Technical report for internal use

A seismological study of the Sos Enattos area, the Sardinia candidate site for the Einstein Telescope

Introduction

The aim of this report is to deliver the technical details about the softwares and the methods used in the analyses for the paper *A seismological study of the Sos Enattos area, the Sardinia candidate site for the Einstein Telescope* by the writing team (Di Giovanni M., Giunchi C., Saccorotti G., Berbellini A., Olivieri M., Boschi L.). The main goal is to facilitate the repeatability of the analyses and to help defining the standard tools and methods to be used in future works about Sos Enattos. After a brief overview of the data and resources used for the paper, I will discuss the methods and softwares used section by section.

Data, softwares and resources

Data management, analysis and visualization have been carried out mainly in Python using Obspy, Numpy and Matplotlib. Obspy is an open source project dedicated to provide a Python framework especially dedicated to the processing of seismological data. Obspy can be downloaded and installed using the instructions in github.com/obspy/obspy/wiki. Seismic velocity data have been processed using GEOPSY, another open source project for geophysical research (www.geopsy.org). The seismic noise amplitude simulations were carried out using the *Computer Programs in Seismology* package by Hermann (http://www.eas.slu.edu/eqc/eqccps.html) available in C and FORTRAN. Data about the average wave height of the Tyrrhenian Sea are available on the Copernicus Marine Environment Service at https://www.copernicus.eu/. The localization of local earthquakes was provided by NonLinLoc, a package for velocity model construction, travel-time calculation and probabilistic earthquake location (http://alomax.free.fr/nlloc/).

Sos Enattos seismic data are stored in local servers. One of the seismic stations has been recently added to the Italian seismological monitoring network as IV.SENA. Its data are publicly available and can be accessed through the International Federation of Digital Seismograph Networks (FDSN) using the following parameters:

- `Client = INGV`
- `Network = IV`
- `Station = SENA`

Three more stations, reachable through FDSN as well, were in used for earthquake location: IV.DGI, IV.AGL and MN.VSL.

- `Client = INGV`
- `Network = IV/IV/MN`

- `Station = DGI/AGL/VSL`

The notebooks with the procedures used for this paper are stored in the ET Repository under `/home/mdigiovanni/materiale_paper`.

Paper sections and methods

Instruments and noise environments

All the analyses in this section were carried out using Obspy functions. The probabilistic power spectral densities (PPSD, Figure 2) were calculated using the method of McNamara and Buland (2004; DOI: 10.1785/012003001). The functions delivering the PPSD are the following:

```
> st = client.get_waveforms(station=sta, channel=cha, network=net,
location='', starttime=start, endtime=stop)
> tr = st.select(channel="HHZ")[0]
> soe0 = PPSD(tr.stats, metadata=inv, skip_on_gaps=True)
> soe0.add(st)
> soe0.plot(cmap=pqlx, show_coverage=False, xaxis_frequency=True,
period_lim=(0.004,10))
```

This method returns PPSD calculated on 1 hour long segments with 50% overlap over the `start-stop` time interval. For the meaning and usage of all the parameters and functions, see Obspy documentation (https://docs.obspy.org).

The day/night medians of Figure 3b are obtained calculating the PPSD separately during daytime (08:00 - 17:00) and nighttime (21:00 - 04:00). The median was calculated extracting the PPSD segments using the entry `soe0._binned_psds` and using the standard Numpy function `np.percentile(array_to_percentiles, [50], axis = 0)`. Frequencies are included in the entry `1/soe0._period_binning[2]`.

The medians of Figure 3a were calculated using the `pwelch` function on MatLab on 1 hour long segments with 50% overlap. Other parameters used are:

- `window = turkey` with `R = 0.2`
- `noverlap = 49152`
- `nfft = 65536`
- `fs = 100`

The spectra were then converted into acceleration by multiplying for omega^2 and dividing by the square modulus of the instrument response. After that, the spectra were averaged using the method of McNamara and Buland (2004), with the difference that the average was on 1/2 octave in 1/16 octave intervals. The median was finally calculated on all the spectra.

Seismic velocities

The Rayleigh-wave group velocity was estimated using a frequency-time analysis (FTAN). The envelopes of the signals at individual stations were calculated using a Gaussian filter with alpha=10 spanning 25 center frequencies between 3 Hz and 8 Hz. Group velocity estimates for each pair of stations were derived using the time differences between the maxima of the envelopes at the different stations. The phase-velocity dispersion curve was calculated using a p-omega procedure.

We assumed that both group and phase velocity dispersions were associated with the fundamental mode of Rayleigh waves. Using this assumption, we performed a joint inversion of those curves for the ground profile of V_s. The inversion was done using the GEOPSY software and using a parametrization of the subspace consisting of two 100 m thick layers overlying the half space. Density was kept constant at 2600 kg/m^3.

Seismic noise amplitude ratio

Seismic noise amplitude was simulated using the routines available in *Computer Programs in Seismology* by Hermann. The routines were used in the following order:

```
> sprep96
> sdisp96
> slegn96
> sregn96
> spulse96 | fmech96
```

The focal mechanism was set using all zeros as inputs, except for `Mrr = 1e20`.

Horizontal to vertical spectral ratios

The HVSR was calculated using the procedure of Atakan *et al.* (2004):

- reading the input time series and offset removal;
- tapering;
- FFT;
- smoothing of Fourier spectra using Konno-Ohmachi smoothing;
- merging of horizontal components;
- HV ratio.

The FFT is calculated on 1 hour long segments with a 50% overlap using the Numpy function `np.fft.rfft()` over the desired time interval. The tapering is performed using a Hanning window with `np.hanning()`. After the FFT, the Konno-Ohmachi smoothing was done using a pre-built function in `obspy.signal` called `konnoohmachismoothing.konno_ohmachi_smoothing(real_part_of_FFT_to_smooth, array_of_frequencies)`. Additional parameters used in the smoothing function are:

- `Bandwidth = 20`
- `Count = 2`

- `Normalize = True`

After the smoothing, the horizontal components are merged with a geometrical mean `H_comb = np.sqrt(H_1*H_2)`. The HV ratio is thus obtained by making the ratio of `H_comb` with the vertical component.

Earthquake detection

In this section, we show correlation between the seismic stations. Correlation was calculated using a simple and straightforward procedure:

- extraction of 10 minutes long data segments;
- apply bandpass filter between 1 Hz and 10 Hz;
- correlate the segments.

Correlation was calculated using `cc = obspy.signal.cross_correlation.correlate(trace_1, trace_2, 300)` where 300 is the max shift applied to correlate the two traces. The function `obspy.signal.cross_correlation.xcorr_max(cc)` then outputs the desired max correlation value for each segment.

Sea waves correlation

Data for average sea wave height are available through the CMEMS service here: https://resources.marine.copernicus.eu/?option=com_csw&view=details&product_id=MEDSEA_ANALYSIS_FORECAST_WAV_006_017. After selecting the coordinates and the time period of interest, the data are downloaded in a .nc format file. This format can be handled in Python using the `NetCDF4` library. In particular, using:

```
>   from netCDF4 import Dataset
>   sea_data = Dataset('path_to_file')
```

The variables of interest to extract from the dataset are:

```
>   time = sea_data.variables['time'][:]
>   onda = sea_data.variables['VHM0'][:]
>   longitude = sea_data.variables['longitude'][:]
>   latitude = sea_data.variables['latitude'][:]
```

The selected geographical region is sampled with a coordinate grid resolution of 4.6 km. To every point in the coordinate grid is associated a time serie with the average wave height data.

At the same time, to correlate the PPSD of *Instruments and noise environment* with the time serie of sea data, the former must be translated into a time serie as well. This is achieved using the Obspy function:

```
>   time_serie,g2,g3,g4 = soe0.extract_psd_values(period=per)
```

Using the provided period `per`, this function selects the period bin whose center value is the closest to `per` and extracts all the PPSD values in that bin for all the analyzed time segments, thus creating a time serie. This way, we can extract the PPSD values at different periods and find for which period the correlation is maximized over the considered geographical region. In particular, the time serie from seismic data is correlated to all the time series in the coordinate grid providing a correlation map.

Please note that, by just using the settings reported in *Instruments and noise environment*, the time serie of seismic data is sampled every 30 minutes. On the other hand, sea wave data are sampled every 60 minutes. Correlation must be performed on series with the same sampling. This can be achieved by both subsampling the seismic data time serie or by resampling the sea wave data using `scipy.interpolate.UnivariateSpline`. After the time series have been resampled, correlation is done using the same functions explained in the previous section.

Also, any gap in time in the seismic data time serie must be carefully addressed, because any missing data can affect the final result. To prevent any error, sea wave data at times missing in seismic data must be deleted. This way, correlation is done exactly on points belonging to the same times.