the `Events` table the type of wave used for the inversion (P, S, SV or SH)
- New command line option (`-u` or `--updatedb`) to update an existing database from a previous version
- Input files are now linked symbolically in the `input_files` subdirectory of the output directory (not implemented for Windows)
- New command line option (`-R` or `--run_id_subdir`) to use `run_id` (if defined) as a subdirectory of the event directory
- Print event info to console and to log file
- HTML report improvements:
  - Event name in the summary table, if available
  - Author, agency and run completion date in the summary table
  - SourceSpec version in the inversion information table
  - Link to input files
  - Information on the type of wave used for the inversion (P, S, SV or SH)

17.2.2 v1.8: Processing

- Use all the available components to compute P-wave spectra (previously, only the vertical component was used)
- Possibility of specifying a free surface amplification factor different from 2
- Possibility of specifying a layered velocity and density model for the source
- Possibility of specifying a different density for the source and for the stations
- If density is not provided (i.e., it is `None`), use the density from the global velocity model "iasp91"
- Teleseismic geometrical spreading model (Okal, 1992)
- New weighting option based on inverse frequency, so that lower frequencies have larger weight in the inversion. If traces contain noise, weights will be set to zero where SNR < 3 (see #37)
- For weights computed from spectral S/N ratio (noise weighting), set to zero all the weights below 20% of the maximum weight, so that these weakly constrained parts of the spectrum are ignored in the inversion
- Possibility of using variable signal window lengths for each station as a function of the travel time of the P or S wave (see #48)

17.2.3 v1.8: Inversion

- Possibility of using the magnitude (or scalar moment) provided in the event file as initial Mw value for the inversion
- Reintroduced the possibility of providing the variability around the initial Mw value
- By combining the previous options, it is now possible to fix the Mw value during the inversion to the value provided in the event file

17.2.4 v1.8: Post-Inversion

- Possibility of choosing the "k" coefficient to compute source radius from corner frequency (Kaneko and Shearer, 2014)
- Better control on the frequency range used for computing radiated energy (see #49)
- Use station-specific radiation pattern (when available) for computing radiated energy

- Take into account for energy partition when computing radiated energy (Boatwright and Choy, 1986). This affects mostly the radiated energy computed from P waves
- New source parameter: apparent stress
- For parameters with no station-level uncertainty defined (currently, radiated energy and local magnitude), use simple mean when computing summary weighted averages (the previous behavior was to not compute weighted averages for these parameters)

17.2.5 v1.8: Plotting

- Show the station radiated energy (Er) value on the station spectra plots
- Show the summary radiated energy (Er) value on the stacked spectra plot
- Station maps improvements:
  - Possibility of choosing a basemap style or no basemap
  - Possibility of not plotting the coastlines
  - Exclude outliers when computing colorbar limits
  - Improved computation of bounding box for regional and teleseismic events
  - Use a global orthographic projection when using stations at large teleseismic epicentral distances (more than 3000 km)
- Changes to `plot_sourcepars`:
  - Read vp, vs and rho from the SQLite database (previously: vs was hardcoded to 3.5 km/s, rho to 2700 kg/m3 and vp was not used)
  - Read the source radius "k" coefficient from the SQLite database (previously: "k" was hardcoded to 0.3724, value for the Brune model)
  - New command line option `--wave_type` to select the wave type (P, S, SV or SH) for plots involving the corner frequency
  - Possibility of plotting histogram of apparent stress
  - Option to filter events by apparent stress

17.2.6 v1.8: Config file

- New config parameter `epi_dist_ranges` to select stations within one or more ranges of epicentral distances. It replaces the old parameter `max_epi_dist`.
- New config parameter `free_surface_amplification` to specify the free surface amplification factor (default: 2)
- New config parameter `layer_top_depths` to specify the depth of the top of the layers in a layered velocity and density model
- The config parameters `vp_source`, `vs_source` and `rho_source` can now be lists of values, to specify a layered velocity and density model for the source
- Config parameter `rho` renamed to `rho_source`
- New config parameter `rho_stations`
- New config parameter `geom_spread_min_teleseismic_distance` to set the minimum epicentral distance for using the teleseismic geometrical spreading model
- New config parameters `kp` and `ks` to set the "k" coefficient for computing source radius from corner frequency

- Config parameter `pi_bsd_min_max` renamed to `pi_ssd_min_max`
- New option `inv_frequency` for the config parameter `weighting` (see #37)
- Config parameter `max_freq_Er` replaced by `Er_freq_range` (see #49)
- New parameters, `qml_event_description` and `qml_event_description_regex`, to obtain the event name from the QuakeML event "description" tag
- New parameter `Mw_0_from_event_file` to use the magnitude (or scalar moment) provided in the event file as initial Mw value for the inversion
- Reintroduced the parameter `Mw_0_variability` to set the variability around the initial Mw value
- New parameter `plot_save_asap` to save plots as soon as they are ready. This uses less memory but slows down the code.
- New parameter `plot_map_style` to choose the map style
- New parameter `plot_map_api_key` to provide a Stadia Maps api key for Stamen Terrain basemap
- New option for the parameter `plot_coastline_resolution`: `no_coastline`
- New config parameter `variable_win_length_factor` to specify window length as a fraction of the travel time of the P/S wave (see #48)

17.2.7 v1.8: Bugfixes

- Fix source radius computation when using P waves (use P-wave velocity instead of S-wave velocity)
- Do not ignore picks labeled with lowercase "p" or "s"
- Fixed: config parameter `p_arrival_tolerance` was used also for S waves, instead of `s_arrival_tolerance` (see #35)
- Fix Boatwright spreading model (log10 instead of natural log)
- Fix bug where signal and noise windows were plotted with the wrong length, under certain circumstances (see #35)
- Fixes related to records with short signal windows (see #39)
- Fix for beachball not plotted anymore with recent versions of Matplotlib.
- Fix bug where traces ignored because of low spectral S/N ratio, where still plotted as if they were valid traces
- Fix bug when specifying an absolute path for output directory: the path was treated as relative (see #40)
- Fix bug where paths starting with tilde (~) were not parsed correctly (see #43 and #44)
- Fix bug where local magnitude was not written to the HYPO71 output file, when using weighted mean as reference statistics
- Fix for Stamen Terrain basemap now requiring an API key from Stadia Maps

17.2.8 v1.8: Requirements

- Python minimum version raised to 3.7
- Matplotlib minimum version raised to 3.2
- Cartopy minimum version raised to 0.21

17.3 v1.7 - 2023-03-31

This release improves trace processing through the use of modern routines for instrument correction, optional baseline removal, a new clipping detection algorithm and a better definition of signal and noise time windows.

A new plot is introduced, "stacked spectra" which allows to compare all the spectra at once (and easily detect problematic stations). Also, a new command line tool, `plot_sourcepars`, allows making aggregate plots of source parameters for many events (starting from the SQLite database).

New config file parameters have been added, while some have been removed. Please run `source_spec -U CONFIG_FILE_NAME` to update your old config file.

As always, many bugfixes and improvements have been made in this release. Thanks to all the users who took time to write and ask questions (by mail or using the official SourceSpec Discussions).

Big kudos to Kris Vanneste @krisvanneste who helped all along the development with code review and testing and submitted pull requests on noise windows and clipping detection.

Below is the detailed Changelog

17.3.1 v1.7: Input/output

- Possibility of using a single PAZ file as a "generic" PAZ file for all the stations
- Command line option `--station_metadata` (or `-w`) for overriding the config file parameter with the same name (see pull request #16)
- Removed command line option `--no-response` for avoiding removing instrument response (use the config option `correct_instrumental_response` instead)
- New output file in YAML format. The old `.out` file is still available but deprecated.
- Information on the inversion procedure in YAML and HTML output
- Option to add an agency logo to the HTML page
- Possibility of generating HTML report without figures (see #30)

17.3.2 v1.7: Processing

- Use modern ObsPy `trace.remove_response()` routine for instrument correction (see #27)
- Option to remove the trace baseline after instrument correction and before filtering (`remove_baseline` config parameter) (see #25)
- New algorithms for clipping detection based on kernel density estimation of the trace amplitude values (see #23, #24, #25)
  - Two methods are available:
    - `clipping_score`: compute a trace clipping score based on the shape of the kernel density estimation.
    - `clipping_peaks`: check if trace is clipped, based on the number of peaks in the kernel density estimation;
  - Use `clipping_detection_algorithm` in the config file to choose the algorithm and the other `clipping_*` parameters to adjust the results.
  - The algorithms can also be called from the command line, e.g. for debug purposes, using the shell command `clipping_detection`.
- Relax noise window requirements if noise weighting is not used. This is useful for older triggered records with noise windows that are short or even missing entirely (see pull request #18)

- Some small improvements were made in the window definitions (see pull request #18):
  - Generate error if signal window is incomplete (P- or S-arrival before the start time of the trace)
  - Warn if noise window overlaps with P-window (instead of S-window, as in previous versions)
  - Constrain `signal_pre_time` for S-phase to half the S-P interval, if this interval is shorter than `signal_pre_time` (i.e., for short-distance records with short S-P interval)
- Extract source and station P and S velocities from global 'iasp91' velocity model, if both `v(p,s)_source` and `v(p,s)_stations` are set to `None` (see #20)
- Magnitude limits for inversion are now autoset between 90% of the minimum of the spectral plateau and 110% of its maximum (see #22)

17.3.3 v1.7: Post-Inversion

- Possibility of choosing the reference summary statistics that will be used for map plots, QuakeML and HYPO output, as well as for the "Event Summary" section in HTML report and for computing station spectral residuals. Available summary statistics are:
  - mean
  - weighted_mean
  - percentiles (new!)
- Possibility of defining the number of sigmas for uncertainties on event means and weighted means

17.3.4 v1.7: Plotting

- New plot: stacked spectra
- Do not zero-pad traces to common length when plotting, so that missing data at beginning or at the end can be easily detected (see #21)
- Plot noise and signal windows separately for each component (see #21)
- Show on the trace plot the reason why a trace has been ignored
- Logscale for boxplots, if parameters span a large interval (see pull request #15)
- Support for SVG format for plot files (can be used in HTML output as alternative to PNG)
- Improved trace plot quality for vector formats (PDF, SVG)
- New command line tool: `plot_sourcepars` to make 1D or 2D plot of source parameters from a sqlite parameter file.

17.3.5 v1.7: Config file

- Removed `sensitivity_only` option from `correct_instrumental_response`
- Removed config parameter: `Mw_0_variability`
- Removed config parameter: `clip_max_percent`
- New config parameter: `remove_baseline`
- New config parameters for clipping detection:
  - `clipping_detection_algorithm`
  - `clipping_debug_plot`
  - `clipping_score_threshold`

  - `clipping_peaks_sensitivity`
  - `clipping_peaks_percentile`
- Config file section `AVERAGES PARAMETERS` renamed to `SUMMARY STATISTICS PARAMETERS`
- New config parameter: `reference_statistics`
- New config parameter: `n_sigma`
- New config parameters for percentiles calculation: `lower_percentage`, `mid_percentage` and `upper_percentage`
- New config parameters for filtering and spectral windowing of displacement signals:
  - `bp_freqmin_disp`, `bp_freqmax_disp`
  - `freq1_disp`, `freq2_disp`
- Default values for `t_star_min_max` (instead of `None`)

17.3.6 v1.7: Code improvements

- Large refactoring of the whole codebase, to make the code more modern and easier to maintain (see #28)

17.3.7 v1.7: Bugfixes

- Properly ignore vertical components when `ignore_vertical` is `True`
- Fix a bug preventing reading phase picks from HYPOINVERSE-2000 files
- Fix for noise window not showing up in PNG trace plots in some cases
- Fix reading velocities from NLL model (see #20)
- HTML report: better scrollbars for station table across all the browsers
- Fix for cropped map for very large station-to-event distances (greater than 500 km)
- Fix a bug in generating evid form origin time when reading origin time from SAC header and the number of seconds was 59
- Fix a crash when no map tiles were available at the selected zoom level
- Fix for a corner case where the three components of the same instrument have different trace length (see #31)
- Fix `source_residuals`, which didn't work anymore

17.4 v1.6 - 2022-08-02

This release introduces several modifications to the config file. You will need to upgrade your old config files manually or using `source_spec -U CONFIG_FILE_NAME`.

This release requires at least Python 3.6.

A lot of effort has been devoted to improve the documentation. Please check it out on https://sourcespec.readthedocs.io/

17.4.1 v1.6: Input/output

- QuakeML output (when using QuakeML input)
- Command line option `--run-id` to provide a string identifying the current run (see pull request #6)
- Write SourceSpec version to parfile

- Write SourceSpec version and run complete time to SQLite file
- Write author and agency info (if specified) to output files, figures and HTML report
- HTML page for misfit plots (when using grid search or importance sampling)
- Station table in HTML report is now sortable (and its header remains fixed)!
- Reduce PNG figures file size, while improving their resolution
- Removed option to read event information and traces from a pickle file (rarely used)

17.4.2 v1.6: Processing

- Support for P-wave spectral inversion (see pull request #9)
- It is now possible to provide different vp and vs velocities, close to the source and close to the stations (see the new config options above and issue #5)
- Possibility to choose a geometrical spreading model between (see issue #8):
  - r (default: n=1 – body waves)
  - Boatwright et al. (2002): "r" below a cutoff distance, frequency-dependent above the cutoff distance
- Use travel time to compute quality factor from t* (and viceversa) (see issue #5)
- Compute travel time from pick and origin time, when possible (see issue #10)
- Warn if noise window ends after P or S arrival

17.4.3 v1.6: Post-Inversion

- Subtract the integral of noise spectrum from the integral of signal spectrum when computing radiated energy, under the hypothesis that energy is additive and noise is stationary

17.4.4 v1.6: Config file

- Config parameter `paz` has been removed and merged into `station_metadata`
- Config parameters `vp` and `vs` have been renamed to `vp_source` and `vs_source` (see issue #5)
- New, optional, config parameter `vp_stations` and `vs_stations`
- Config parameter `pre_p_time` and `pre_s_time` have been renamed to `noise_pre_time` and `signal_pre_time`, respectively (see pull request #9)
- Config parameter `rps_from_focal_mechanism` renamed to `rp_from_focal_mechanism` (see pull request #9)
- New config parameter: `geom_spread_model` (see issue #8)
- Config parameters `PLOT_SHOW`, `PLOT_SAVE` and `PLOT_SAVE_FORMAT` are now lowercase (`plot_show`, `plot_save` and `plot_save_format`)
- New, optional, general config parameters for specifying author and agency information. This information is written to output files and figures, if specified
- New config parameter, `event_url`, to link the event page from the HTML report
- Removed `DEBUG` config parameter
- Parameters from `GENERAL PARAMETERS` section reorganized into a new section called `TRACE AND METADATA PARAMETERS`

- Some parameters from `INVERSION PARAMETERS` moved into a new section called `SPECTRAL MODEL PARAMETERS`

17.4.5 v1.6: Bugfixes

- Fix for not working `weighting` options: `frequency` and `no_weight`
- Fix for negative weights occasionally generated by interpolation
- Fix bug when event coordinates are written into sqlite as binary blobs

17.5 v1.5 - 2022-05-22

This is a pretty big release coming after several months of work on a large dataset of more than 5000 events in Mayotte.

Please read through the changelog to discover all the improvements and new features.

You will need to update your old config files via `source_spec -U CONFIG_FILE_NAME`

Note that v1.5 is no more compatible with Python 2!

17.5.1 v1.5: Input/output

- Write output files into a subdirectory of OUTDIR, whose name is the event id
- Support for HYPOINVERSE-2000 output files
- Removed autodetection of hypo71 file paths (specific to CRL case)
- SQLite output: added radiated energy, weighted averages, errors on parameters, number of observations, hypocentral location and origin time
- Removed `-C` argument to apply station correction to spectra. Now spectra are automatically corrected if `residuals_filepath` is specified in the configuration file
- Save an additional event parameter to output files: average quality factor
- Save additional station parameters to output files: source radius, Brune stress drop, source radius, quality factor
- Mark outliers in `.out` file and in html report
- Colored console output for log messages! (Not supported on Windows)

17.5.2 v1.5: Processing

- New parameter for setting the width of the spectral smoothing window in terms of frequency decades: `spectral_smooth_width_decades` (see issue #2)
- Compute spectral weights after spectral correction (when a station residuals file is specified via `residuals_filepath`)
- Removed configuration parameter `trace_format`
- New configuration parameter `sensitivity` to provide a constant sensor sensitivity (flat response curve), which overrides any response curve provided in metadata
- New parameter for manually specifying trace units: `trace_units` (defaults to `auto`)
- New approach for trace clipping detection (requires just one configuration parameter, named `clip_max_percent`)
  - Check for trace clipping only in the processing window
  - Use histogram of samples to detect clipping

- Fix for wrong component used for 'SV' spectra (see issue #3)

17.5.3 v1.5: Inversion

- New config option: `Mw_0_variability`. Allowed variability around `Mw_0` during the main inversion. Previously hardcoded to 0.1
- New inversion methods for grid sampling:
  - grid search (very slow!)
  - importance sampling of the misfit grid using k-d tree (faster, but less accurate)
- Fix for Basin-hopping algorithm not running

17.5.4 v1.5: Post-Inversion

- New set of post-inversion parameters to reject certain inversion results, per-station: `pi_fc_min_max`, `pi_t_star_min_max`, `pi_bsd_min_max`, `pi_misfit_max`
- Reject inversion results when inverted `fc` is within 10% of `fc_min` or `fc_max`
- Fix: use logarithmic error width for weighted logarithmic averages
  - previous way of computing weighted logarithmic averages was not correct!
- Option to reject outliers using the IQR (interquartile range) method: parameter `nIQR`
- Support for non symmetric error on station spectral parameters
- Compute additional, per-station parameters: source radius, Brune stress drop and quality factor
- Compute errors for all station parameters
- Compute weighted averages for all event parameters (except radiated energy)
- Compute spectral residuals using weighted average spectral parameters

17.5.5 v1.5: Plotting

- Source parameter box plots to evaluate parameter dispersion across stations and visually detect outliers
- Misfit plot (2D and 1D) when using grid sampling
- `cartopy` removed as installation dependency, since it is not easily installable via `pip`
- Use GSHHS database to draw coastlines.
  - New config option: `plot_coastline_resolution`
- Correctly show circles on maps with diagonal smaller than 1 km
- Fix plotting map colorbar on Matplotlib 3.5
- Make average and errorbar lines more visible on map colorbar
- Fix for error on plotting fc map, when only one station is available
- Fix trace plot scaling for traces with larger signal outside the plot window
- Do not plot 'H' spectrum if there is only one instrument component (since it will coincide with the only component)
- Plot uncorrected spectrum when station correction is used

17.6 v1.4 - 2021-10-13

- New config option `rps_from_focal_mechanism` to compute station-specific S-wave radiation pattern from focal mechanism, if a focal mechanism is available in the QuakeML file
- Plot the focal mechanism on maps, if it is available
- Change default inversion algorithm to TNC (truncated Newton algorithm)
- Config option `dataless` renamed to `station_metadata`
- Config option `traceids` renamed to `traceid_mapping_file`
- Config options `ignore_stations` and `use_stations` renamed to `ignore_traceids` and `use_traceids`, respectively
- Support for 2D NonLinLoc grids (via `nllgrid >= 1.4.1`)
- Possibility of using a generic `DEFAULT` NonLinLoc time grid
- Added `cartopy` as an installation dependency
- Fixed: `nllgrid` was always requested at runtime
- Fixed: gracefully handle the case when there is no internet connection and map tiles cannot be downloaded
- Fixed (Windows): suppress colored terminal output, which is not supported
- Fixed (Windows): it is now possible to relaunch the same run, without having to delete the output directory first
- Fixed (Windows): use same timezone names than Linux and macOS

17.7 v1.3.1 - 2021-09-13

- Fix for HTML report not showing trace and spectral plots
- HTML report: add Corner Frequency in Event Summary

17.8 v1.3 - 2021-08-20

- HTML reports
- Option to provide station-specific spectral windowing

17.9 v1.2 - 2021-05-20

- Use `python-versioneer` to manage version numbers

17.10 v1.1 - 2021-04-07

- Bug fixes:
  - Accept band code `C` for broadband seismometers sampled at >=250 Hz
  - Require `cartopy>=0.18` for compatibility with `matplotlib>=3.3`

17.11 v1.0 - 2021-03-03

- Simplification of time window parameters:
  - an unique window length, `win_length`, is used for time-domain S/N ratio calculation and spectral analysis
  - the old parameters, `noise_win_length` and `s_win_length`, are no more supported and generate an error
- Reorganized config file and improved inline documentation within the file
  - removed unused option: `fc_0`
  - removed `pre_filt` option
  - post-processing check on `fc` bounds has been removed
- Option to specify non standard instrument codes (e.g., `L` for acceleration)
- Option to specify filter frequencies for local magnitude computation
- Plotting improvements:
  - Show S/N ratio on trace plots
  - Show spectral S/N ratio on spectrum plots
  - Option to show/hide ignored traces/spectra (low S/N)
- Bug fixes:
  - Pay attention to location code when building average spectra
  - Plotting: avoid overlapping traces with different location code
  - Plotting: avoid overlapping spectra with different location code

17.12 v0.9 - 2020-04-24

- Support for QuakeML input and StationXML
- Support for Python 3.5
- Only compatible with ObsPy >= 1.1.0
- Project reorganization:
  - Project renamed to SourceSpec
  - `ssp_residuals` renamed to `source_residuals`
  - New installable package (e.g., via `pip`)
- Spectra are smoothed in log-freq (no more Konno-Ohmachi)
- Inversion is performed in a log-freq space
- Option to invert for `t_star_0` on the plateau level
- Traces are filtered before computing S/N ratio
- Trace clipping detection
- Traces are always plotted, even if no inversion is performed
- Use by default a global model for theoretical travel time calculation
- Possibility of using NonLinLoc travel time grids (requires `nllgrid`)
- New options for P and S arrival time tolerance

- New option for maximum epicentral distance for trace processing
- Possibility of using a NonLinLoc model grid for obtaining vs at the source and at the receiver (requires `nllgrid`)
- Use `log10` of weighting function, since the inversion is done in magnitude units
- Use `json` format for `traceid` correction file
- Save config file to output dir (only for `source_spec`)
- Save run completion time into output file
- Logarithmic average and logarithmic (asymmetric) error bars for Mo, fc and source radius
- Computation of radiated energy
- Station-specific filters
- New parameters: `gap_max`, `overlap_max`
- Add legend to spectral plots
- Add event information, code version and run completion time to plots
- Multifigure plotting for traces and spectra (for large number of stations)
- New option to plot a station map, color-coded by station magnitude (requires Cartopy)
- Refactoring of local magnitude computation code:
  - Wood-Anderson amplitude is computed from the whole trace converted to W-A and not converting only the min and max peaks (which is the default in ObsPy)
  - Trace windowing is computed on HF envelopes
  - New option for custom Ml coefficients
  - Remove unnecessary Ml options
- Code cleaning and optimization:
  - Switch from `optparse` (deprecated) to `argparse`
  - Code style fixes and refactoring
- New option to update a config file from previous versions
- BUGFIX: Fix a major bug in reading hypo pick file
- BUGFIX: Fix for pick file not read in `source_model`

17.13 v0.8 - 2014-07-11

- Trace plot showing S and noise time windows
- Improved handling of paz files
- Per-station Mo on output file
- Code cleaning and optimization

17.14 v0.7 - 2014-04-07

- Code reorganization:
  - inversion code split to its own functions

- Option to use bounded inversion
- Station residuals, through `ssp_residuals.py`
- `source_model` can now output tables of trade-off between parameters
- Fix in the way noise trace is computed and processed
- Documentation!

17.15 v0.6 - 2013-06-05

- Signal to noise weighting
- Improved local magnitude computation
- New options:
  - time domain integration
  - vertical component
- Largely improved plotting function
- More data formats supported (Antilles, IPOC)
- `source_model`: a code for plotting theoretical spectra
- Code refactoring

17.16 v0.5 - 2013-02-10

- Azimuth computation
- Construction of an overall database
- Local magnitude computation
- konnoOhmachiSmoothing

17.17 v0.4 - 2012-04-10

- Logging infrastructure
- Code reorganization

17.18 v0.3 - 2012-02-10

- Output is no more printed at screen, but to file
- The plots can be saved to a file as well
- We differentiate between short periods and broad bands

17.19 v0.2 - 2012-02-06

Extended and generalized for the CRL application.

17.20 v0.1 - 2012-01-17

Initial Python port.

The following is a list, probably incomplete, of journal and conference papers that have used SourceSpec and that I'm aware of. If you have used SourceSpec in a publication and would like to have it listed here, please let me know.

17.21 Journal papers

17.22 Conference papers

CHAPTER

EIGHTEEN

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[1] R. Madariaga. Earthquake scaling laws. In *Extreme Environmental Events*, pages 364–383. Springer New York, New York, NY, 2011. URL: https://www.researchgate.net/publication/226065848_Earthquake_Scaling_Laws, doi:10.1007/978-1-4419-7695-6_22.

[2] J. N. Brune. Tectonic stress and the spectra of seismic shear waves from earthquakes. *Journal of Geophysical Research (1896-1977)*, 75(26):4997–5009, 1970. doi:10.1029/JB075i026p04997.

[3] J. Boatwright, G. L. Choy, and L. C. Seekins. Regional Estimates of Radiated Seismic Energy. *Bulletin of the Seismological Society of America*, 92(4):1241–1255, 05 2002. doi:10.1785/0120000932.

[4] E. A. Okal. A Student's Guide to Teleseismic Body Wave Amplitudes. *Seismological Research Letters*, 63(2):169–180, April 1992. doi:10.1785/gssrl.63.2.169.

[5] M. Lancieri, R. Madariaga, and F. Bonilla. Spectral scaling of the aftershocks of the Tocopilla 2007 earthquake in northern Chile. *Geophysical Journal International*, 189(1):469–480, 04 2012. doi:10.1111/j.1365-246X.2011.05327.x.

[6] Y. Kaneko and P. M. Shearer. Seismic source spectra and estimated stress drop derived from cohesive-zone models of circular subshear rupture. *Geophysical Journal International*, 197(2):1002–1015, March 2014. doi:10.1093/gji/ggu030.

[7] R. Madariaga. Dynamics of an expanding circular fault. *Bulletin of the Seismological Society of America*, 66(3):639–666, June 1976. doi:10.1785/bssa0660030639.

[8] T. Sato and T. Hirasawa. Body wave spectra from propagating shear cracks. *Journal of Physics of the Earth*, 21(4):415–431, 1973. doi:10.4294/jpe1952.21.415.

[9] M. Di Bona and A. Rovelli. Effects of the bandwidth limitation of stress drops estimated from integrals of the ground motion. *Bulletin of the Seismological Society of America*, 78(5):1818–1825, 10 1988. URL: http://bssa.geoscienceworld.org/cgi/doi/10.1785/BSSA0780051818, doi:10.1785/BSSA0780051818.

[10] J. Boatwright and G. L. Choy. Teleseismic estimates of the energy radiated by shallow earthquakes. *Journal of Geophysical Research*, 91(B2):2095, 1986. doi:10.1029/jb091ib02p02095.

[J1] C. Duverger, S. Lambotte, P. Bernard, H. Lyon-Caen, A. Deschamps, and A. Nercessian. Dynamics of microseismicity and its relationship with the active structures in the western Corinth Rift (Greece). *Geophysical Journal International*, 215(1):196–221, June 2018. doi:10.1093/gji/ggy264.

[J2] M. Laporte, L. Bollinger, H. Lyon-Caen, R. Hoste-Colomer, C. Duverger, J. Letort, M. Riesner, B. P. Koirala, M. Bhattarai, T. Kandel, C. Timsina, and L. B. Adhikari. Seismicity in far western Nepal reveals flats and ramps along the Main Himalayan Thrust. *Geophysical Journal International*, 226(3):1747–1763, April 2021. doi:10.1093/gji/ggab159.

[J3] F. Masson, S. Auclair, D. Bertil, M. Grunberg, B. Hernandez, S. Lambotte, G. Mazet-Roux, L. Provost, J. Saurel, A. Schlupp, and C. Sira. The Transversal Seismicity Action RESIF: A Tool to Improve the Dis-

tribution of the French Seismicity Products. *Seismological Research Letters*, 92(3):1623–1641, March 2021. doi:10.1785/0220200353.

[J4] C. Duverger, G. Mazet-Roux, L. Bollinger, A. Guilhem Trilla, A. Vallage, B. Hernandez, and Y. Cansi. A decade of seismicity in metropolitan France (2010–2019): the CEA/LDG methodologies and observations. *BSGF - Earth Sciences Bulletin*, 192:25, April 2021. doi:10.1051/bsgf/2021014.

[J5] E. Oral and C. Satriano. Future magnitude 7.5 earthquake offshore Martinique: spotlight on the main source features controlling ground motion prediction. *Geophysical Journal International*, 227(2):1076–1093, June 2021. doi:10.1093/gji/ggab245.

[J6] M. Craiu, A. Craiu, M. Mihai, and A. Marmureanu. An automatic procedure for earthquake analysis using real-time data. *Acta Geodaetica et Geophysica*, 58(1):1–18, February 2023. doi:10.1007/s40328-023-00402-1.

[J7] V. Lefils, A. Rigo, and E. Sokos. MADAM: A temporary seismological survey experiment in Aetolia-Akarnanian region (Western Greece). *Bulletin of the Geological Society of Greece*, 59(1):158–174, January 2023. doi:10.12681/bgsg.31714.

[J8] O. Lengliné, J. Schmittbuhl, K. Drif, S. Lambotte, M. Grunberg, J. Kinscher, C. Sira, A. Schlupp, M. Schaming, H. Jund, and F. Masson. The largest induced earthquakes during the GEOVEN deep geothermal project, Strasbourg, 2018–2022: from source parameters to intensity maps. *Geophysical Journal International*, 234(3):2445–2457, April 2023. doi:10.1093/gji/ggad255.

[J9] S. Panebianco, V. Serlenga, C. Satriano, F. Cavalcante, and T. A. Stabile. Semi-automated template matching and machine-learning based analysis of the August 2020 Castelsaraceno microearthquake sequence (southern Italy). *Geomatics, Natural Hazards and Risk*, May 2023. doi:10.1080/19475705.2023.2207715.

[J10] T. C. Sunilkumar, V. K. Gahalaut, D. Srinagesh, and B. Naresh. Seismotectonic significance of the December 1, 2020 Haridwar, India earthquake (M 4.3), a lower crust event near the Himalayan topographic front. *Journal of Earth System Science*, March 2023. doi:10.1007/s12040-023-02072-7.

[J11] Z. Wang, R. Liu, and W. Liu. Source characteristics of the aftershocks of the Wenchuan and Lushan earthquake sequences in the Longmen-Shan fault zone. *Frontiers in Earth Science*, January 2023. doi:10.3389/feart.2022.1061754.

[J12] P. Gehl, P. Dominique, H. Aochi, M. Delatre, J. Kinscher, and I. Contrucci. Development of an empirical ground-motion model for post-mining induced seismicity near Gardanne, France. *Journal of Sustainable Mining*, 23(2):98–117, February 2024. doi:10.46873/2300-3960.1408.

[J13] K. Drif, O. Lengliné, J. Kinscher, and J. Schmittbuhl. Induced Seismicity Controlled by Injected Hydraulic Energy: The Case Study of the EGS Soultz-Sous-Forêts Site. *Journal of Geophysical Research: Solid Earth*, May 2024. doi:10.1029/2023jb028190.

[J14] O. Boulahia and F. Semmane. Directivity analysis and source parameter estimation: insights from the 2014 Arzew earthquake, Mw3.9, northwestern Algeria. *Journal of Seismology*, 28(4):951–972, June 2024. doi:10.1007/s10950-024-10226-3.

[J15] A. F. Peña Castro, B. Schmandt, J. Nakai, R. C. Aster, and J. Chaput. (Re)Discovering the Seismicity of Antarctica: A New Seismic Catalog for the Southernmost Continent. *Seismological Research Letters*, 96(1):576–594, July 2024. doi:10.1785/0220240076.

[J16] Q. Rodríguez-Pérez and F. R. Zúñiga. Statistical and source characterization of the 2023 Kahramanmaraş Türkiye earthquake sequence. *Acta Geophysica*, 73(2):1241–1260, November 2024. doi:10.1007/s11600-024-01428-x.

[J17] P. M. Shearer, I. Vandevert, W. Fan, R. E. Abercrombie, D. Bindi, G. Calderoni, X. Chen, W. Ellsworth, R. Harrington, Y. Huang, T. Knudson, M. Roßbach, C. Satriano, M. Supino, D. T. Trugman, H. Yang, and J. Zhang. Earthquake Source Spectra Estimates Vary Widely for Two Ridgecrest Aftershocks Because of Differences in Attenuation Corrections. *Bulletin of the Seismological Society of America*, December 2024. doi:10.1785/0120240134.

[J18] S. Hicks, P. González, A. Lomax, A. Ferreira, R. Ramalho, N. Mitchell, G. Silveira, N. Dias, J. Fontiela, R. Fernandes, S. Custódio, M. Tsekhmistrenko, V. Mendes, A. Pimentel, R. Silva, G. Prates, W. Sturgeon, A. Marignier, F. Carrilho, R. Marques, M. Miranda, and A. Garcia. Leaky faults modulated magma ascent and seismicity during the 2022 São Jorge (Azores) volcanic unrest. *EarthArXiv*, December 2024. doi:10.31223/x5ht4v.

[J19] Z. Wang, R. Liu, T. Wang, Y. Zhang, and X. Ren. Energy Release Efficiency and Mechanisms of Mining-Induced High-Energy Earthquakes in China. *Rock Mechanics and Rock Engineering*, 58(5):5579–5593, February 2025. doi:10.1007/s00603-025-04386-y.

[J20] M. Alloncle, M. Bonnin, and A. Mocquet. Earthquake Source Spectra and Site Attenuation in Northwestern France. *Bulletin of the Seismological Society of America*, 115(3):965–982, April 2025. doi:10.1785/0120240122.

[J21] R. E. Abercrombie, A. Baltay, S. Chu, T. Taira, D. Bindi, O. S. Boyd, X. Chen, E. S. Cochran, E. Devin, D. Dreger, W. Ellsworth, W. Fan, R. M. Harrington, Y. Huang, K. B. Kemna, M. Liu, A. Oth, G. A. Parker, C. Pennington, M. Picozzi, C. J. Ruhl, P. Shearer, D. Spallarossa, D. Trugman, I. Vandevert, Q. Wu, C. Yoon, E. Yu, G. C. Beroza, T. Eulenfeld, T. Knudson, K. Mayeda, P. Morasca, J. S. Neely, J. Roman-Nieves, C. Satriano, M. Supino, W. R. Walter, R. Archuleta, G. M. Atkinson, G. Calderoni, C. Ji, H. Yang, and J. Zhang. Overview of the SCEC/USGS Community Stress Drop Validation Study Using the 2019 Ridgecrest Earthquake Sequence. *Bulletin of the Seismological Society of America*, May 2025. doi:10.1785/0120240158.

[J22] R. Cabieces, T. C. Junqueira, K. Harris, J. Relinque, C. Satriano, and J. Vackář. surfQuake: A New Python Toolbox for the Workflow Process of Seismic Sources. *Seismological Research Letters*, May 2025. doi:10.1785/0220240186.

[J23] L. Moratto, S. Panebianco, C. Satriano, T. A. Stabile, and E. Priolo. Using ambient noise k0 estimation to improve microearthquake source parameters assessment in the High Agri Valley. *Geophysical Journal International*, May 2025. doi:10.1093/gji/ggaf191.

[J24] L. Gailler, C. Grace, G. Boudoire, M. Grunberg, J. Battaglia, C. Merciecca, P. Labazuy, T. Souriot, J. Douchain, N. Cluzel, R. Guillard, V. Fréret Lorgeril, C. Aumar, V. Rafflin, V. Boulenger, S. Buvat, J. Kuzamba, and E. Thébault. Scientific response to the 2021–2022 seismic swarm in the Monts Dore volcanic province (France): structural insights from punctual surveys (1/2). *Comptes Rendus. Géoscience*, 357(G1):61–78, May 2025. doi:10.5802/crgeos.284.

[J25] J. Liu, J. Zahradník, V. Plicka, F. Gallovič, C. R. Bina, and H. Čížková. Deep-Focus Earthquakes Under Northeast China—An Imprint of the Complex Tectonic History of Pacific Plate Subduction. *Journal of Geophysical Research: Solid Earth*, September 2025. doi:10.1029/2024jb030215.

[J26] J. Jacob, J. Dyment, D. Ghosal, C. Satriano, P. Dewangan, and H. K. Haridas. Stresses and seismicity along the Andaman-Sumatra-Java subduction zone: impact of subducting plate heterogeneities. *Journal of Asian Earth Sciences*, 305:107073, June 2026. doi:10.1016/j.jseaes.2026.107073.

[C1] A. A. S. Putra, B. A. D. Nugraha, C. N. T. Puspito, and D. D. P. Sahara. Preliminary result: source parameters for small-moderate earthquakes in Aceh segment, Sumatran fault zone (Northern Sumatra). In *18th Annual Meeting of the Asia Oceania Geosciences Society*. WORLD SCIENTIFIC, April 2022. doi:10.1142/9789811260100_0076.

[C2] G. Ganzorig Davaasuren and O. Monkhor. Scaling relations of moment magnitude (Mw) and local magnitude (ML) for large earthquakes in northern Mongolia. 2023. doi:10.13140/RG.2.2.15069.59362.

[C3] A. Chouli, L. Costes, D. Marsan, J. Münchmeyer, S. Giffard-Roisin, and A. Socquet. Search for repeaters in the central part of the Chilean subduction zone. March 2024. doi:10.5194/egusphere-egu24-17042.

[C4] M. Grunberg and S. Lambotte. A new workflow for revising the seismicity catalog for mainland France, covering the period 2010-2018. March 2024. doi:10.5194/egusphere-egu24-5100.

[C5] R. Cabieces, T. C. Junqueira, and J. Relinque. SurfQuake (SQ): A new Python toolbox for the workflow process of seismic sources. March 2024. doi:10.5194/egusphere-egu24-2816.

[C6] J. Corbeau, OL. Gonzalez, C. Satriano, J.-M. Saurel, M.-P. Bouin, and A. Lemarchand. Relationship between magnitudes commonly calculated by the French observatories and the moment magnitude Mw, and variability of stress drop in the Lesser Antilles subduction zone. March 2026. doi:10.5194/egusphere-egu26-15337.

PYTHON MODULE INDEX

INDEX

H

I

K

L

M

O

P

Q

R

S

T

U

V

W