magsdetector

Event detector based on waveform cross-correlation.

Description

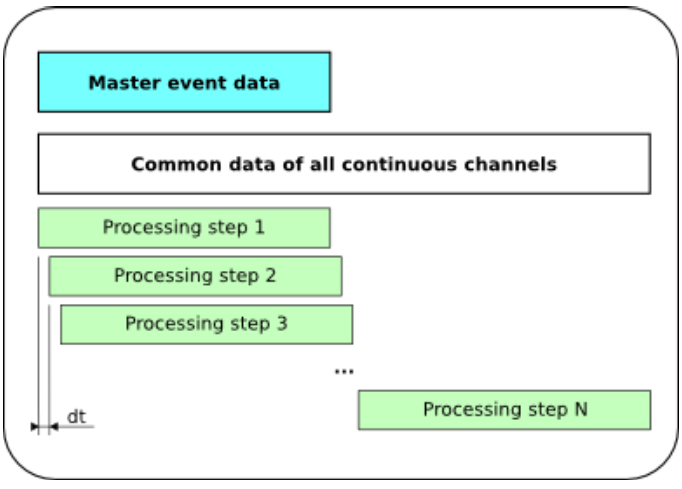
magsdetector detects events by finding similarities in defined master events based on cross-correlation of waveforms or envelopes. An event (origin) is declared if the waveform fit is above a configured threshold. Location and depth are taken from the master events whereas time is taken from the time window currently processed and magnitude is computed from the ratio of the master event PGV to the continuous PGV.

$$M = M_{master} + \frac{1}{N}\sum_{i=1}^{N} \ln \frac{PGV_{continuous_i}}{PGV_{master_i}} \quad (1)$$

magsdetector can be ran in realtime or offline.

Workflow

magsdetector subscribes to all configured **channels**. On each received record the common time window of all channels is computed and if sufficent data are available they are processed.



Processing of a data time window of contiuous channels. The step of each processing step is the time of one sample. See Processing for how each time window is processed.

Each processed time window is *matched* with the master event. If the overall fit exceeds the **configured threshold** the detector is started to search for the maximum fit within the **configured timewindow**. The timestamp of the maximum fit is taken as event time and an origin with a magnitude is sent to the messaging.

Processing

The data are processed according to the configuration. Both schemas are illustrated below.

Processing schema in either frequency domain and time domain. ***data*** *are the input data received from records and sensitivity corrected.* ***data*** *are also the outputs used by subsequent steps.*

The processing is applied to the data of the master events as well as to the continuous data. Processing in the frequency domain is usually more accurate and does not introduce phase delays as with recursive filters. On the other hand time domain processing is much faster.

Envelope

The cross-correlation can be done on the seismograms itself or on the envelopes of the seismograms. We compute the envelope using the Hilbert-transform in frequency domain:

$$\hat{y}_i^j = \sqrt{(y_i^j)^2 + (H\{y_i^j\})^2} \tag{2}$$

or a running root mean square in time domain:

$$\hat{y}_i^j = \sqrt{\frac{2}{N} \sum_{k=i-N}^{i} (y_k^j)^2} \tag{3}$$

Here y_i^j is the output signal of the single trace operation, which in this case is the envelope of the signal. y_i is the input signal of the single trace operation, which in this case is the filtered seismogram. i and j are counters for time and trace number, respectively. H(y) is the Hilbert-transform of function y. N is the number of samples used for the RMS which is calculated as:

$$N = \frac{\text{sampling frequency}}{\text{envelope.hiFreq}}$$

Using the envelope of seismograms instead of the seismograms itself, the cross-correlation becomes less sensitive to small changes in the source location and in source mechanism between master signal and the earthquake to be detected.

Logarithm of trace

As an option we can compute the logarithm of a trace:

$$y_i^j = sgn(y_i^j)\, ln|y_i^j| \tag{4}$$

Application of this single trace operation increases the importance of small amplitudes in the cross-correlation. Therefore, it will gen- erally amplify the noise level, which is an unwanted effect. On the other hand using this option we can increase the importance of small amplitude waves like coda waves in comparison to direct P- and S-waves or we can increase the importance of recordings at distant stations in comparison to close station (if we compute the network cross-correlation).

Noise removal

If we use seismogram envelopes, the constant background noise level causes a problem in the computation of the cross-correlation. Even if we remove the mean value in the seismograms, a constant offset appears in the envelopes. This offset value in the envelopes corresponds to the standard deviation of noise in the original seismograms. If we correlate a constant (noise) trace with the master event, a large correlation coefficient results possibly causing wrong detections. A simple solution to this problem is to remove the noise level from the master event traces as well as from the current time window traces. We estimate the noise level y^j in a time window j before the potential signal at envelope trace y_i^j:

$$\overline{y^j} = \frac{1}{N_n} \sqrt{\sum_{i=1}^{N_n} y_i^j} \tag{5}$$

Here i is the counter for time samples, N_n is the number of time samples in the time window to estimate the noise level, and j is a counter for the traces (seismometers times components). Then we remove the noise level, which is a constant offset in the envelopes:

$$\hat{y}_i^j = y_i^j - \overline{y^j} \tag{6}$$

Here, the time window, where we remove the offest (and where we later compute the

correlation) is generally different from the time window, where we estimate the noise level.

Cross-correlation

Single trace

In a first step we compute the single trace cross-correlation with zero lag at channel j:

$$R^j = \frac{1}{\sqrt{\left(\sum_{i=1}^{N} x_i^j x_i^j\right)\left(\sum_{i=1}^{N} y_i^j y_i^j\right)}} \sum_{i=1}^{N} x_i^j y_i^j \tag{7}$$

Here i is the counter for time samples, N is the number of time samples in the used correlation time window, x^j is the filtered seismogram or envelope of the master event after single trace operations described in *Processing*, y^j is the filtered seismogram or envelope of the current time window after applying the same operations as to the master traces, and j is a counter for the traces (seismometers times components).

This cross-correlation coefficient is computed for all traces j=1 to M, where M/3 is the total number of seismometers of the master event, if we use three-component sensors. The first condition for a detection is, that

$$R^j > \text{detector.channelThreshold, for all } j = 1 \text{ to } M_{min} \leq M \tag{8}$$

where R^j is defined in equation (7), R_1 is a configurable input threshold to be chosen in the order of $R_1 \approx 0.6$ - 0.8, and M_{min} is the minimum number of seismograms, where the earthquake is detectable.

In other words: This condition requires that the current time window and the master event shows similar waveforms ($R^j > R_1$) for at least M_{min} seismograms. We use the parameter M_{min} to account for the fact that some seismograms may show large local noise ($R^j < R_1$), when a small earthquake occurs.

M_{min} is also useful, when data transmission of some sensors is interrupted. In that case the correlation coefficient is set to zero $R^j = 0 < R_1$ and the result is similar to a local noise disturbance. A detection is only possible, if the number of noisy or otherwise disturbed traces does not exceed M - M_{min}.

M_{min} is implicitely given through **detector.minimumChannelRatio**.

Network

In a next step we compute the network cross-correlation with zero lag, where two options, trace (A) and total (B), are available. What option is used is configurable with **processing.normalization**.

Note

The network cross-correlation is only computed if the ratio of time window traces and master event traces does not fall below **detector.minimumChannelRatio** and if the ratio of time window stations and master event stations does not fall below **detector.minimumStationRatio**. Otherwise the network cross-correlation is set to 0.

In option A we compute the average single trace cross-correlation coefficient of the M_{min} traces with the highest single trace cross-correlation coefficient:

$$R_{trace} = \frac{1}{M_{min}} \sum_{j=1}^{M_{min}} R^j = \frac{1}{M_{min}} \sum_{j=1}^{M_{min}} \frac{1}{\sqrt{(\sum_{i=1}^{N} x_i^j x_i^j)\,(\sum_{i=1}^{N} y_i^j y_i^j)}} \sum_{i=1}^{N} x_i^j y_i^j \tag{9}$$

In option B we compute the matrix cross-correlation with zero lag, where one dimension of the matrix is time and the other dimension are the different traces:

$$R_{total} = \frac{1}{\sqrt{(\sum_{j=1}^{M_{min}} \sum_{i=1}^{N} x_i^j x_i^j)(\sum_{j=1}^{M_{min}} \sum_{i=1}^{N} y_i^j y_i^j)}} \sum_{j=1}^{M_{min}} \sum_{i=1}^{N} x_i^j y_i^j \tag{10}$$

The major difference between R_{trace} and R_{total} is that in equation (9) the relative amplitudes between stations and components are neglected, whereas they are taken into account in equation (10).

Then, the second and third condition for a detection are:

$$R_{trace} > \text{detector.threshold} \tag{11}$$

$$R_{total} > \text{detector.threshold} \tag{12}$$

Here **`detector.threshold`** is a configurable threshold, which judges the network similarity and should be set to about detector.threshold ≈ 0.6 - 0.8.

Examples

1. Running magsdetector offline with a multiplexed miniseed volume and an inventory xml file. Neither a messaging nor a database connection is required.

```
magsdetector --inventory-db inventory.xml -I test-sorted.mseed --offline
```

Configuration

`etc/defaults/global.cfg`
`etc/defaults/magsdetector.cfg`
`etc/global.cfg`
`etc/magsdetector.cfg`
`~/.seiscomp3/global.cfg`
`~/.seiscomp3/magsdetector.cfg`

magsdetector inherits *global options*.

`channels`

Type: *list:string*

Defines the data channels to be used as list of stream codes. Using full channel codes (3 characters) will use only this channel, defining 2 characters will make use of all 3 components, e.g. AB.STA01..HHZ vs. AB.STA01..HH

`events`

Type: *list:string*

Defines the master events. It is a list of event identifiers.

filter.order

Type: *int*

Specifies the filter order. Can be overridden per master event. Default is `4`.

filter.loFreq

Type: *double*

Specifies the frequency of the general hi-pass filter. If this parameter is equal to 0 the hi-pass filter is not used. Can be overridden per master event. Default is `10`.

filter.hiFreq

Type: *double*

Specifies the frequency of the general lo-pass filter. If this parameter is equal to 0 the lo-pass filter is not used. Can be overridden per master event. Default is `40`.

filter.bandStop

Type: *boolean*

Enables a bandstop filter at 50Hz and 100Hz. Can be overridden per master event. Default is `false`.

envelope.enable

Type: *boolean*

Compute envelope to be used for fitting the master events. Can be overridden per master event. Default is `true`.

envelope.samplingFrequency

Type: *int*

Defines the target sampling frequency of the envelope (0 = no resampling). Can be overridden per master event. Default is `0`.

envelope.resampleAverage

Type: *boolean*

Enables averaging of neighbor samples when downsampling. The width of the kernel is the ratio of SR/samplingFrequency. Can be overridden per master event. Default is `false`.

envelope.hiFreq

Type: *double*

Frequency of lowpass for smoothing the envelope (0 = no filter). Can be overridden per master event. Default is `20`.

envelope.acausal

Type: *boolean*

Defines if the envelope function is done in the frequency domain with hilbert transformation (true) or recursive (false) Can be overridden per master event. Default is `false`.

processing.acausal

Type: *boolean*

Defines whether the waveform processing (filtering, envelope, ...) is done in the frequency domain (true) or recursive in the time domain (false). Processing in time domain is much faster but less accurate. Can be overridden per master event. Default is `false`.

processing.logarithm

Type: *boolean*

Enables usage of the logarithm of the final trace (either waveforms or envelope). Can be overridden per master event. Default is `false`.

processing.bufferSize

Type: *int*

Defines the data buffer size in seconds (integer values). The default is 10 minutes (600 seconds). This parameter is important when testing with record volumes where records are not ordered by time. Default is `600`.

processing.interval

Type: *int*

Defines the processing interval in seconds (integer values). If records are received it will process the data every n seconds not more often. 0 disables the interval and starts processing whenever a new record is received and enough data are available. Default is `0`.

processing.maximumLatency

Type: *double*

Defines the maximum data latency in seconds tolerated by the detector. If data latency is higher than this value processing is triggered without this particular channel. Default is `10`.

processing.normalization

Type: *string*

Flag for normalization: trace, total. Default is `total`.

processing.maximumStepFrequency

Type: *int*

The maximum frequency of processing steps per time window. If the data sampling frequency is much higher this parameter reduces processing time by increasing the time steps. A value of zero derives the time steps from the datas or envelopes sampling frequency. Default is `0`.

detector.threshold

Type: *double*

Detector threshold to search for maximum to declare an event. Default is `0.55`.

detector.channelThreshold

Type: *double*

Detectors minimum channel fit to be used for matching with a master event. Default is `0.55`.

detector.window

Type: *double*

Length of time window (in s) to search for maximum after trigger. Default is `2`.

detector.minimumStationRatio

Type: *int*

Minimum ratio in percent of matches stations vs. stations in master event.

detector.minimumChannelRatio

Type: *int*

Minimum ratio in percent of matches overall channels vs. channels in master event.

detector.minimumProcessingWindow

Type: *double*

Sets the minimum processing time window length in seconds to minimize processing steps on each record arrival. 0 processes the data as fast as possible. Default is `0`.

detector.publicationTimeout

Type: *int*

If event groups are used and an event is declared which still needs to wait for all other events in the group to be processed this timeout is used to publish the best event of this group after a certain amount of time regardless of unprocessed events in the group. A negative value disables this feature and a queued event never times out. Default is `10`.

output.waveforms.enable

Type: *boolean*

Defines if each processed event is dumped to MiniSEED for e.g. debugging reasons. Also all data that are processed after the trigger is reached and withing the processing time window (detector.window) is dumped. Default is `false`.

output.waveforms.mseed

Type: *boolean*

Defines if each time step after a trigger within detector.window is saved as multiplexed MiniSEED file which contains all channels. Default is `false`.

output.waveforms.path

Type: *string*

Defines the waveform output path. A subdirectory for each master event (by its name) is created. Default is `@LOGDIR@/magsdetector/waveforms`.

output.fit.enable

Type: *boolean*

Defines if the fit function for an event and each channel is dumped to output.waveforms.path. The fit function starts at the trigger and ends after trigger+detector.window. The naming convention is date-event.fit and data-[channel.id].fit. The first column contains the timestamp and the second column the fit. The channel fit functions include additionally the fit contribution to the event in the 3rd column and the weight with respect to the overall fit. Default is `false`.

output.events.file

Type: *string*

Define output file for matched events. Each match creates a new line with format yyyy mm dd

HH MM SS.FFF Lat Long Mag city CF usedPhases (cha:fit, ...). If not set no event file is created.

output.fit.enable

Type: *boolean*

Defines if the fit function for an event and each channel is dumped to output.waveforms.path. The fit function starts at the trigger and ends after trigger+detector.window. The naming convention is date-event.fit and data-[channel.id].fit. The first column contains the timestamp and the second column the fit. The channel fit functions include additionally the fit contribution to the event in the 3rd column and the weight with respect to the overall fit. Default is `false`.

Note

event.$name.* *Defines a master event. Processing options such as filter and envelope parameters can be overridden otherwise the global values are used.* $name is a placeholder for the name to be used and needs to be added to **events** to become active.

```
events = a,b
event.a.value1 = ...
event.b.value1 = ...
# c is not active because it has not been added
# to the list of events
event.c.value1 = ...
```

event.$name.negative

Type: *boolean*

If enabled this event is a negative event. Negative event detections don't create an origin. They can be used to stop false alarms. Default is `false`.

event.$name.baseID

Type: *string*

Defines the database eventID to use. This will extract the origin time and the magnitude. Furthermore arrivals and picks are collected which are copied to the final origin. Using this parameter requires a database connection.

event.$name.xml

Type: *string*

Defines an event parameters XML which is used to extract event information: origin time and magnitude. It and makes this parameters optional. Furthermore arrivals and picks are collected which are copied to the final origin.

event.$name.time

Type: *string*

Defines the start time of the event in format "%F %T.%f". This time is used as start time to extract the data. If either baseID or xml is given this time must match the extracted origin time otherwise an error is reported. When defining either baseID or xml this parameter is not required but can be used for further validation.

event.$name.noiseBegin

Type: *double*

Defines the begin of the noise window if envelope is enabled. The mean is computed from the noise window and removed from the whole envelope. This value is relativ to time. If not given

then noiseBegin is set to signalBegin. The minimum mean of both defined noise windows is used finally.

event.$name.noiseEnd

Type: *double*

Defines the end of the noise window if envelope is enabled. The mean is computed from the noise window and removed from the whole envelope. This value is relativ to time. If not given then noiseEnd is set to signalBegin + 1. The minimum mean of both defined noise windows is used finally.

event.$name.noise2Begin

Type: *double*

Defines the begin of the second noise window if envelope is enabled. This value is relativ to time. If not given then noise2Begin is set to signalBegin. The minimum mean of both defined noise windows is used finally. This parameter is optional and set to noiseBegin if not given.

event.$name.noise2End

Type: *double*

Defines the end of the second noise window if envelope is enabled. This value is relativ to time. If not given then noiseEnd is set to signalBegin + 1. The minimum mean of both defined noise windows is used finally. This parameter is optional and set to noiseEnd if not given.

event.$name.signalBegin

Type: *double*

Defines the begin of the signal window used for cross-correlation.

event.$name.signalEnd

Type: *double*

Defines the end of the signal window used for cross-correlation.

event.$name.place

Type: *string*

A string of the place of the event. Is used when adding a line to the event file.

event.$name.group

Type: *string*

Defines a group name. Only one event of a group can occur at one time. If the group is empty the event is independent and forms a group containing only this event.

event.$name.magnitude

Type: *double*

Defines the reference magnitude which is used to compute the magnitude of the continuous data. When defining either baseID or xml this parameter is not required but can be used to override the extracted magnitude.

event.$name.deltaM

Type: *double*

Defines a magnitude offset which is applied to the final magnitude. This parameter is optional.

event.$name.latitude

Type: *double*

Defines the latitude of the master event which is taken as it is if a match is found.

event.$name.longitude

Type: *double*

Defines the longitude of the master event which is taken as it is if a match is found.

event.$name.depth

Type: *double*

Defines the depth of the master event which is taken as it is if a match is found.

event.$name.data

Type: *string*

Defines the data URI for fetching the waveforms. The format is equal to the --record-url parameter.

event.$name.filter.order

Type: *int*

Specifies the filter order. If not given the global settings are used.

event.$name.filter.loFreq

Type: *double*

Specifies the frequency of the general hi-pass filter. If this parameter is equal to 0 the hi-pass filter is not used. If not given the global settings are used.

event.$name.filter.hiFreq

Type: *double*

Specifies the frequency of the general lo-pass filter. If this parameter is equal to 0 the lo-pass filter is not used. If not given the global settings are used.

event.$name.filter.bandStop

Type: *boolean*

Enables a bandstop filter at 50Hz and 100Hz. If not given the global settings are used.

event.$name.envelope.enable

Type: *boolean*

Compute envelope to be used for fitting the master events. If not given the global settings are used.

event.$name.envelope.samplingFrequency

Type: *int*

Defines the target sampling frequency of the envelope (0 = no resampling). If not given the global settings are used.

event.$name.envelope.resampleAverage

Type: *boolean*

Enables averaging of neighbor samples when downsampling. The width of the kernel is the ratio of SR/samplingFrequency. If not given the global settings are used. Default is `false`.

event.$name.envelope.hiFreq

Type: *double*

Frequency of lowpass for smoothing the envelope (0 = no filter). If not given the global settings are used.

event.$name.envelope.acausal

Type: *boolean*

Defines if the envelope function is done in the frequency domain with hilbert transformation (true) or recursive (false) If not given the global settings are used.

event.$name.processing.enable

Type: *boolean*

Enables or disables processing of master event data. If disabled the data is used as provided. Incoming data are processed in any way. If disabled the provided data must have been processed the same way as the current configuration. Default is `true`.

event.$name.processing.acausal

Type: *boolean*

Defines whether the waveform processing (filtering, envelope, ...) is done in the frequency domain (true) or recursive in the time domain (false). Processing in time domain is much faster but less accurate. If not given the global settings are used.

event.$name.processing.logarithm

Type: *boolean*

Enables usage of the logarithm of the final trace (either waveforms or envelope). If not given the global settings are used.

Command-line

Generic

-h, **--help**

show help message.

-V, **--version**

show version information

--config-file `arg`

Use alternative configuration file. When this option is used the loading of all stages is disabled. Only the given configuration file is parsed and used. To use another name for the configuration create a symbolic link of the application or copy it, eg scautopick -> scautopick2.

--plugins `arg`

Load given plugins.

-D, **--daemon**

Run as daemon. This means the application will fork itself and doesn't need to be started with &.

--auto-shutdown arg

Enable/disable self-shutdown because a master module shutdown. This only works when messaging is enabled and the master module sends a shutdown message (enabled with --start-stop-msg for the master module).

--shutdown-master-module arg

Sets the name of the master-module used for auto-shutdown. This is the application name of the module actually started. If symlinks are used then it is the name of the symlinked application.

--shutdown-master-username arg

Sets the name of the master-username of the messaging used for auto-shutdown. If "shutdown-master-module" is given as well this parameter is ignored.

Verbosity

--verbosity arg

Verbosity level [0..4]. 0:quiet, 1:error, 2:warning, 3:info, 4:debug

-v, **--v**

Increase verbosity level (may be repeated, eg. -vv)

-q, **--quiet**

Quiet mode: no logging output

--component arg

Limits the logging to a certain component. This option can be given more than once.

-s, **--syslog**

Use syslog logging back end. The output usually goes to /var/lib/messages.

-l, **--lockfile** arg

Path to lock file.

--console arg

Send log output to stdout.

--debug

Debug mode: --verbosity=4 --console=1

--log-file arg

Use alternative log file.

Messaging

-u, **--user** arg

Overrides configuration parameter **connection.username**.

-H, **--host** arg
: Overrides configuration parameter **connection.server**.

-t, **--timeout** arg
: Overrides configuration parameter **connection.timeout**.

-g, **--primary-group** arg
: Overrides configuration parameter **connection.primaryGroup**.

-S, **--subscribe-group** arg
: A group to subscribe to. This option can be given more than once.

--encoding arg
: Overrides configuration parameter **connection.encoding**.

--start-stop-msg arg
: Sets sending of a start- and a stop message.

Database

--db-driver-list
: List all supported database drivers.

-d, **--database** arg
: The database connection string, format: service://user:pwd@host/database. "service" is the name of the database driver which can be queried with "--db-driver-list".

--config-module arg
: The configmodule to use.

--inventory-db arg
: Load the inventory from the given database or file, format: [service://]location

--db-disable
: Do not use the database at all

Records

--record-driver-list
: List all supported record stream drivers

-I, **--record-url** arg
: The recordstream source URL, format: [service://]location[#type]. "service" is the name of the recordstream driver which can be queried with "--record-driver-list". If "service" is not given "file://" is used.

--record-file arg
: Specify a file as record source.

--record-type arg
: Specify a type for the records being read.

Mode

--offline

Do not connect to a messaging system.

--test

Runs the detector without sending any objects to the messaging system.

--dump-fit

Dumps the fit function of each master event processed continuously. Useful for testing. Both the overall fit and the channel fits are dumped in separate files. The overall fit goes into [event.name].fit whereas the channel fits go into [event.name]-[net.sta.loc.cha].fit. The overall fit plot file is using 3 columns: counter, value, timestamp. The channel fit plot file is using 4 columns: counter, single trace fit, trace fit contribution to overall fit, timestamp.