



Bachelor Thesis

Seismic Data Analysis with ObsPy

Department of Earth and Environmental Sciences,
Ludwig-Maximilians-University Munich

by

Nico Trebbin

First Examiner: Prof. Dr. Heiner Igel

Second Examiner: Dr. Joachim Wassermann

Date of submission: Monday, August 23, 2010

Matriculation number: 3040238

Affidavit

by Trebbin, Nico

last name

mat. number

I hereby declare that the Bachelor Thesis with the title:

Seismic Data Analysis with ObsPy

has been written only by me and without any assistance from third parties.

Furthermore, I confirm that no sources have been used in the preparation of this thesis other than those indicated in the thesis itself, and that quotations used were identified.

The thesis does not contain any work that I have handed in or have had graded as a Prüfungsleistung earlier on.

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quelle als Entlehnung kenntlich gemacht. Die Arbeit enthält kein Material, das ich bereits zu einem früheren Zeitpunkt als Prüfungsleistung eingebracht habe.

Date

Signature

Abstract

Goal of the present thesis is the automated generation of graphics for the observation of co-seismic ground motions (translations and rotations) at the seismic measurement station Wettzell, Germany. For this purpose a program was developed that is based on the programming language Python.

It uses the ObsPy toolbox developed by the Department of Earth and Environmental Sciences at the Ludwig-Maximilians-University, Munich and expands the functionalities provided by now.

Furthermore, it includes several interfaces to work with other web services and allows the seismologists of the International Working Group on Rotational Seismology to fast process the data they are interested in. Thus it provided the ability to pre-check seismic data for their relevance.

Within this thesis the reader can obtain an insight of the development of the program and gets an introduction to the operating mode and output of the program.

List of Figures

Figure 1 Schematic sketch of the operating mode	12
Figure 2 Plot of all simulated phases calculated with the ak135 model	15
Figure 3 Uncorrected BHN component of the Wettzell seismometer for Nicobar Islands event at 2010-06-16 with UTC time on the time axis and some undefined amplitude values by now.	17
Figure 4 Benchmark results from a synthetic 100.000 sample noise test.....	19
Figure 5 Computing time of calculations done during Nicobar Islands processing	20
Figure 6 Example command line output of the Nicobar Islands event.....	22
Figure 7 Unaffected seismic streams for the Nicobar Islands event	23
Figure 8 Corrected signal of the first horizontal component of the seismometer	24
Figure 9 Second horizontal component of the seismometer.....	25
Figure 10 Vertical component of the seismogram.....	25
Figure 11 Ringlaser component with fixed nm/s axis for better coherence with other channels	26
Figure 12 Separated transversal acceleration & rotation rate time series.....	27
Figure 13 Superposition of TA (red) and RR (black)	27
Figure 14 As we see here the correlation coefficient is very high since P-wave arrival. Even if the main phases arrived the coefficient stays high.	28
Figure 15 In the spectrogram of the transversal acceleration you can clearly determine the dominant frequencies from 0.01Hz to a maximum of 2Hz around the S wave arrival	29
Figure 16 The dominant frequencies of the rotation rate are nearly equal to the frequency spectrum of the transversal acceleration	29
Figure 17 Cross correlation plot of the IJR event.....	32
Figure 18 Spectrogram of the transversal acceleration component with interesting frequency-bands at 6.5Hz and 3.7Hz.....	32
Figure 19 Spectrogram of rotation rate with nothing more than the low frequency-band from 0.01Hz to 0.5Hz	33
Figure 20 No significant excess value could be retrieved by looking at this noise-like seismogram.....	34
Figure 21 Again we can see the 7Hz frequency-band and the expected tele-seismic low frequency-band	34
Figure 22 The rotation rate has a frequency range from 0.01Hz to 1Hz.....	35

List of Tables

Table 1 Simulation of arrival times with the ak135 models for all phases of the Nicobar Islands event on 2010-06-12. dT/dD describes the travel time with respect to distance, dT/dh the travel time with respect to source depth and d^2T/dD^2 the second derivative of travel time with respect to distance.	14
Table 2 Details of the benchmarked computers the script has been tested on	19
Table 3 Red marked events are the ones that fits best to our specifications.....	31

Table of Contents

Affidavit	II
Abstract	III
List of Figures	IV
List of Tables	IV
1. Introduction	1
1.1. Motivation	2
1.2. Background	2
1.3. Approach	3
2. Scientific Computing with Python	4
2.1. The use of ObsPy	4
2.2. NumPy as a fundamental package	5
2.3. SciPy as a supplement for numeric simulation	5
2.4. Matplotlib for visualizing seismic data	6
2.5. SOAP web service as a relocated database	6
3. Fundamental seismic theory behind the scene	7
3.1. Distance and travel times	7
3.2. Instrument response	7
3.3. Rotation of horizontal components	8
3.4. Estimation of spectra	9
3.5. Cross correlation	9
4. Operating mode of the program	10
4.1. Introduction to the operating mode	11
4.2. TauP web service	13
4.2.1. How TauP works	13
4.2.2. The choice of the model	13
4.3. ObsPy routines	17
4.4. Cross correlation in a moving time window	18
4.5. Performance of the program	18
5. The output of the program	21
5.1. Command line response	21
5.2. Unaffected seismic streams	23
5.3. Instrument correction	24

5.4. Transverse acceleration vs. rotation rate	26
5.5. Spectrograms	28
6. Using the program.....	30
6.1. Recent past seismic events	30
6.2. Observations using the program.....	31
7. Conclusions	36
8. Acknowledgments.....	37
9. References.....	38
Appendix	A

1. Introduction

Seismology is a science primarily based on computational processing and inversion of translational ground motions recorded by seismometers. Inertial seismometers measure three components of translational ground displacement and form the basis for any interpretation of seismic events. (Igel *et al.* 2005) There are several global seismic networks that record any ground motion induced by seismic activity. Thanks to the digital revolution, modern seismometers record their measurements digitally and it is easy to use the data for further processing. Despite the fact that there is no real standard for recording and storage of the different seismometers that form a regional or global net, it causes to take a definite form that a few data formats will achieve the world standard. More and more data centers collect and receive snowballing amount of data from all over the world. The availability of high-quality datasets for further computational processing increases every day. Seismology has been used for decades to understand and model the internal structure of the earth. With increasing computational capacities and huge supercomputers seismologists are able to simulate even complex and detailed models of the inner Earth's structure. But until some years ago, there had only been two types of measurements that could be used in these models. The first type, the classical seismometer, which records either analogically or digitally, and the second type aims at measuring the strain (deformation) of the Earth. In the past years, ring laser gyroscopes were developed that close the missing gap in order to fully describe the complete motion of a seismic wave propagating through the earth. (Igel *et al.*, 2005, 2007) By feeding datacenters with those new ground rotation rates as a fourth component for two seismic stations in the world (Christchurch, New Zealand and Wettzell, Germany), seismologists are able to complement the recent seismic models of the earth. Thus, in April 2010 seismologists could observe rotational motions from the Earth's free oscillations (Kurrle *et al.*, 2010). Furthermore, there were several articles in geophysical related journals that pick out rotational motions as a central theme. A group of seismologists established an international working group on rotational seismology in 2007 that made an issue out of the new measurement technique by ring laser technology.

1.1. Motivation

Since a seismologist's field of work changed over the past few years due to several automated systems and a general automation of processes that had to be done by seismologists before, recent work is focused on the interpretation of huge datasets. The availability of internet data for nearly every region on the globe allows seismologists to compute and simulate nearly any scenario induced by seismic activity. As typical for an industry that uses computers to simulate and illustrate data, there are several operating systems and programs used to achieve the desired output. Every seismologist is used to his own architecture and environment and prefers different tools written in different languages as C, C++, FORTRAN, Java™ or Python just to mention the common ones. Furthermore, the quasi standard for scientists regardless of their scientific field is Matlab®. The motivation behind the thesis is to develop a program that enables the seismologist to easily obtain a first overview of translational and rotational ground displacements induced by earthquakes (velocity, acceleration, rotation rate). With the tool developed in this thesis, a seismologist who is interested in transversal and rotational similarity can easily get a first overview of the earthquake he is interested in.

1.2. Background

The objective of scientists who programmed a toolbox for seismology at the LMU in Munich is to provide a base that simplifies the usage of programming and scripting for seismologists.

With this intention, they used the programming language Python to design an easy to use framework for developing programs with unique and special requirements on the typical mathematical methods used in geophysics for processing seismograms (Beyreuther *et al.*, 2010). The toolbox mentioned before is called ObsPy which contains all basic and essential seismological routines a seismologist usually needs. Part of this thesis is to find a solution to handle seismic datasets and process them in several different routines. Due to the rapidly growing popularity of Python in the scientific world in the past few years, it just seems consequential to program a tool in this language. The excellent capabilities to access compiled C and FORTRAN functions enable seismologists to use their old approved codes

within a new coding environment. As a result ObsPy seems to be the best basement for the program we want to develop.

1.3. Approach

To obtain maximized autonomy from local datacenters and to maintain the redundant availability of seismic data 24 hours a day with a uniform storage system for all connected stations, it seems consequent to decide for the ArcLink web service of the GEOFON initiative of the Geoforschungszentrum Potsdam (GFZ).

One goal of the program developed within this thesis was to automate nearly any process and to simplify the usage of the program. To provide a self-explaining in- and output of the program we need to calculate the time-window of the seismic event we want to process. Seismic events in the ArcLink web service are generally stored depending on their time of origin. The ArcLink Client of ObsPy needs three options to receive the time series from the servers: the origin time and the forward- / backward-time of the P-wave arrival. Since we do not know how long the time-window of an earthquake could last until we downloaded some part of the time series and compute the travel times of the individual modes, we need to simulate them by using models. Unfortunately there is no implementation of TauP for Python, so we need to extend the script functionality differently. There are several ways to access the local TauP functionality within Python, but the development of such a client and the necessary Java™-bindings to further process the TauP output does not seem to be the ideal way to develop a largely independent and closed package. Therefore it is reasonable to pick another web service. ORFEUS (Observatories and Research Facilities for European Seismology) provides a web service which allows a direct integration of the TauP functionality with much less effort than the local Java™-bindings.

The rest of the functions added to the program could be obtained within ObsPy, the packages ObsPy is based on, or Python core functionalities.

2. Scientific Computing with Python

Python is a portable, interpretative and object-orientated programming language. It is coordinated by the non-commercial organization Python Software Foundation. It can be used on nearly every operating system like Unix, Windows or Mac OS. Therefore an important requirement in a scientific environment is given. A Python program – often called script -is a text which could be executed by an interpreter. The Python syntax allows very compact scripting and the layout of the code is not only for a better legibility of the code but has a meaning. As an example, the end of a code line is the end of a command. Command-blocks (e.g. the inner of a loop) are defined by insertion. Thus, lines of a code which are inserted at the same place are part of a command-block. Python is object-orientated, imperative and functional. There is no declaration of variables with a data type. The data type is maintained by the context and Python does a type conversion automatically if it is necessary. Multiple variables can be defined by one tuple which includes all the variables. To exchange the value of x and y you just need to type $x, y = y, x$. In addition, common mathematical spelling is supported by the syntax (Weigend, 2008). Altogether Python is a very powerful programming language with an intuitive syntax and an integrated help system. Due to its easy syntax it is the first choice for people who are already familiar with mathematical, analytical software like Maple™, Mathematica or Matlab®.

2.1. The use of ObsPy

As mentioned before, ObsPy is a powerful toolbox developed for the needs of seismologists. Nearly every routine and algorithms often used in seismology is implemented. Waveform import and export functions like GSE2, SEED and SAC support is implemented and allows the user to start with analyzing the seismic data quickly. For this purpose ObsPy offers, besides imaging and signal processing routines like waveform plotting, imaging spectrograms, filtering and rotating data. Additionally ObsPy includes several clients for easy access to data saved in data centers like WebDC or the internal LMU-operated Seishub service. Consequently the user can start with nothing more than his code. The data he wants to analyze could be received from an external source and computed right on the

user's machine. Thanks to NumPy and SciPy that are dependencies of ObsPy the processing of the data goes really fast. (ObsPy Documentation, 2010)

2.2. NumPy as a fundamental package

NumPy is the fundamental package for scientific computing in Python. It is a library that provides several multidimensional array objects. Numerous routines allow fast operations on arrays, including mathematical, logical, sorting, selecting, shape manipulation and discrete Fourier transform just to mention the important ones. On the contrary to the standard Python core, NumPy uses ndarray objects with many operations being performed in compiled code for performance issues. Thus the performance of NumPy is similar to machine-oriented languages like C because it uses pre-compiled C code. Furthermore NumPy is the core of almost every scientific Python code that included mathematical routines. It is more a must-have extension to extend Python's core abilities than just a simple library. (NumPy Documentation, 2010)

2.3. SciPy as a supplement for numeric simulation

SciPy is an additional library which is very useful for scientists. It contains several mathematical algorithms and convenient functions and adds significant power by exposing the user high-level commands and classes for data manipulation and visualization. The user obtain a data-processing and interactive prototyping environment such as Matlab®. With SciPy and Python the user obtains the advantages of both, a very powerful advanced mathematical processing software like SciPy and a very powerful portable connection to the system with the core abilities of Python. With a community continuously developing new features and algorithms like parallel computing and cluster rendering, this combination enables the scientist to compute even computationally intensive algorithms in an acceptable time frame. (SciPy Documentation, 2010)

2.4. Matplotlib for visualizing seismic data

Matplotlib is a python package to generate high quality and ready to publish 2D plots. Its numerous output formats and vector based graphics makes it the ideal package for visualizing seismic data with the ability to easily post-process the plots generated by the program. The possibility of embedding Matplotlib in a GUI library like wxWidgets or GTK+ makes it a really powerful package for further development and improvement. The plotting library uses an object-oriented API (application programming interface) resembling the one you maybe know from Matlab®. (Sandro, 2009)

Thus the pylab interface of Matplotlib is a set of functions which allows the user to maintain plots with quite similar code used for Matlab®. Matplotlib is the core image processor of the ObsPy toolbox and is used for every graphical output. The advantage of interactive manipulation of graphical data by using additional packages as IPython enables the user to achieve the desired results really intuitive and fast. To model 3D models of the Inner Earth, e.g. to simulate wave propagation, one rapidly reaches the limits of Matplotlib. VTK (The Visualization Toolkit) should be the first choice for this purpose. At this point the illustration of seismic data within 3D graphics is not needed.

2.5. SOAP web service as a relocated database

SOAP (Simple Object Access Protocol) is a network protocol with allows to share data between two independent machines only connected through the internet. It provides remote procedure calls which one can swap processes from the host to the remote machine with. SOAP is based on XML (eXtensible Markup Language) to represent the data and TCP/IP (Transmission Control Protocol/ Internet Protocol) to transmit the data. The user does not need to run every process on his own machine or program an interface for the program he wants to implement in his own code. Sometimes it is reasonable to outsource some processes to either conserve system resources or to reduce dependencies of software the user might need to install elsewhere. In this thesis SOAP is used to save the user from installing TauP which is a program to compute travel times and which is written in Java™. Even though Java™ is also system platform independent it is rather to spare the user the installation and configuration of additional software.

3. Fundamental seismic theory behind the scene

To understand the mathematical and general connection of the routines used in this program we need to recapitulate the fundamental theories and algorithms. Due to the limitation of this bachelor's thesis, only the most important formulae are revised.

3.1. Distance and travel times

To obtain the corresponding time-window for a specific earthquake, we need to know the latitude, longitude and depth of the event source and the longitude and latitude of the seismic station recording the seismic event. There are a number of formulae that are able to calculate accurate azimuths and distances on the WGS84 ellipsoid. One formula that works with an accuracy of a few millimeters, ranging within a distance of a few cm to 20.000km, is the Vincenty's formulae. As we want to know the distance from the event source to our seismic station we are confronted with an inverse problem. The formula to calculate the epicentral distance s from source to station is given by

$$s = bA(\sigma - \Delta\sigma) \quad (1)$$

where b is the length of the minor axis of the ellipsoid and σ the angular distance on the sphere. (Wikipedia, 2010) There are several models to calculate the ray path for a specific phase velocity. But we only need the arrival times of the corresponding phases, especially the last phase to obtain the parameter for the time-window. To derive it, we just need to divide the phase velocity by the distance.

3.2. Instrument response

Modern seismographs like the STS-2 Streckheisen seismometer located at Wettzell, Germany, are designed to achieve a linear response to Earth motions over a wide frequency band. It is desirable to have an instrument as sensitive to be able to record below typical Earth noise levels and at the same time be able to record large earthquakes. To obtain such

results modern seismograms use force-feedback designs in which the mass is maintained at a fixed point. Thus the measurement is not done by measuring the amplitude of the mass due to seismic activity but the force that is required to keep the mass at the fixed point. Consequently the STS-2 uses three identical mechanical sensors to measure the horizontal and vertical components of a seismic event. Furthermore the output proportional to velocity is not filtered and the feedback system delivers it directly from the feedback loop. The response from the seismometer is defined by a corner frequency from 8.33 mHz up to 50 Hz. (STS-2 Manual, 1995) The instrument's response can be defined in terms of the relationship between digital counts in the recorded time series and the actual ground motion of the Earth. The gain of an instrument is the ratio between the digital counts and the measurement of the ground motion. All information specifically oriented and calibrated sensors are described by the complete instrument's response function which is embedded in the seismic record that can be downloaded from datacenters. To clear the time series and enable the seismologist to further process the data we have to subtract the instrument response function given by poles, zeros, gain and damping. Thus we convert poles and zeros to frequency response and the resulting output contains the frequency zero which is the offset of the time series. For the G-ring no instrument correction is necessary because of the uniform transfer function of optical sensors due to a mass-less recording system. The raw data is converted and scaled correctly to compare with the seismometer's data.

3.3. Rotation of horizontal components

To rotate the North- and East-Component of a seismogram to Radial and Transversal components, we need to calculate the back-azimuth. The back-azimuth is the angle measured between the vector pointing from the station to the source and the vector pointing from the station to geographic north. To compute the back-azimuth ObsPy uses the WGS84 ellipsoid model and the algorithm from the Geocentric Datum of Australia Technical Manual. The rotation of the seismograms itself for the radial component r and the transversal component t is done by

$$r = e * \sin\left(ba + 180 * \frac{2\pi}{360^\circ}\right) + n * \cos\left(ba + 180 * \frac{2\pi}{360^\circ}\right) \quad (2)$$

$$t = e * \cos\left(ba + 180^\circ * \frac{2\pi}{360^\circ}\right) - n * \sin\left(ba + 180 * \frac{2\pi}{360^\circ}\right) \quad (3)$$

3.4. Estimation of spectra

To identify periodicities in the seismic data and estimate e.g. the Earth's normal modes, we need to calculate the logarithmic spectrum of the transversal and rotational component. This is done by filtering the record by removing all energy near and above the Nyquist frequency first and then using a sampling interval so that the Nyquist frequency lies above the highest frequency in the original data. Then the data gets split into NFFT (Nonequidistant Fast Fourier Transform) length segments, and computed for the power spectral density (PSD) by Welch's average periodogram method (Welch, 1967). Each segment is then detrended by removing a best fit line and windowed by x times with the hanning window of the time sequence. So the sequence is tapered to smooth the ends of the time series to zero which reduces spectral leakage but broadens central peaks. The last step pads the time series with zeros to make the number of samples a power of 2 for the FFT (Fast Fourier Transform). This also smooth the spectrum by interpolation while not increasing the actual resolution, but giving more points in the corresponding plot. (Bendat *et al.*, 1986; Shearer, 2009; Gubbins, 2008)

3.5. Cross correlation

In signal processing, cross correlation is a measurement technique to check the similarity of two given time sequences. Here we want to check if the rotated horizontal components correlate with the rotation rate at the same time. The cross-correlation of two functions $a(t)$ and $b(t)$ is equivalent to the convolution of $a * (-t)$ and $b(t)$. The cross correlation of two time series a and b is given by

$$c_k = \frac{1}{N + M - 1} \sum_p a_p b_{k+p} \quad (4)$$

where k is called the lag, N and M are the lengths of a and b , and the sum is taken over all possible products. $(N + M - 1)$ (Shearer, 2009)

The correlation coefficient for the whole seismogram is equal to the cross correlation normalized function. It returns the value 1 if two sequences are perfectly correlated and identical. The correlation coefficient is defined as

$$\psi_k = \frac{\sum_p a_p b_{k+p}}{\sqrt{\sum_p a_p a_p \sum_p b_p b_p}} \quad (5)$$

If $\psi_k = 1$ there is a perfect correlation of two time sequences, but in general it is considered as $|\psi_k| \leq 1$ for real seismic data. (Shearer, 2009)

The empiric correlation coefficient used for computing the sliding time window is defined by

$$\kappa = \frac{\frac{1}{n} \sum_p (a_p - \bar{a})(b_p - \bar{b})}{\sqrt{\sum_p (a_p - \bar{a})^2 \cdot \sum_p (b_p - \bar{b})^2}} \quad (6)$$

With this formula (Wikipedia, 2010) we calculate the correlation of all 600 samples (30s) and plot the highest correlation value of each window. The resulting function is then plotted underneath the superposition of the transversal acceleration and rotation rate to provide a good overview.

4. Operating mode of the program

The program was written and tested under a Windows™ environment with several packages installed. To run the program it is inevitable to ensure to have installed all necessary dependencies. It is highly recommended to run the program under a 64bit system because the program needs a lot of system memory to calculate and plot the spectrograms. Under the precondition of a 64-bit enabled processor and more than three Gigabytes of RAM (Random-Access-Memory) it is recommended anyway. For a 32-bit Windows™ system you have to consider that only 3.5 GB of RAM could be addressed by Python and the program could break the operation because of a memory error.

The essential programs you need to install and that are tested are:

- Python 2.6.5
- Numpy 1.4.1
- SciPy 0.8.0
- LXML 2.2.7
- Matplotlib 1.0.0
- pyPDF 1.12 (optional and only if you want to convert the plots to PDF)

4.1. Introduction to the operating mode

To give you a quick overview of the operating mode without looking in the code of the program in detail, Figure 1 provides you a schematic sketch of the routines used to generate the output. First of all the user defines some core variables the script needs to operate. Therefor he needs to type in the origin UTC (Universal Time Coordinated) time of the earthquake he is interested in, the longitude and latitude, depth and the event name. These values can be obtained from the WebDC web service or an email alert system like the one provided by the GFZ Potsdam¹. Optionally he can specify a personal directory for the desired event on the user's machine. If no directory is defined the script would write every data to the directory where the script got executed.

¹ GFZ Potsdam – Earthquake Bulletin: <http://geofon.gfz-potsdam.de/db/eqinfo.php>

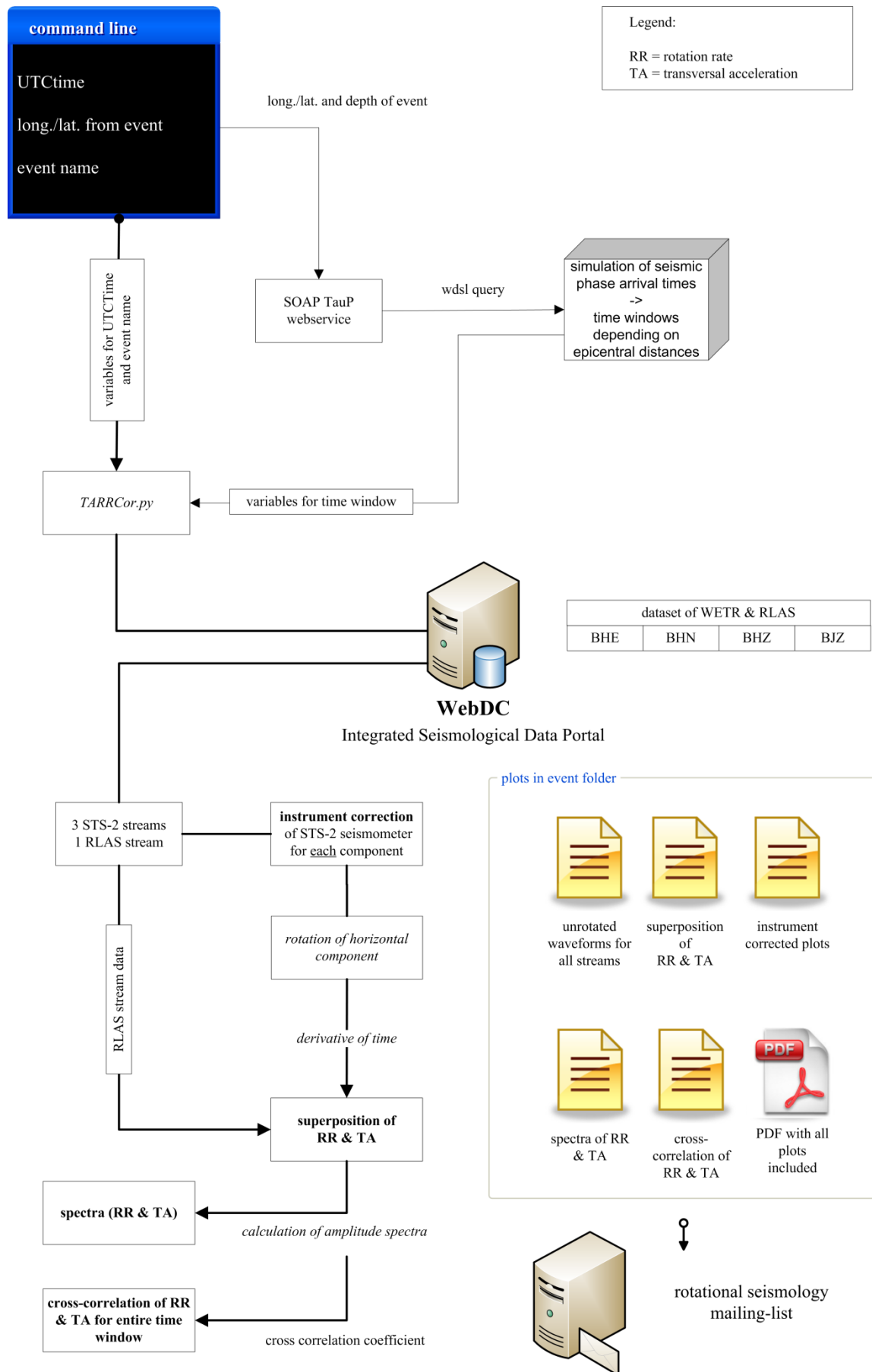


Figure 1 Schematic sketch of the operating mode

4.2. TauP web service

The purpose of the TauP web service, operated by ORFEUS, is to emphasize the use of their SOA (Service Oriented Architecture) and to directly access the backbone services. Typically a user is just interested in the upper layer of the architecture provided by ORFEUS. But our script needs a machine-to-machine-communication to calculate everything in the background without any further work for the user. Therefore we need a direct connection to the ORFEUS services to simulate and calculate the travel times we need to further process our own routines. This direct connection to the ORFEUS servers is provided by the Taup WS². It offers the possibility to compute arrival times using a few default velocity models.

4.2.1. How TauP works

TauP is a Java™ based software package developed by the University of South Carolina. With the package the user is able to compute and calculate seismic travel times for every single phase of an earthquake. The method behind TauP is described by Buland and Chapman (1983). The IASPEI times package is a widely-used implementation of the methodology.

With TauP the user could extract derivative information, such as ray paths through the earth, pierce and turning points, as well as travel time curves. (Crotwell *et al.* 1999). But the web service only offers the possibility of calculating travel times. Since this is the only reason for using this service, it serves our needs.

4.2.2. The choice of the model

There are two default models available for the simulation of travel times. The first one, *iasp91* was developed by Kennett and Engdahl (1991) and the second one *ak135* by Kennett et al. (1995). Both models can be downloaded from IRIS (Incorporated Research Institutions for Seismology)³.

The second model *ak135* “gives a significantly better fit to a broad range of phases than

² Taup WS: <http://www.orfeus-eu.org/wsdl/taup/taup.wsdl>

³ IRIS –DMC: <http://www.iris.edu/software/downloads/processing/>

provided by the *iasp91* or *sp6* models ... The differences in velocity between *ak135* and these models are generally quite small except at the boundary of the inner core, where reduced velocity gradients are needed to achieve satisfactory performance for PKP differential time data.”⁴ Thus in general, the *ak135* tables represent an update of the *iasp91* model and try to match the behavior of a wider range of seismic phases. It is a radially stratified velocity model and provide a consistent basis for all phases. The P wave travel times are very similar to *iasp91*, but changes in S and core phases justify the use of the newer *ak135* model. To provide an example of the single phase arrival times of a specific earthquake, we compute the arrival times with the *ak135* model for a seismic event at Nicobar Islands, India Region at 318.67° epicentral distance from Wettzell, Germany and a depth of 11 km. As you can see in Table 1, the perfect time window for this event should be about 3250s from origin time.

delta	#	code	time(s)	min	s	dT/dD	dT/dh	d2T/dD2
318.67	1	P	465.59	7	45.59	8.2118	-1.56E-01	-4.55E-03
	2	pP	469.01	7	49.01	8.2181	1.56E-01	-4.55E-03
	3	sP	470.47	7	50.47	8.2167	2.88E-01	-4.55E-03
	4	PP	561.11	9	21.11	10.8373	-1.42E-01	-6.30E-03
	5	PnPn	565.71	9	25.71	11.6905	-1.37E-01	-2.53E-03
	6	PP	569.72	9	29.72	9.2174	-1.51E-01	-4.28E-02
	7	PcP	583.74	9	43.74	3.2722	-1.70E-01	5.87E-03
	8	ScP	813.63	13	33.63	3.9661	-2.95E-01	6.31E-03
	9	PcS	815.02	13	35.02	3.9665	-1.69E-01	6.31E-03
	10	S	840.70	14	0.70	14.8320	-2.66E-01	-3.26E-03
	11	pS	844.82	14	4.82	14.8480	1.09E-01	-3.27E-03
	12	sS	846.55	14	6.55	14.8413	2.66E-01	-3.27E-03
	13	PKiKP	1011.45	16	51.45	0.8895	-1.72E-01	1.53E-02
	14	pPKiKP	1015.24	16	55.24	0.8894	1.72E-01	1.53E-02
	15	sPKiKP	1016.62	16	56.62	0.8894	2.98E-01	1.53E-02
	16	SS	1025.70	17	5.70	19.8944	-2.38E-01	-2.58E-03
	17	SS	1032.48	17	12.48	22.2673	-2.20E-01	-1.35E-03
	18	SnSn	1034.45	17	14.45	24.0409	-2.04E-01	-1.05E-02
	19	SS	1035.45	17	15.45	23.7658	-2.07E-01	6.38E-03
	20	SS	1038.39	17	18.39	16.6262	-2.57E-01	-7.82E-03
	21	ScS	1069.65	17	49.65	6.0569	-2.93E-01	3.13E-03
	22	SKiKP	1223.22	20	23.22	0.9343	-2.97E-01	1.48E-02
	23	PKKPdf	1895.93	31	35.93	-0.9490	-1.72E-01	-1.40E-02
	24	SKKPdf	2107.73	35	7.73	-0.9011	-2.98E-01	-1.46E-02
	25	PKKSdf	2109.11	35	9.11	-0.9010	-1.72E-01	-1.46E-02
	26	SKKSdf	2320.80	38	40.80	-0.8575	-2.98E-01	-1.53E-02
	27	P'P'df	2401.12	40	1.12	-1.2221	-1.72E-01	-1.18E-02
	28	P'P'ab	2472.90	41	12.90	-4.3499	-1.68E-01	4.68E-02
	29	S'S'df	3254.04	54	14.04	-0.9656	-2.97E-01	-1.38E-02

Table 1 Simulation of arrival times with the *ak135* models for all phases of the Nicobar Islands event on 2010-06-12. dT/dD describes the travel time with respect to distance, dT/dh the travel time with respect to source depth and d2T/dD2 the second derivative of travel time with respect to distance.

⁴ Text quotes are from Kennett and Engdahl (1991) & Kennett et al. (1995)

To complete the review of the *ak135* model, a plot of all phases dependent on epicentral distance and time for the whole earth model described by *ak135* is provided. As you can see in Figure 2, the interesting P-wave, S-wave and core-wave phases should be visible on the recording seismogram within 40min. This conclusion correlates with the example data of Nicobar Islands in Table 1.

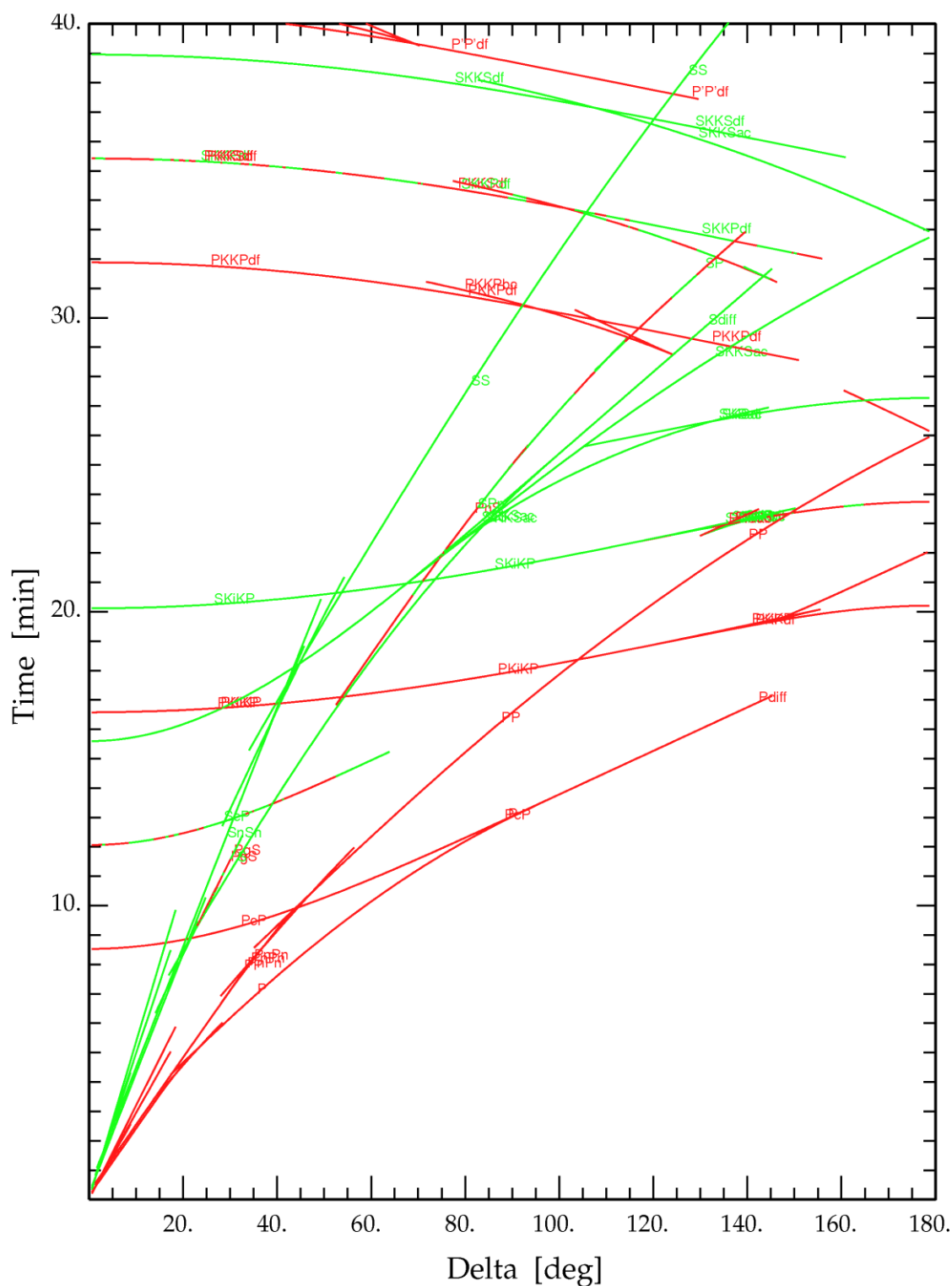


Figure 2 Plot of all simulated phases calculated with the *ak135* model

Now that we know the individual arrival times of all phases, we can automatically calculate the corresponding time-window for every seismic event computed with the program.

As tested for several depths and distances the S'S'df phase always arrives in an interval of 3240-3260 seconds. The assumption that every direct phases arrives within 3260 seconds results in a time window of 3260 seconds. Unfortunately the real case scenario differs from our theory and additional time needs to be added to analyze the seismic data in total.

Since we want to be able to see every phase of the event, we start 300 seconds before origin time, then double the S'S'df phase time and add additional time for the individual distance of the event from the seismic measurement station in Wettzell, Germany.

Our formula used to calculate the time window is not based upon a theory but reflects the experiences gained during the testing of the script.

So for the time window we get the easy formula

$$\theta = 300 + 2 \mu + \frac{\gamma}{10} \quad (7)$$

where θ is the time window in seconds, μ is the S'S'df phase arrival and γ is the event distance converted 1:1 from km to s.

4.3. ObsPy routines

After computing the arrival times for the single phases we have all parameters to start our database query and to receive the seismic time series. For the WebDC query we use the ArcLink client routine from *obspy.arclink*.

The client class only needs the origin time and the time window before and after it. To get a better overview we set the prior time window to a fixed value of 300s and the past time window to the double of the last arriving phase plus an additional 300s window. For our example in Table 1 we therefor receive a 6850s time window. (Figure 3) We double the last phase value to not only receive the arrivals of the first incomes but rather any reflected phase of the event.

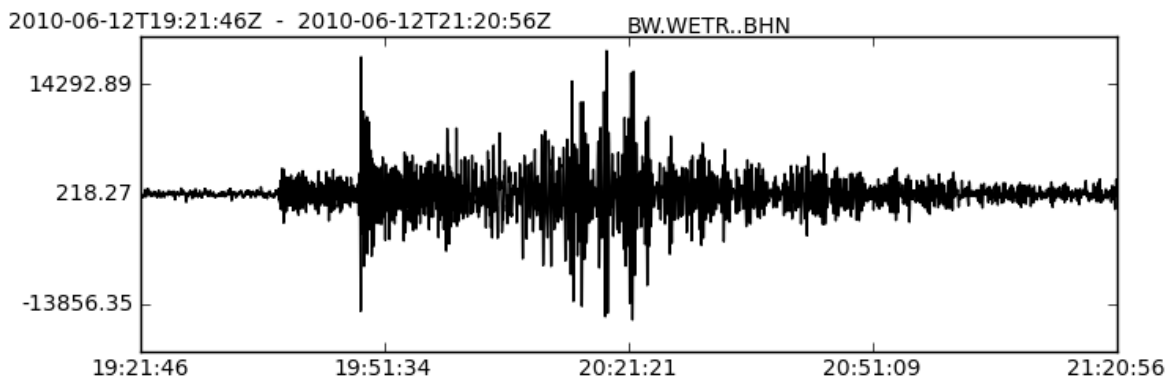


Figure 3 Uncorrected BHN component of the Wettzell seismometer for Nicobar Islands event at 2010-06-16 with UTC time on the time axis and some undefined amplitude values by now.

After receiving all four channels (three from the seismometer and one for the Ringlaser) and writing them to disk, we need to correct them for their instrument response. As we get the data as integer values that are transformed to a stream/trace object within ObsPy we need to transform them to float64 NumPy arrays and demean them. The demeaning is necessary to obtain a smooth signal without any offset at the beginning and at the end of the time series. The next step before correcting the signal for the instrument response is to filter the signal through a bandpass filter from 100mHz to 40 Hz. This is done by *obspy.signal* routine *bandpass*. The actual instrument correction is then done by *corn-Freq2Paz* and *seisSim* that are also part of *obspy.signal*.

Now we have the corrected signals from all channels, even if we do not need a correction for the Ringlaser data due to its properties as an optical sensor. The next step is the rotation of the horizontal components with *rotate_NE_RT*, where we need the BHN and the BHE channel and the back-azimuth calculated with *gps2DistAzimuth* both from functions are provided by *obspy.signal*.

Since the derivative of the rotated horizontal component should in theory be the same as the rotation rate recorded from the Ringlaser, the rotated signal is differentiated using a core NumPy routine *np.diff*. As it is common to compute a spectrogram for every time series, we also provide a calculation of the spectra of the transversal acceleration and rotation rate. This is done by the *spectrogram* class within *obspy.imaging*.

4.4. Cross correlation in a moving time window

As one of the specifications of the program should be a cross correlation coefficient in a sliding time window along the time series, it has been necessary to develop an additional class to provide this feature. ObsPy does not include such a function so far. After defining a time window of 30s and determining the automated moving steps until the end, every time window is computed with *np.corrcoef* which is in general the same as equation 7. The corresponding maximum value of the correlation is then plotted underneath the correlated super positioned time series.

4.5. Performance of the program

As mentioned before, the performance of NumPy is similar to machine-oriented languages like C because it uses pre-compiled C code. That might be correct in theory but in the testing and development environment used for this purpose, there had been some unlovely performance leaks using all provided ObsPy classes.

To make the results comparable we need to introduce the machines the code had been tested on. (Table 2) A very surprising fact of the Intel® computer is, that although he has dual-core capabilities and AMD64 support, it only uses one core and as a consequence halve the

potential computing power of the CPU. Since the Intel® CPU is generally much faster than the old AMD CPU, there are almost no performance leaks as expected before.

Intel® Core™2 Duo T7300 @ 2.00 GHz	AMD Athlon™ X2 @ 2,41GHz
Windows 7 Professional 64bit	Windows 7 Ultimate 64bit
2x DDR2 800 Hexon 2048 MBytes	2x DDR2 800 Corsair 1024 MBytes
Seagate ST160823AS @7200 RPM	WD Raptor X 150 @10000 RPM
Python 2.6.5 [MSC 64bit AMD64]	Python 2.6.5 [MSC 64bit AMD64]

Table 2 Details of the benchmarked computers the script has been tested on

To illustrate the single steps and their resulting performance, we measure the time needed for every single process to be computed by one of the machines. This measurement is done by a virtual stopwatch correlated with the output of the script. To compare both performances, a random 5000s long sample from the WebDC database mirrored at the LMU is taken (the server is much more stable than the one hosted by webdc.eu). Both machines will compute exactly the same routines and are using the same architecture. This is one of the main specifications of the benchmark.

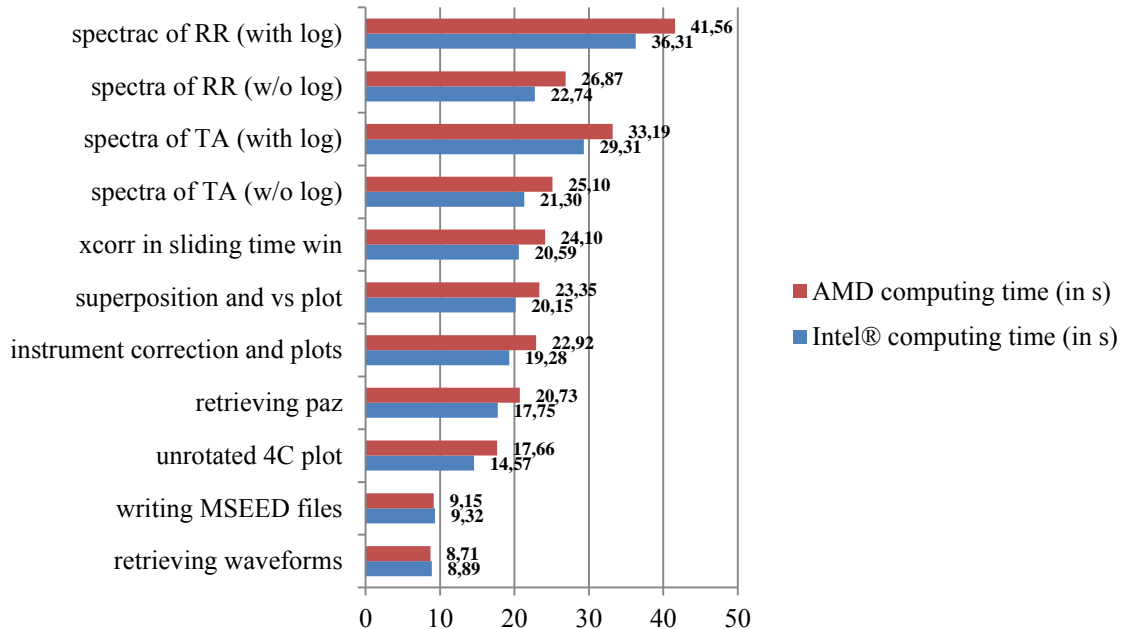


Figure 4 Benchmark results from a synthetic 100.000 sample noise test

During our benchmark (Figure 4) with a 5000s long random time series, with nothing more than noise, we got acceptable results in the computing time and just needed 216 MB of RAM. With a real seismic event data from Nicobar Islands, India Region with coordinated 7.7N and 92.0E and a 6850s long time series we cannot compute the logarithmic diagram

any longer and need 829 MB of RAM to pass all the steps. The log-view of the spectrogram had to be interrupted after 2 hours computing time and 6,5 Gbytes RAM usage.

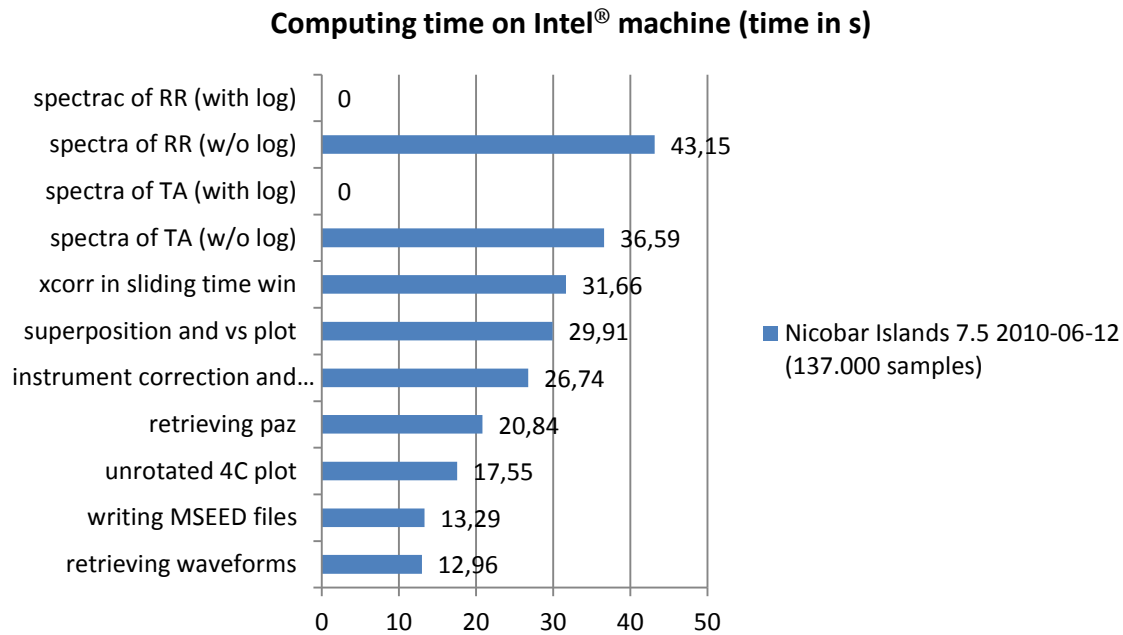


Figure 5 Computing time of calculations done during Nicobar Islands processing

The heavy performance leaks experienced within computing real seismic data have their origin in the insufficient memory available and so the data got swapped on the hard drive which is about a factor 100.000 slower than the RAM. (mechanical device vs. electronic device) The read/write speed of a hard drive is much slower than RAM, and the technology of a hard drive is not equipped to access small pieces of data at a time. If the system has to rely too heavily on virtual memory, you will notice a significant performance drop. If you do not have enough RAM installed in your machine the operating system has to constantly swap information back and forth between RAM and the hard disk. The process is called trashing and slows down your computer. Since the logarithmic diagram of the spectrogram produces a very large data set with thousands of small pieces, the missing memory causes an unacceptable performance. In another example with 150.000 samples (7500s) the script was run for nearly two days (42 hours) without any result and the decision was made to interrupt the computing. Since about 20 GBytes of RAM would be needed in the mentioned case to calculate the spectrogram with full speed, it is not recommended to run the script on a single desktop machine. Python in general is able to use multi-threading and multiple cores, but the architecture for it is just in development and not fully supported yet. Furthermore the cluster for the calculation is not accessible by now since the script needs a Python environment. In summary, the average computing time for all steps of the script for

a common seismic event lies around 1-3 minutes if the logarithmic mode for the spectrogram is turned off.

5. The output of the program

All the previous work leads to the output and plots of the script and now the result of the processed seismic data with the script is discussed in detail. Since we used the Nicobar Islands 7.5 Magnitude earthquake from 2010-06-16 for earlier demonstration, we will now continue using it and explain every plot separately.

5.1. Command line response

All the input and written output is done in the command line or shell of your operating system. First of all, the user sets the global event variables that the script needs to operate normally.

The input of the event name results in the global title on every plot. So it is quite important to set a valid and easily understandable variable. In our example *Nicobar Islands 7.5 Mag 2010-06-12* is used as it describes shortly what event is displayed on the plots.

The second variable is the event time. It names the origin time of the seismic event in UTC time. A valid input for our example would be *2010-06-12 19:26:47*. Other time formats are also supported of course, but they might not be that comprehensible. If you want to input the data in another format, please look up the Obspy manual and especially the class *UTCDateTime* under *obspy.core*.

The third variable is the event depth that you need to input in kilometers.

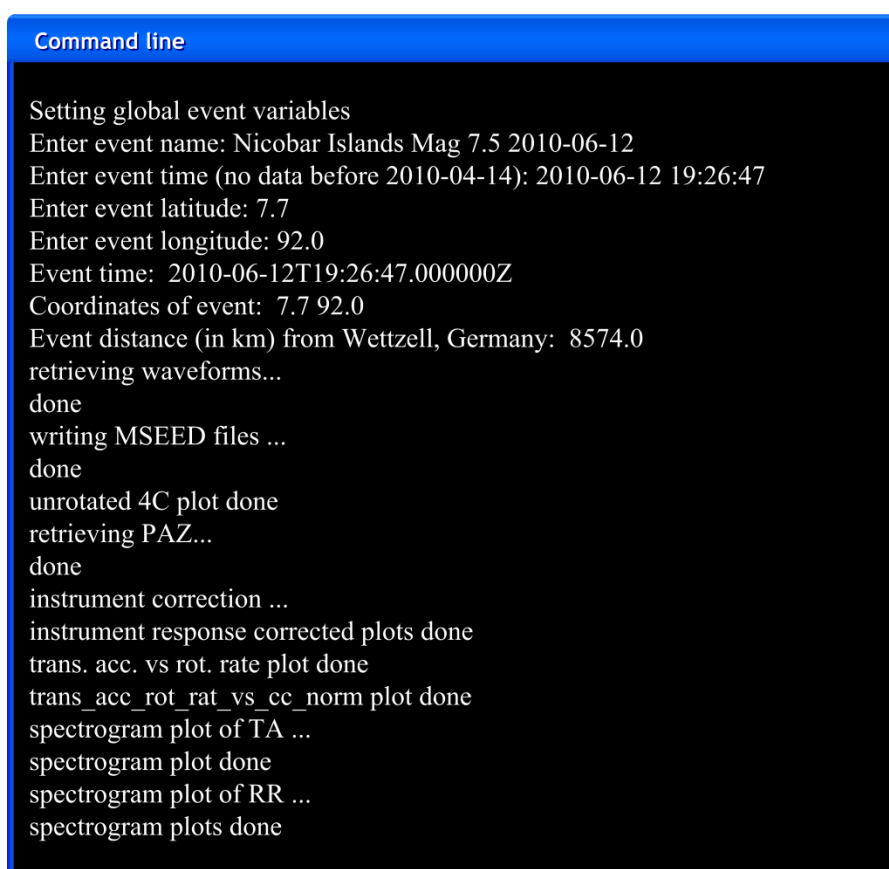
The fourth variable you need to enter is the event latitude, where values on the northern hemisphere are put in as positive value (e.g. 7.7 for 7.7N) and values on the southern hemisphere as negative values (e.g. -9.2 for 9.2S).

The last value the user needs to input is the longitude of the event in the same manner as the latitude: a positive value for the eastern component and a negative value for the west component.

After executing the last variable the program starts to operate. The output of the program summarizes the event time and coordinates for you and optionally returns the path the data would be saved in if you declared it. If set on default, the script would save every data in the directory it got executed.

The first real calculation and output is done in the next line. The user receives the event distance of the event location to our measurement station in Wettzell, Germany located 49.144°N and 12.878°E. These values are default values and cannot be changed within the command line.

The following output lines (Figure 6) are just visual feedback for the user to follow the further progress of the program. Due to the characteristics of the client service of ObsPy and the sometimes strange behavior of the ArcLink protocol, the program might interrupt as it tries to receive waveforms or PAZ (poles and zeros) for the instrument correction. The user only needs to start the script again to process further. A loop of the program appears to be needless during development since even an enhancement of the timeout interval seems useless and causes the same timeout error over and over again.



```

Command line

Setting global event variables
Enter event name: Nicobar Islands Mag 7.5 2010-06-12
Enter event time (no data before 2010-04-14): 2010-06-12 19:26:47
Enter event latitude: 7.7
Enter event longitude: 92.0
Event time: 2010-06-12T19:26:47.000000Z
Coordinates of event: 7.7 92.0
Event distance (in km) from Wettzell, Germany: 8574.0
retrieving waveforms...
done
writing MSEED files ...
done
unrotated 4C plot done
retrieving PAZ...
done
instrument correction ...
instrument response corrected plots done
trans. acc. vs rot. rate plot done
trans_acc_rot_rat_vs_cc_norm plot done
spectrogram plot of TA ...
spectrogram plot done
spectrogram plot of RR ...
spectrogram plots done
  
```

Figure 6 Example command line output of the Nicobar Islands event

5.2. Unaffected seismic streams

To receive a quick overview of all data downloaded from the servers, the first plot provides the user with an unaffected display of the four different seismic streams. Here the user could pre-check if the download of the data went right. Towards the following display of the data, the time axis is presented with original UTC time here and the amplitude axis is given in integer values directly received from the stored sensor data.

2010-06-12T19:21:46Z - 2010-06-12T21:15:56Z

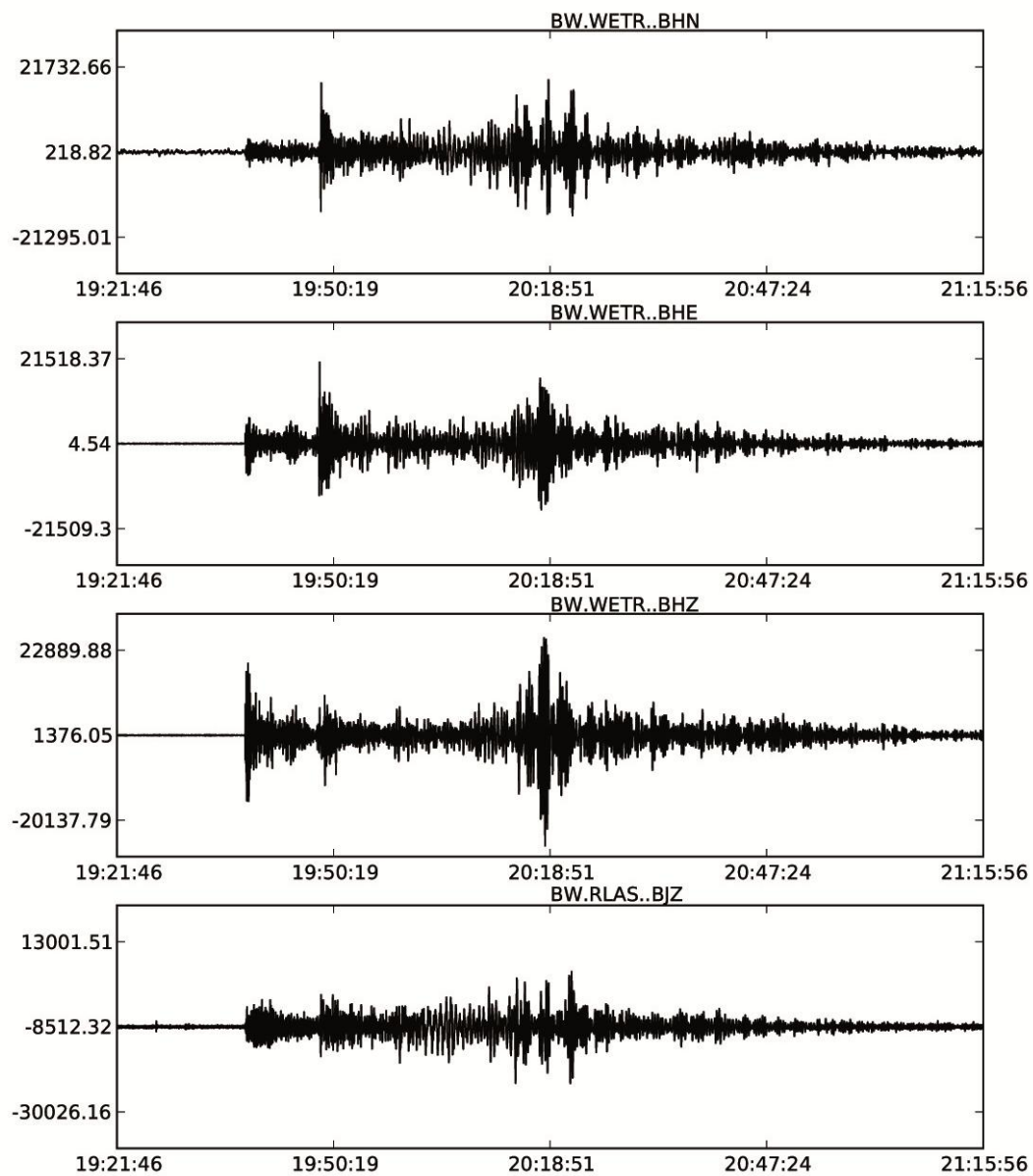


Figure 7 Unaffected seismic streams for the Nicobar Islands event

5.3. Instrument correction

To really work with the data and gain an excess value we need to correct the data for their instrument response (Figure 8-10). This is done by the next step and results in the following plots. The upper seismogram is the original seismogram received from the storage server and the second seismogram is the instrument corrected data with a now defined amplitude axis in nm/s . The corrected seismogram is now used for all further steps and calculations. Unlike the STS-2 channels, the rotation rate is normally presented in $\times 10^{-4} \text{ rad/s}$, but here (Figure 11) we doubled the values due to the fact, that “transversal acceleration and rotation rate should have the same waveform and their amplitudes should scale proportionally to phase velocity depending on wave type” (Igel *et al.*, 2007). Thus the user is able to correlate the two time series he is interested in by just looking at them.

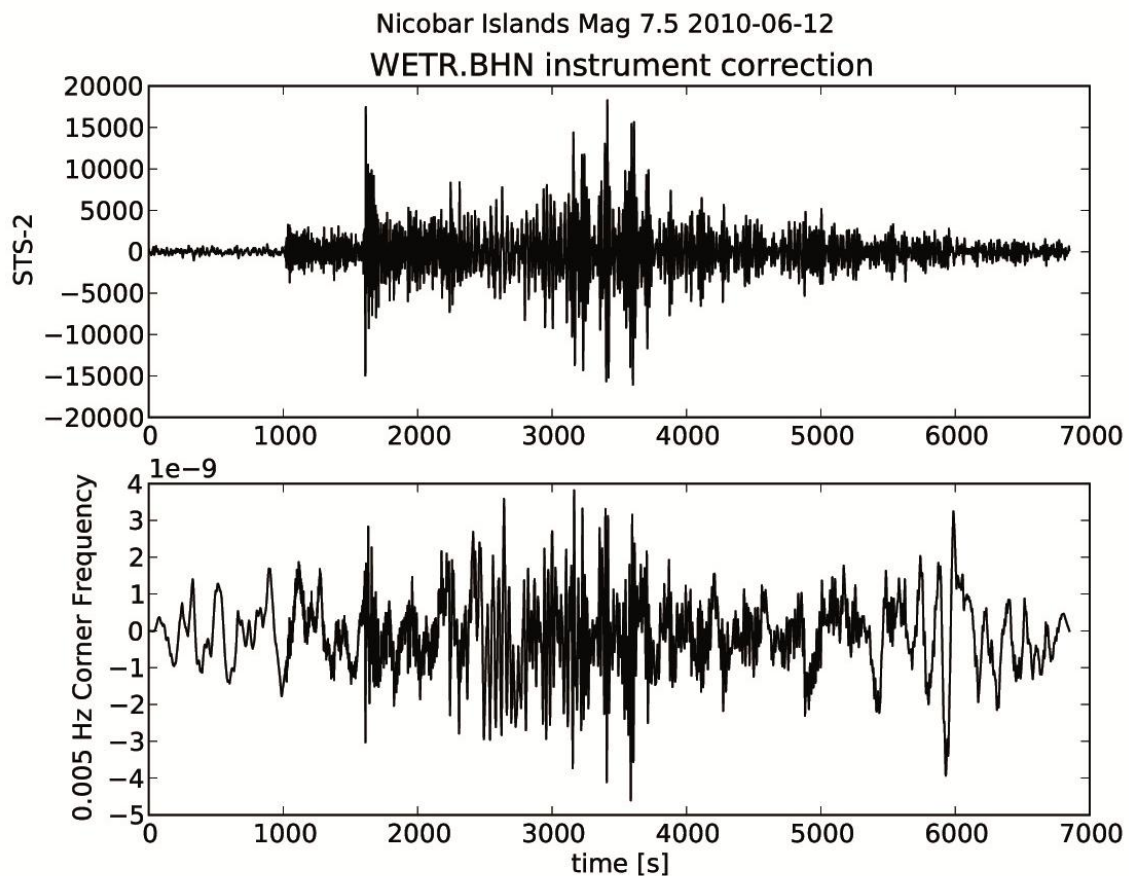


Figure 8 Corrected signal of the first horizontal component of the seismometer

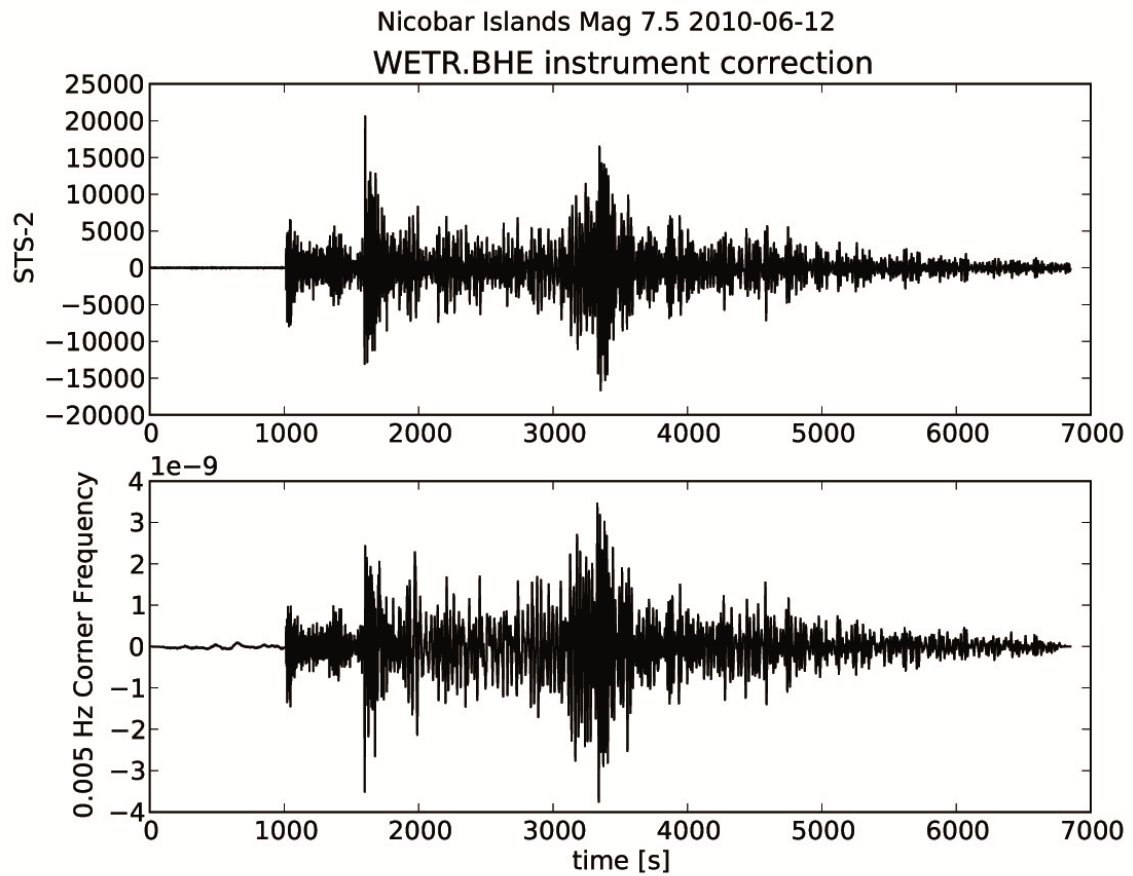


Figure 9 Second horizontal component of the seismometer

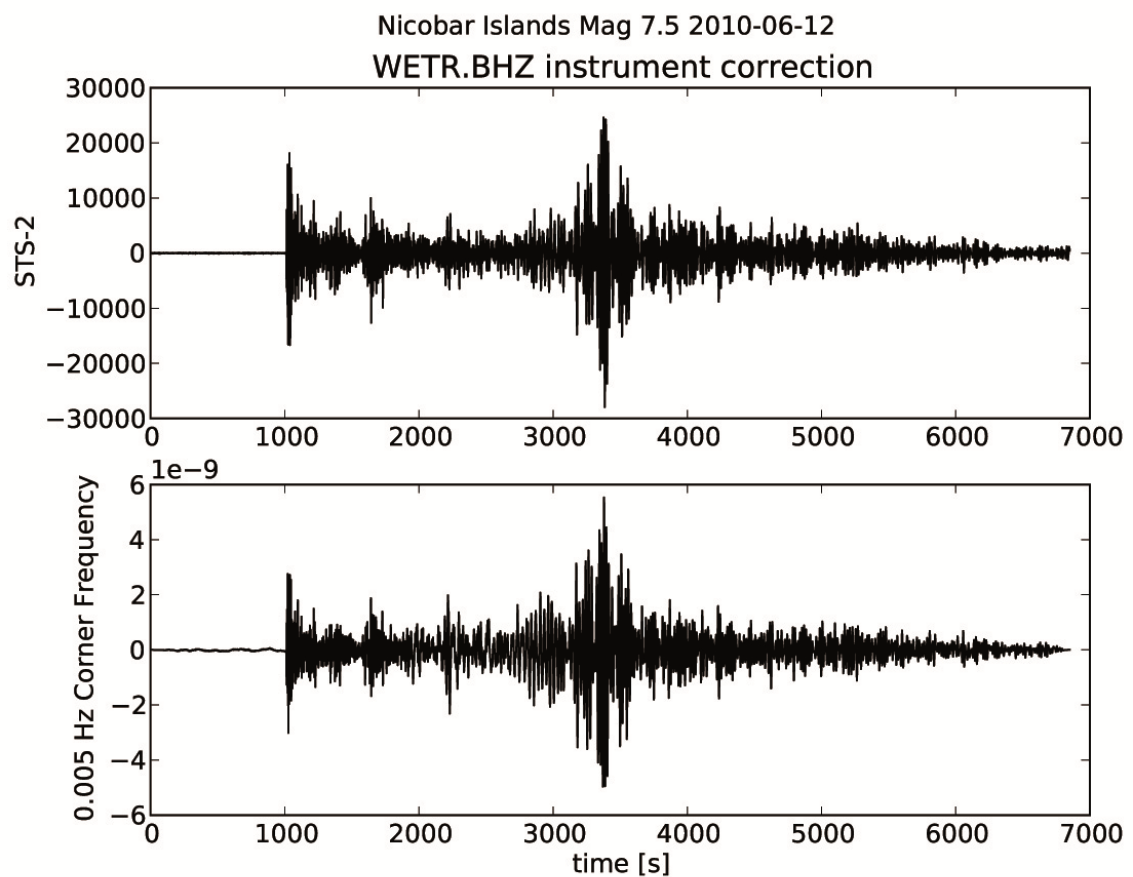


Figure 10 Vertical component of the seismogram

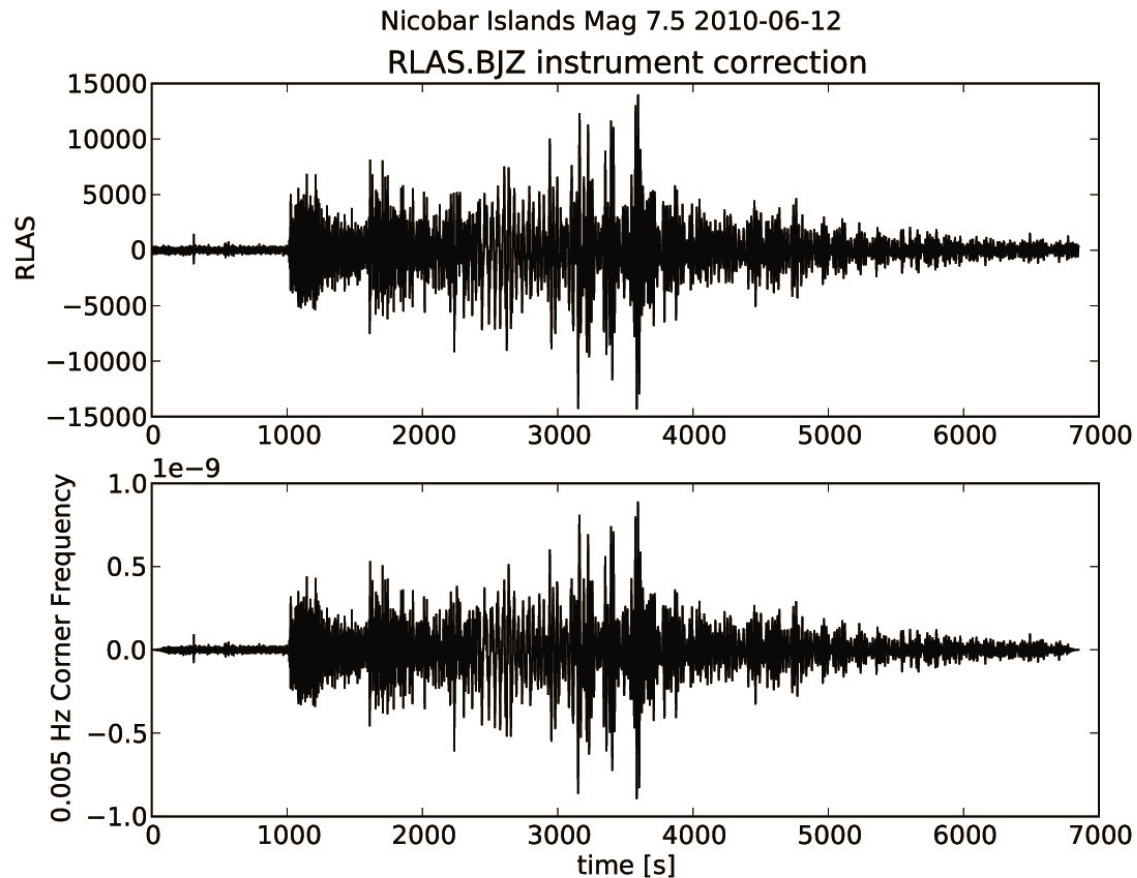


Figure 11 Ringlaser component with fixed nm/s axis for better coherence with other channels

5.4. Transverse acceleration vs. rotation rate

One of the main guidelines and ideas behind this thesis was the consideration of the transversal acceleration and the rotation rate measured by the Ringlaser. To obtain the transversal acceleration we rotate the horizontal seismometer components and then differentiate the result.

Due to the fact that this step is one of the most important ones the script provides three plots for it. One plot displays both time series separately (Figure 12), the second one is a superposition plot of the two signals (Figure 13) and the third one is a superposition plot with a sliding time window directly underneath the superposition to show the correlation coefficient and their coherence (Figure 14). The numerous peaks in the maximum cross-correlation coefficient are an evidence for a good correlation of both time series.

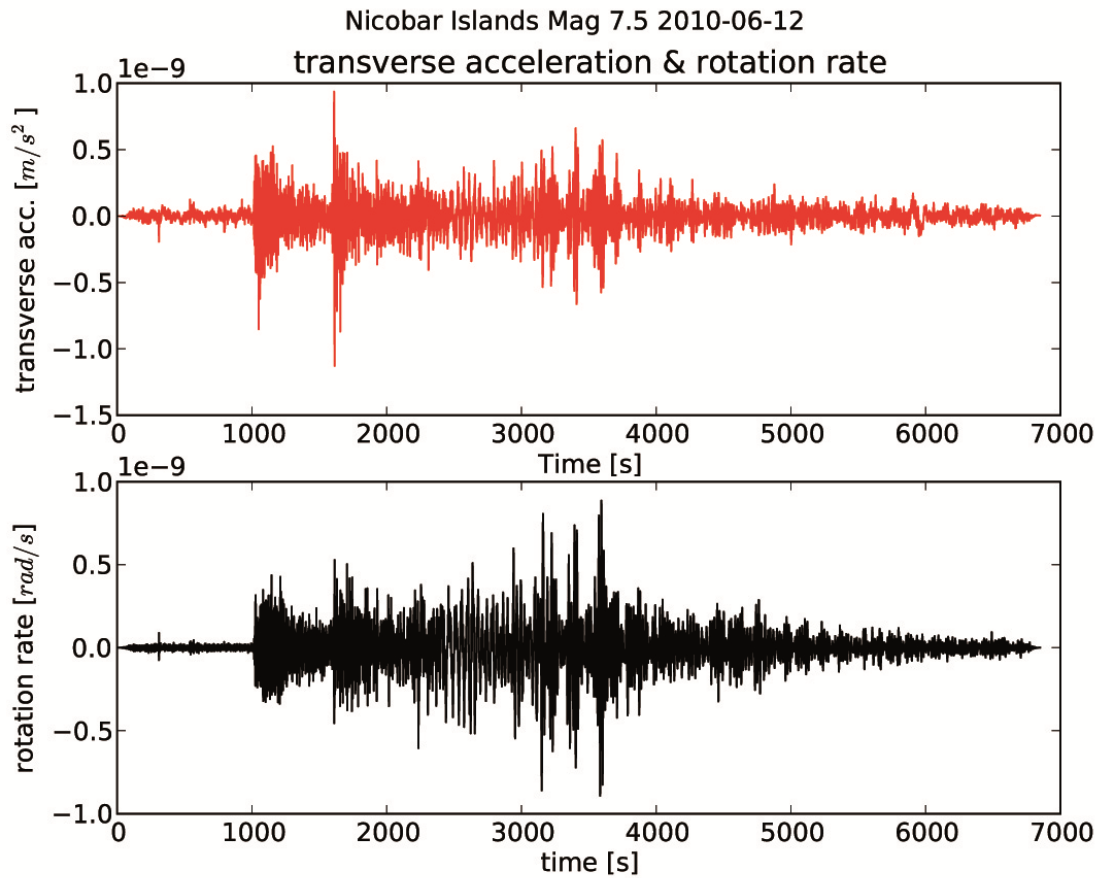


Figure 12 Separated transversal acceleration & rotation rate time series

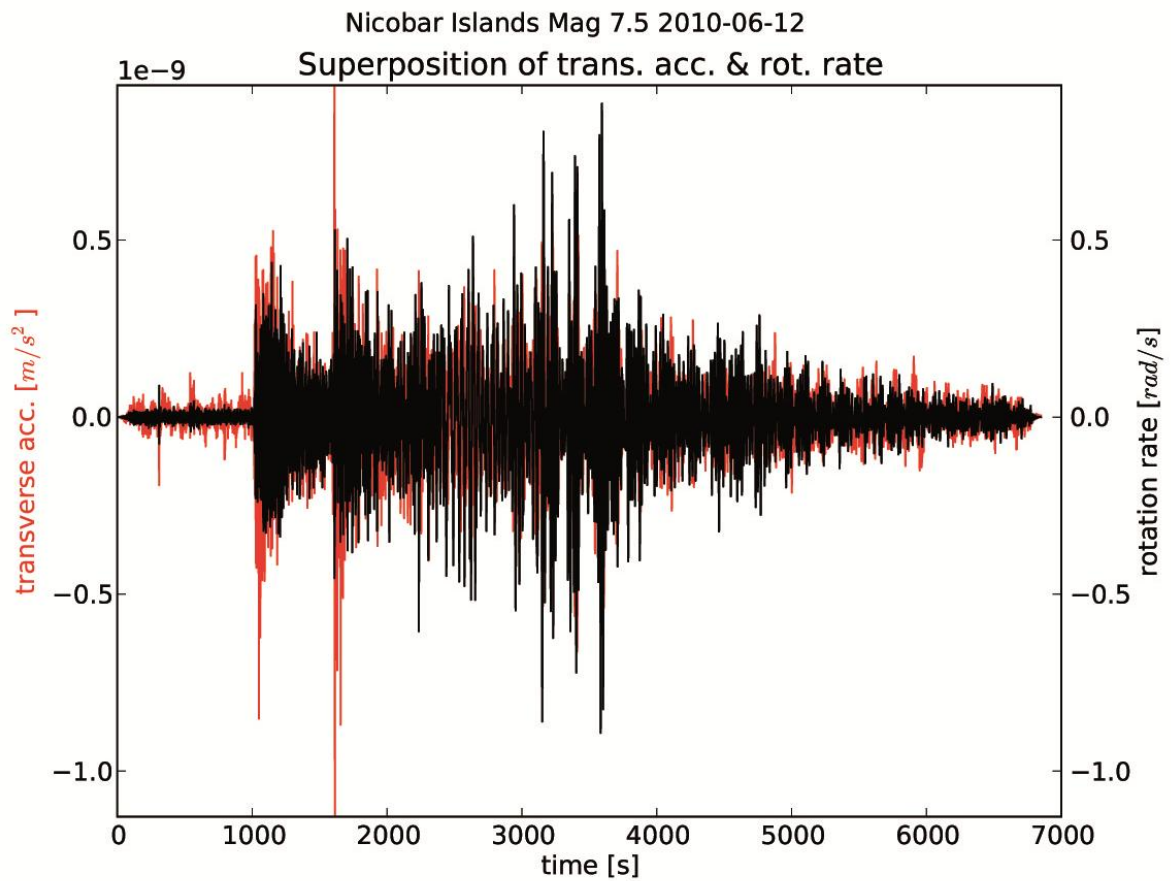


Figure 13 Superposition of TA (red) and RR (black)

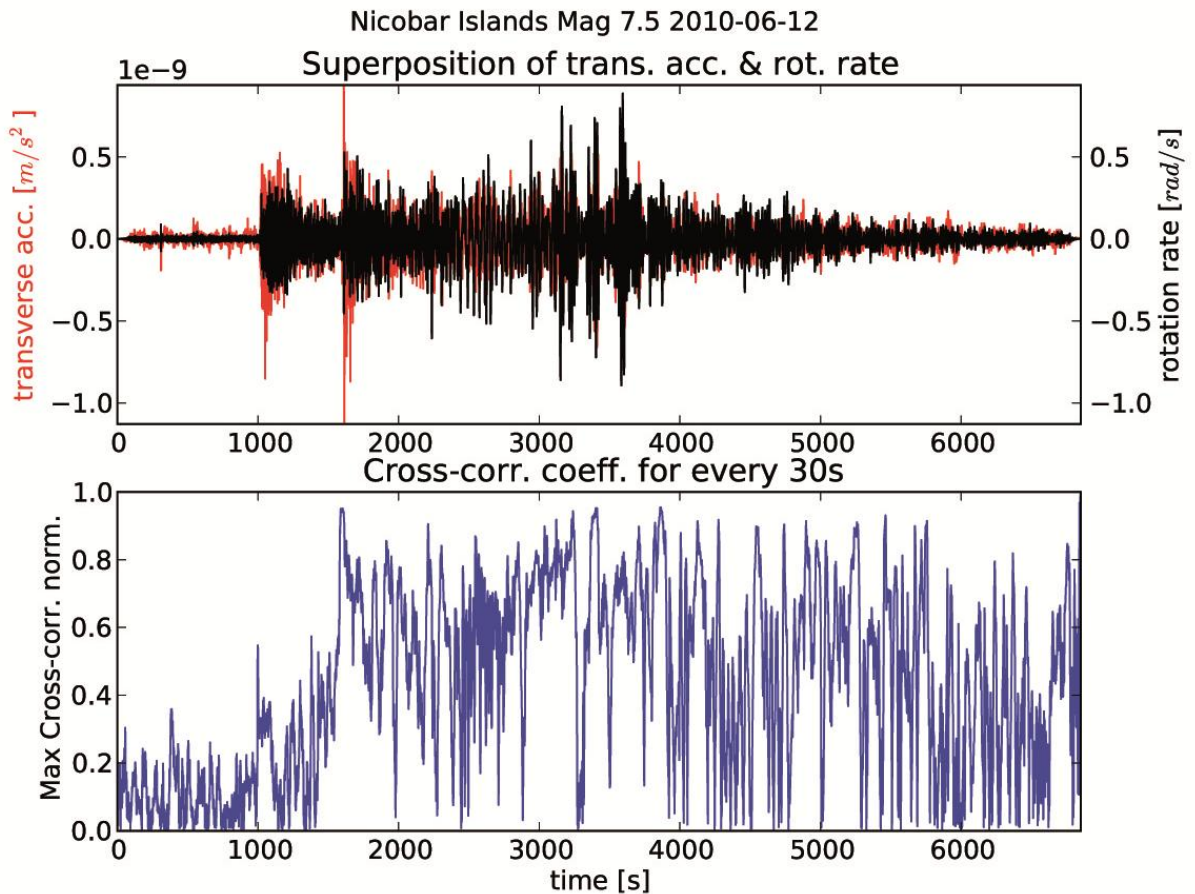


Figure 14 As we see here the correlation coefficient is very high since P-wave arrival. Even if the main phases arrived the coefficient stays high.

5.5. Spectrograms

The periodicities of the time series are displayed by plotting the spectra of both signals. With the spectrograms you cannot only determine the appearing frequencies but also the specific time these frequencies dominate. Although the logarithmic axis of the spectrogram is more significant than the linear axis, it is quite not possible to provide a logarithmic plot because of the problems mentioned in Chapter 4.5. But we still clearly see that frequencies from 0.01Hz – 0.5Hz dominate both spectrograms (Figure 15 and Figure 16). This fact accords to the general assumption of tele-seismic events. (Shearer 2009)

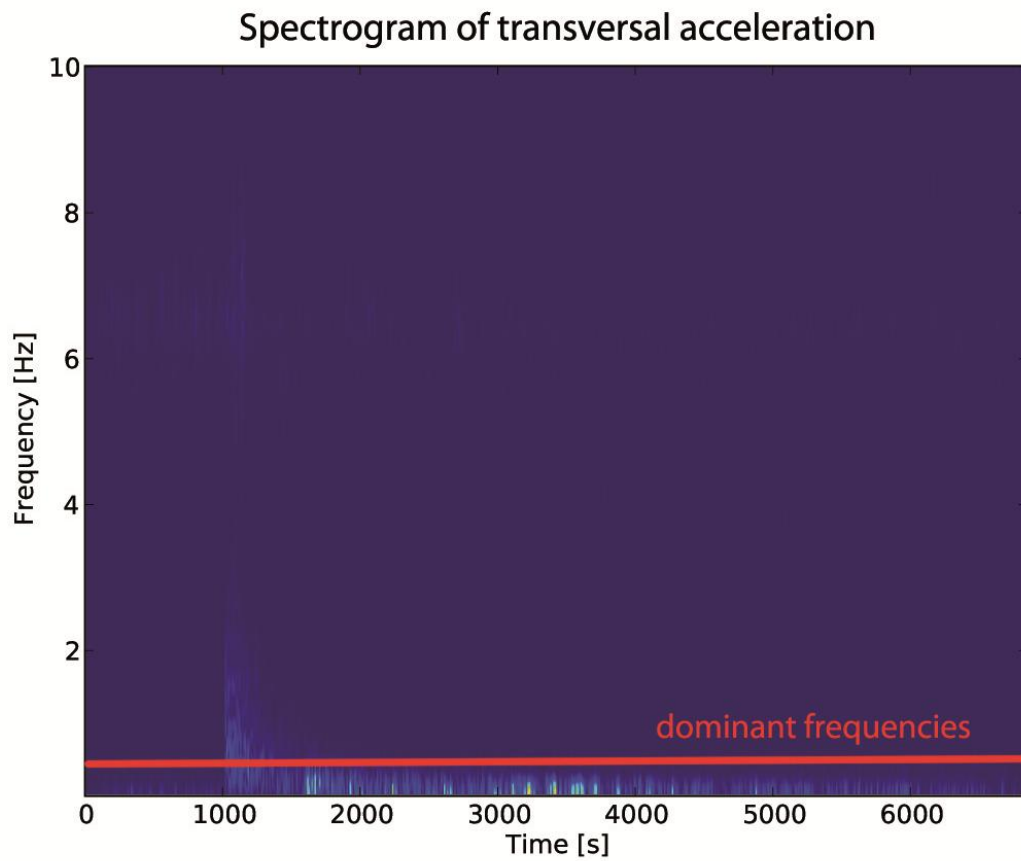


Figure 15 In the spectrogram of the transversal acceleration you can clearly determine the dominant frequencies from 0.01Hz to a maximum of 2Hz around the S wave arrival

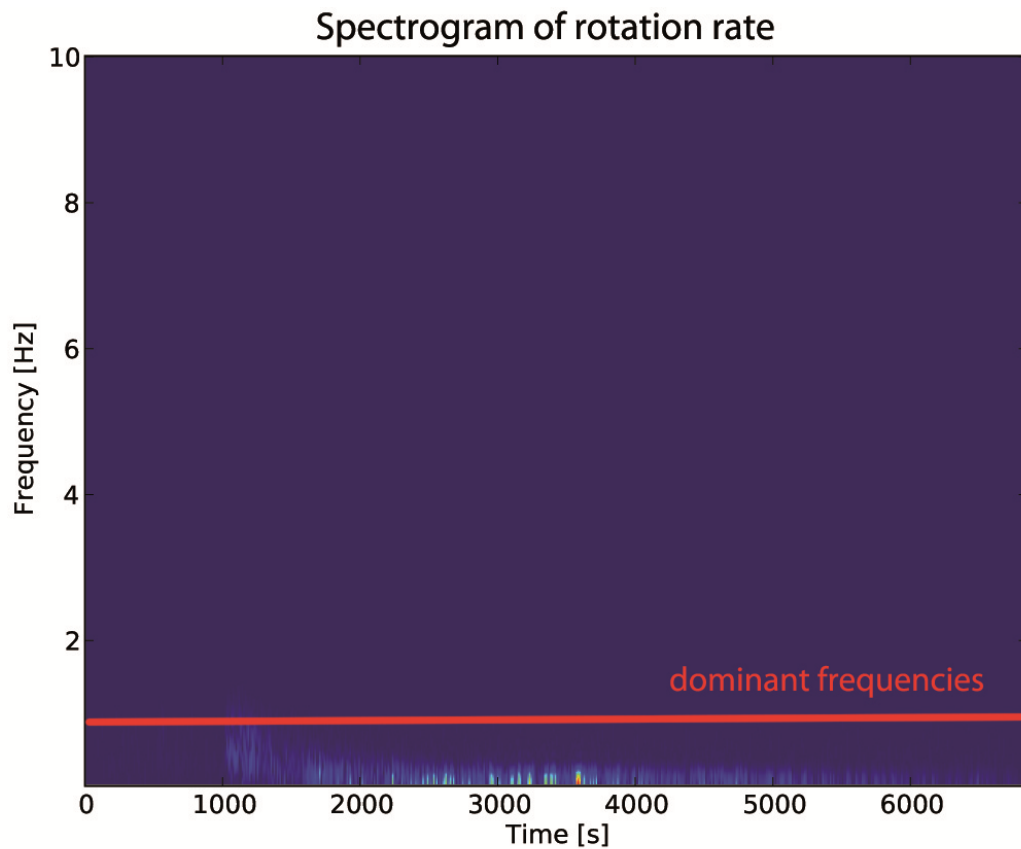


Figure 16 The dominant frequencies of the rotation rate are nearly equal to the frequency spectrum of the transversal acceleration

6. Using the program

After introducing the different outputs the program offers, we would like to test the program of their relevance for real case scenarios. As mentioned several times before, we want the program to provide the user a quick and easy method to analyze seismic events for their coherence between transversal acceleration and rotation rate measured by the Wettzell STS-2 seismometer and Ringlaser. Since seismologist usually are just interested in huge events (because of their better signal-to-noise ratio) with a magnitude greater than 7.0, we need to check the WebDC database or alternatively our email alert-systems mentioned before. Another criterion the dataset needs to comply with is that its source is not located that deep in the earth crust. For now we are only interested in shallow earthquake events not deeper than 70km. Unfortunately the script covers only a timespan from 2010-04-14 to now due to a readjustment of the internal storage structure of the four channels we are interested in.

The Wettzell Ringlaser data had been renamed according to the FDSN (Federation of Digital Broadband Seismographic Networks) rules (Email contact with Wassermann J., 2010) and the old data is actually available, but if the script tries to access the data it ends up with a metadata error. By trying to manually access the data via web query on the WebDC portal, we receive datasets without any content. Thus we need to narrow our search request to a very limited timespan until the problem is fixed.

6.1. Recent past seismic events

Despite our very limited timespan we receive several seismic events that fit our specifications (Table 3). Since we already used the Nicobar Islands event for explaining the operating mode and output of the program, we now want to focus on the seismological work and interpretation of the data. For this purpose we use two additional events. The Vanuatu Islands event located 13.7°S and 166.7°E, and the Irian Jaya Region event located 2.2°S and 136.5°E. Both are tele-seismic events that were recorded by the Wettzell measurement cluster and so there are ideal to analyze.

Origin Time UTC	Mag	Latitude degrees	Longitude degrees	Depth km	Region Name
2010-08-13 21:19:37	7.1	12.5 N	141.5 E	32	South of Mariana Islands
2010-08-10 05:23:46	7.3	17.5 S	168.1 E	35	Vanuatu Islands
2010-07-23 23:15:10	7.4	6.8 N	123.2 E	642	Mindanao, Philippines
2010-07-23 22:51:12	7.4	6.4 N	123.5 E	576	Mindanao, Philippines
2010-07-18 13:35:03	7.1	6.0 S	150.6 E	55	New Britain Region, P.N.G.
2010-07-18 13:04:11	7.1	6.0 S	150.4 E	44	New Britain Region, P.N.G.
2010-06-16 03:16:30	7.2	2.2 S	136.5 E	19	Irian Jaya Region, Indonesia
2010-06-12 19:26:47	7.5	7.7 N	92.0 E	11	Nicobar Islands, India Region
2010-05-27 17:14:48	7.1	13.7 S	166.7 E	49	Vanuatu Islands

Table 3 Red marked events are the ones that fits best to our specifications

The Irian Jaya Region event, now called as IJR event, has an event distance of 12573km and a depth of 19km. Doing the SOAP query we get the last phase arriving after 3256 seconds. The ideal time window (after Eq. 7), dependent on arrival times and distance, we want to look at is about 8080 seconds long.

The second event at Vanuatu Islands (VI) located at 13.7°S and 166.7°E, has an event distance of 15407km and a depth of 49km. The last phase arrives at 3260 seconds, so the ideal time window would be about 8360 seconds.

As most of the plots are not necessary for our analysis of the coherence of transversal acceleration and rotation rate we left them aside. If you are interested in the other outputs please look at the CD attached with this thesis. Every event got its own folder with all data included.

6.2. Observations using the program

As you can see (Figure 17) at first sight, the signal-to-noise ratio of the IJR event is not as nice as the ratio of the Nicobar Islands event. As a result the cross correlation coefficient does not reach the good values of the example event over the whole time window. The peaks around 4000s where also the cross correlation coefficient is very high have to be the arriving love waves. The whole seismogram differs in its characteristics from the Nicobar Islands example. Maybe the event distance is too high for a magnitude 7.2 earthquake to be recorded accurately. But the program operates quite well and fulfills its original purpose to offer a quick overview before looking at the seismogram in detail.

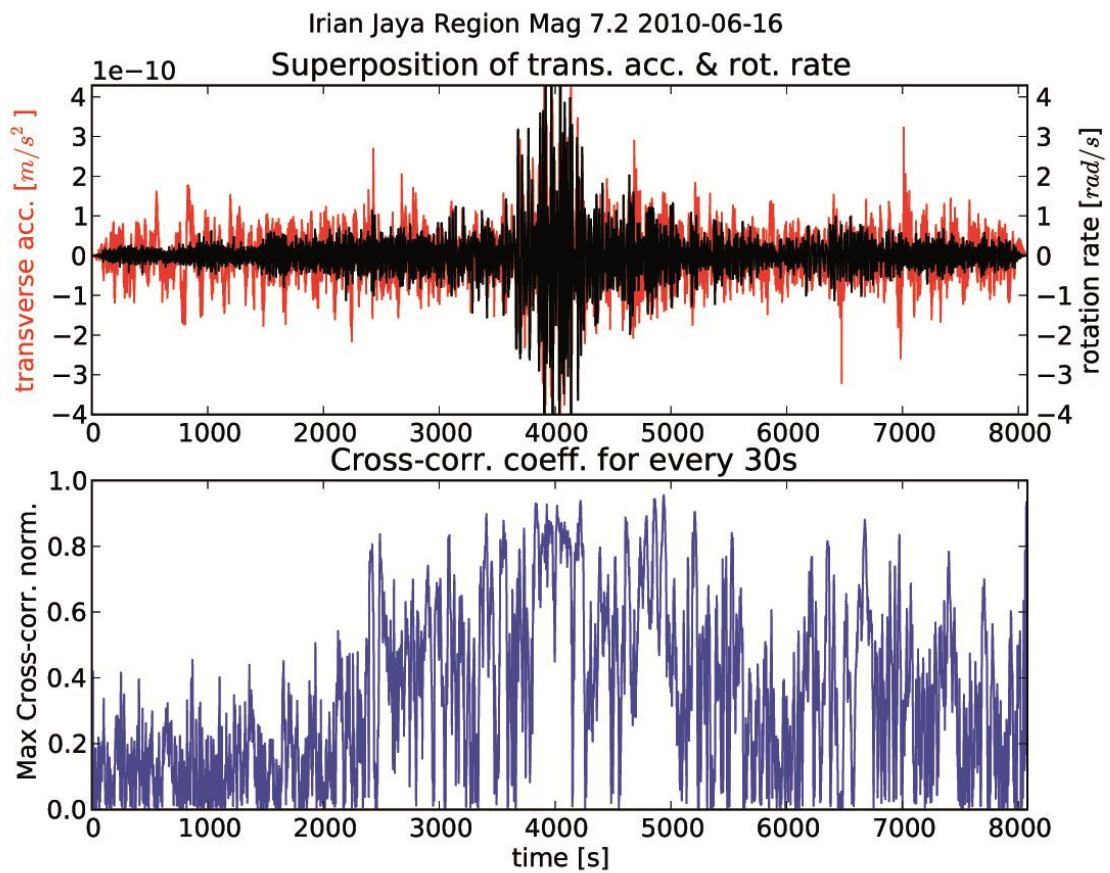


Figure 17 Cross correlation plot of the IJR event

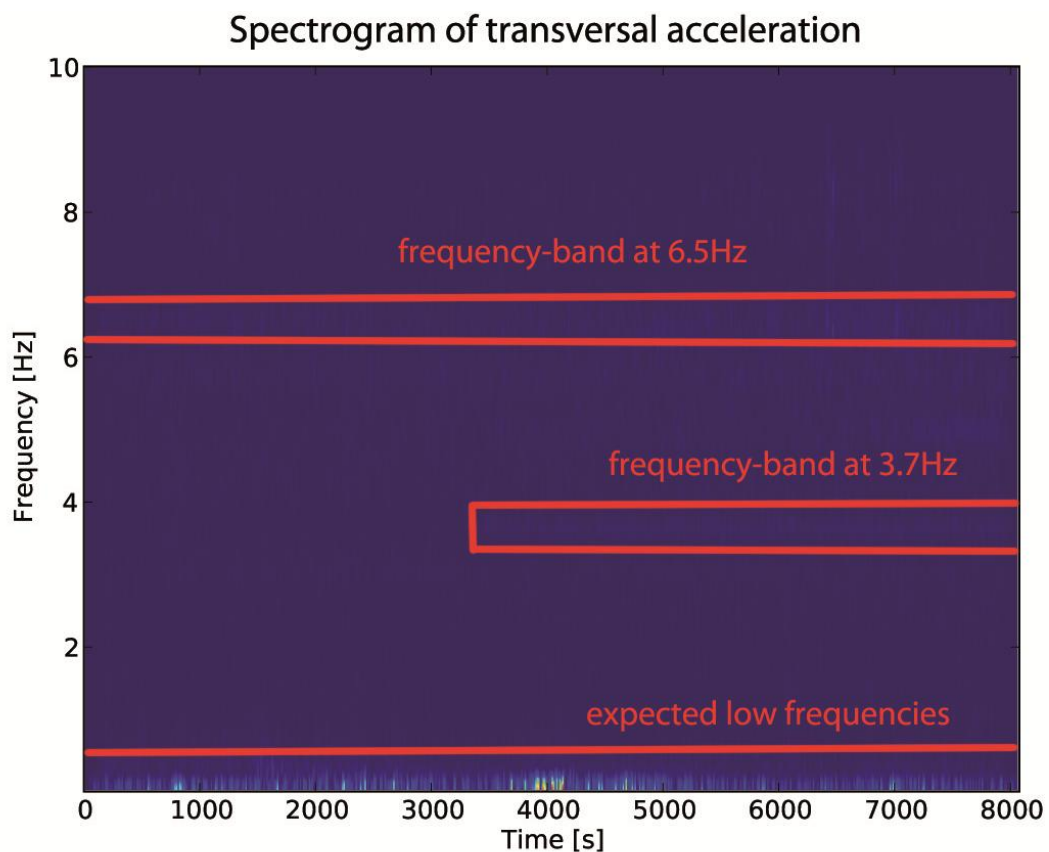


Figure 18 Spectrogram of the transversal acceleration component with interesting frequency-bands at 6.5Hz and 3.7Hz

Another interesting point is the comparison of both spectrograms (Figure 18 and Figure 19). As we see the expected low frequencies on the rotation rate plot, we suddenly get a noisy transversal spectrogram with two dominant frequency-bands at 6.5Hz and 3.7Hz. Almost every single frequency could be observed by zooming into detail. Since only the detection of such characteristics with the developed program should be part of this chapter, we leave it at that.

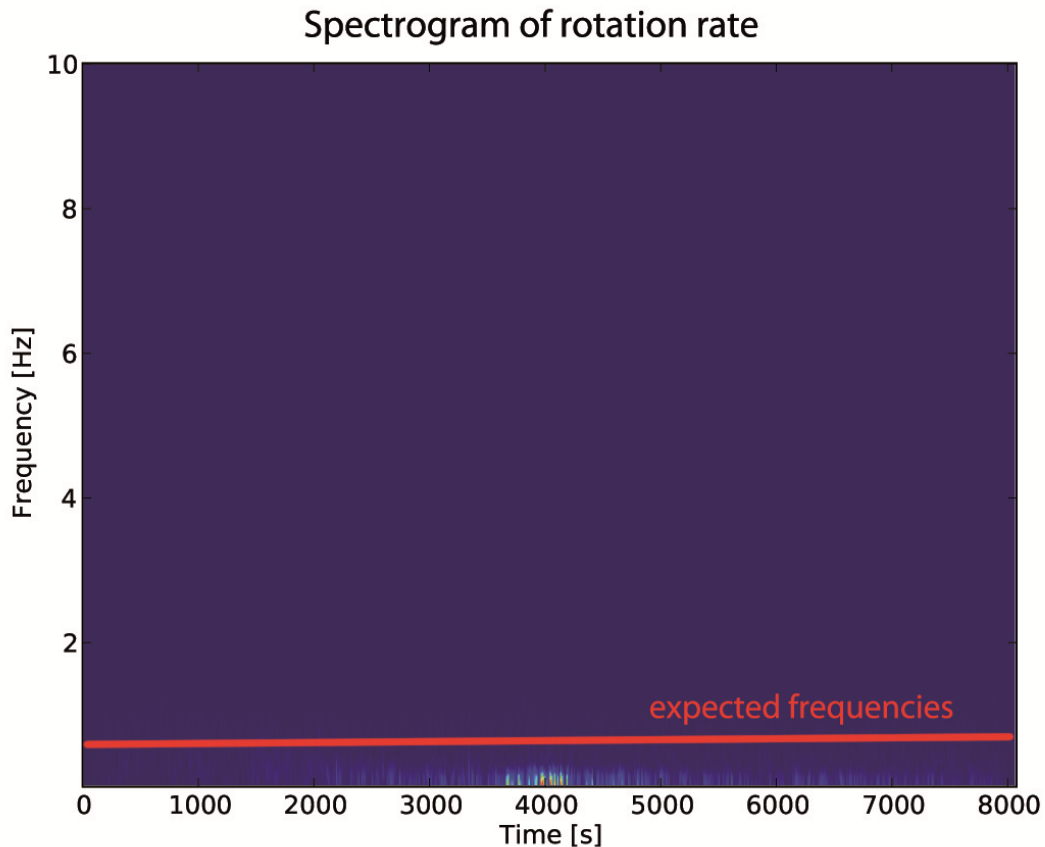


Figure 19 Spectrogram of rotation rate with nothing more than the low frequency-band from 0.01Hz to 0.5Hz

The second event at Vanuata Islands is a very good example for the limits we can be confronted with. While the signal-to-noise ratio of the Ringlaser for the IJR event was that good that you could “see” the seismic event and nearly all their phases we now have difficulties detecting the event. There is not much more than regular noise and the cross correlation values are consequently very bad of course (Figure 20). It seems like the resolution of the Ringlaser is reached although it is able to detect rotation rates as small as $10^{-10} \text{ rad s}^{-1}/\sqrt{\text{Hz}}$. (Igel *et al.*, 2007)

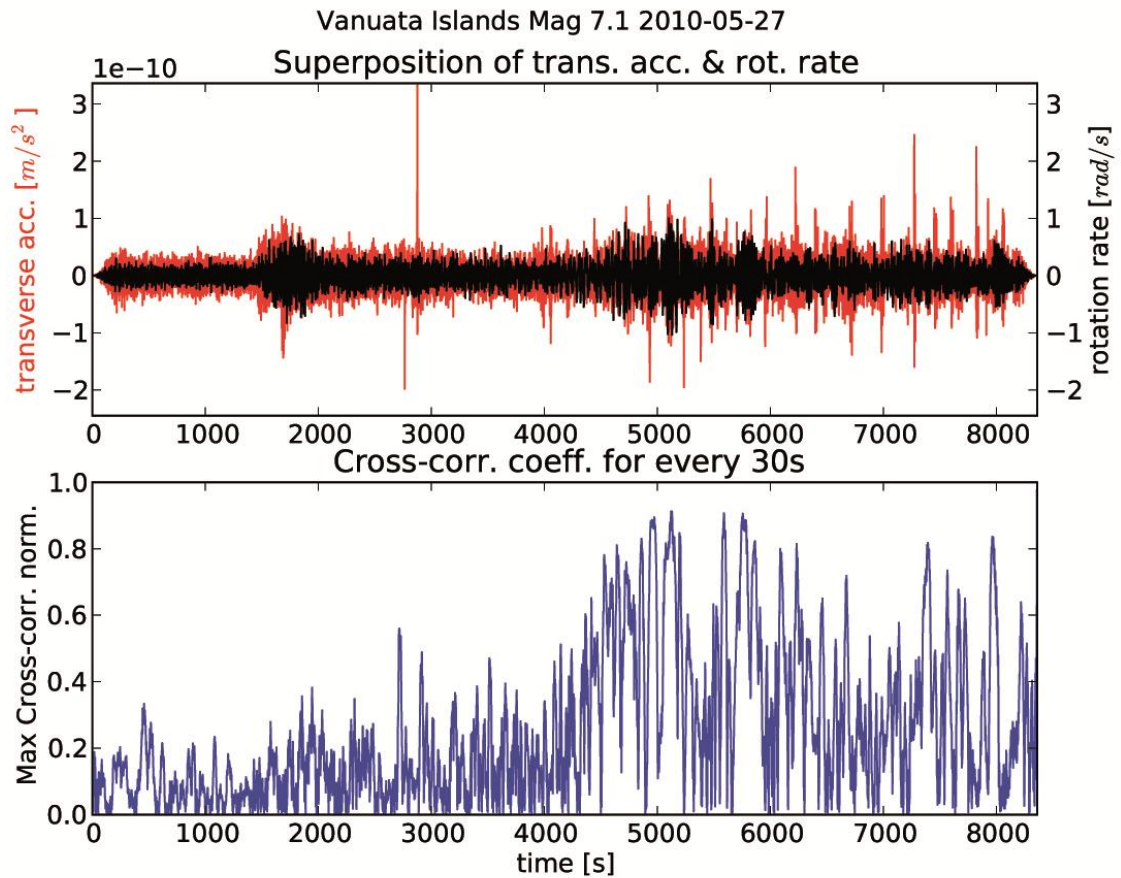


Figure 20 No significant excess value could be retrieved by looking at this noise-like seismogram

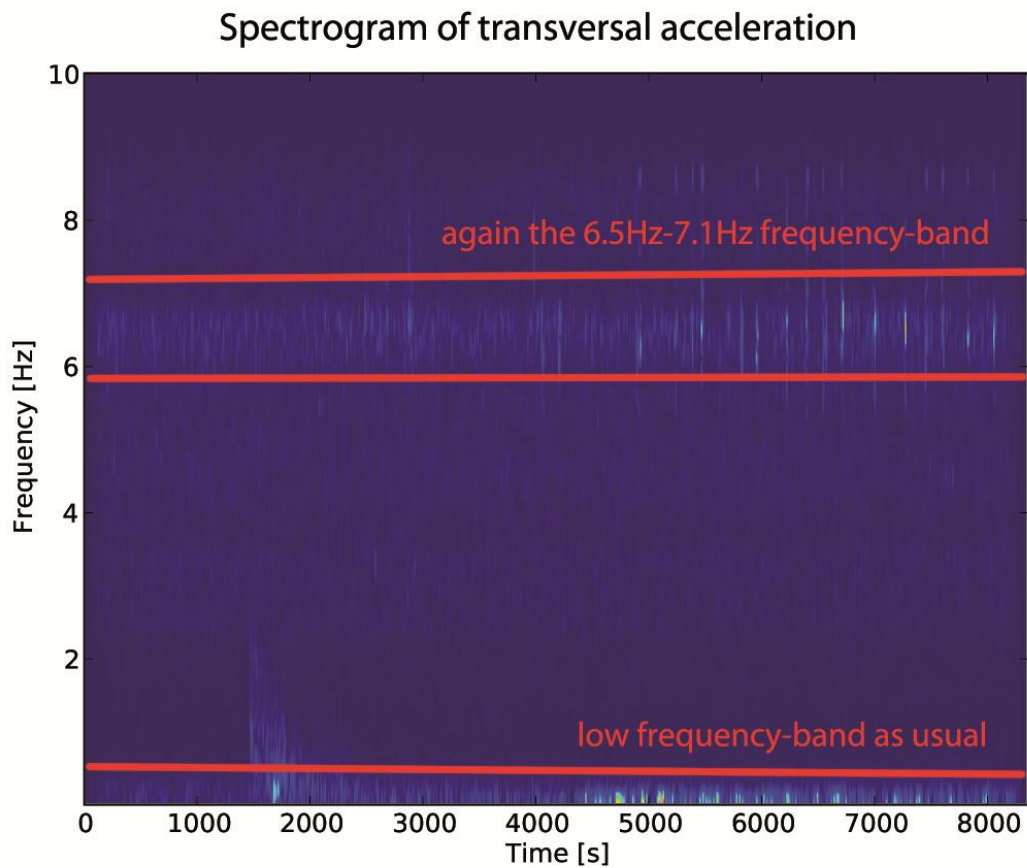


Figure 21 Again we can see the 7Hz frequency-band and the expected tele-seismic low frequency-band

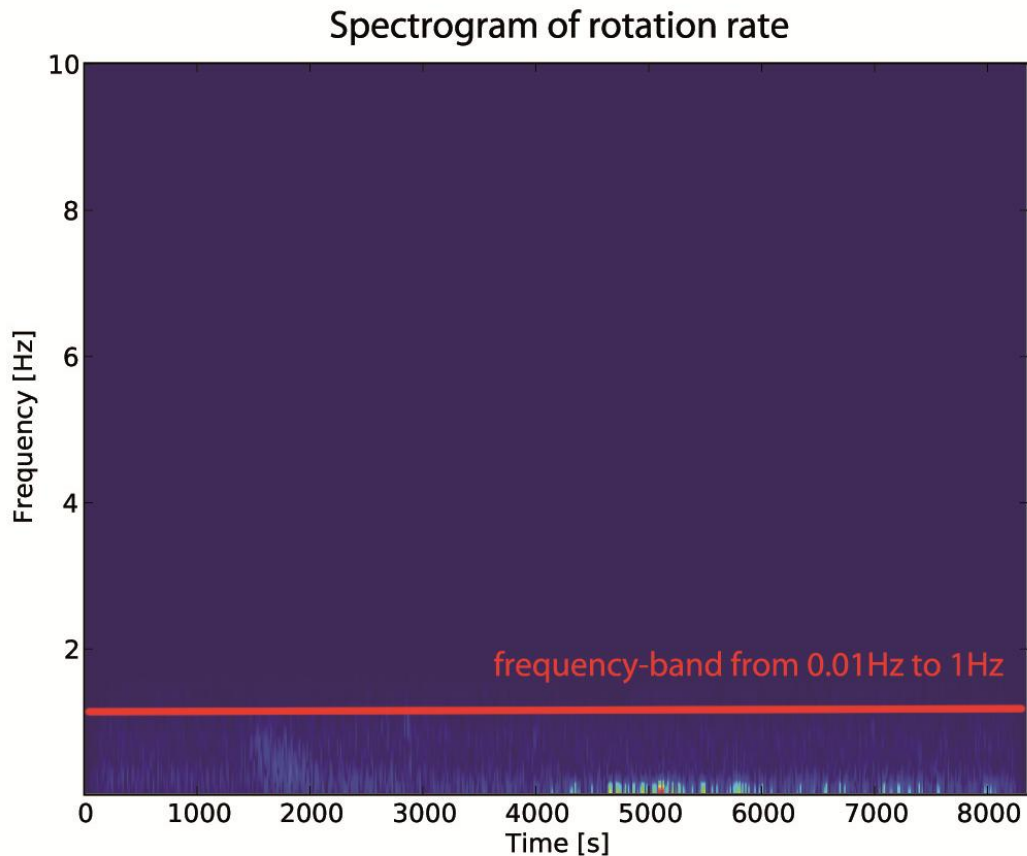


Figure 22 The rotation rate has a frequency range from 0.01Hz to 1Hz

Even if it does not really make sense to look at noisy seismograms in detail, we can at least detect the 7Hz frequency-band in the seismometer recording again (Figure 21 and Figure 22).

These interesting observations could be made within seconds after computing them with the program. Now a seismologist could examine the earthquakes he is really interested in in detail and come to his final conclusion dependent on his previous intension. The VI event for example would be left aside and classified as inoperative, whereas the Nicobar Islands event, our previous example, could be worth a second inspection.

7. Conclusions

In conclusion, the present program has its relevance for the seismological community as it delivers a very fast way to check seismic events regarding their pertinence for the relatively new field of rotational ground motion study.

There are some possible improvements someone could implement of course. So a GUI would simplify the usage of the program and the complicated way of publishing the data over the single outputs is not the ideal. For the last point there is a solution of a script converting the single Postscript files to one PDF (Portable Document Format) file. You could find that script and the dependent Python program pyPDF on the CD. Since maybe a seismologist wants to use a single plot, the output of single Postscript files is not that bad.

During the development and testing of the program we reached the limits of the current ObsPy version and the interaction between a Python script and a data center, in this case operated by ORFEUS (i.e. the problems with the logarithmic view of the spectra and timeout problems during operating with the ArcLink client of ObsPy).

Besides we reached the limits of current Ringlaser technology, as we tested our program with the Vanuata Island event.

This thesis is a good example for the inertial idea behind ObsPy. We developed a program that is based on Python and ObsPy core routines.

Maybe this program will be the base of a greater project for further development initiated by the rotational seismology community.

8. Acknowledgments

This work was supported by the Department of Earth and Environmental Sciences and its employees. A special thank appertains to the ObsPy development team, that exists of Dr. Robert Barsch, Moritz Beyreuther, Tobias Mergies and Lion Krischer. If I had problems during the development they helped me out with advices and their experience.

Furthermore I am grateful for my supervisors Prof. Dr. Heiner Igel and Dr. Joachim Wassermann for their advices and in special Prof. Dr. Igel for the issue of the bachelor thesis.

A further acknowledgment goes to Hanna Ziegowski who gave me advices regarding the phrase of some topics. I also want to thank her for the review of my thesis.

In conclusion, I also want to thank every single developer of the tools, programs and scripts I used during the development of this thesis. The development of my program is applicable based on excellent programming skills by all the developers of these programs.

9. References

Books

Gubbins, David (2008): Time Series Analysis and Inverse Theory for Geophysicists. 3. Edition.: Cambridge University Press.

Shearer, Peter (2009): Introduction to Seismology: Cambridge University Press.

Tosi, Sandro (2009): Matplotlib for Python developers. Build remarkable publication quality plots the easy way. Birmingham: Packt Publ.

Weigend, Michael (2008): Python ge-packt. [schneller Zugriff auf Module, Klassen und Funktionen ; Tkinter, Datenbanken und Internet-Programmierung ; für die Versionen Python 3.0 und 2.x]. 4., updated edition. Heidelberg: mitp.

Manuscripts

Kurrle, D. Ferreira A. Igel H. J. Wassermann U. Schreiber (2010): First observations of Earth's free oscillations on a ring laser system. Unpublished manuscript, 2010.

Beyreuther, M.; Barsch R.; Krischer L.; Megies T.; Behr Y.; Wassermann J. (2010): ObsPy: A Python Toolbox for Seismology. Unpublished manuscript, 2010.

Articles

Bendat, J. S. (1982): Random Data - Analysis and measurement procedures . In: Current Contents/Engineering Technology & Applied Sciences, H. 25, S. 16.

Buland, R., and C. H. Chapman (1983). The computation of seismic travel times. In: Bulletin of the Seismological Society of America Vol. 73, p. 1.271-1302

Crotwell, H. P. T. J. Owens and J. Ritsema (1999): The TauP Toolkit: Flexible seismic travel-time and ray-path utilities. In: Seismological Research Letters, H. Vol 70, S. 154–160.

Fang K; Brown R. J. (1996): A new algorithm for the rotation of horizontal components of shear-wave seismic data. In: CREWES Research Report, H. Vol 8, p. 12-1 - 12-14.

Igel, H.; Schreiber, U.; Flaws, A.; Schuberth, B.; Velikoseltsev, A.; Cochard, A. (2005): Rotational motions induced by the M8.1 Tokachi-oki earthquake, September 25, 2003. In: Geophysical Research Letters, Jg. 32, H. 8.

Igel, Heiner; Cochard, Alain; Wassermann, Joachim; Flaws, Asher; Schreiber, Ulrich; Velikosevtsev, Alex; Dinh, Nguyen Pham (2007): Broad-band observations of earthquake-induced rotational ground motions. In: *Geophysical Journal International*, Jg. 168, H. 1, p. 182–196.

Kurrle, D.; Igel, H.; Ferreira, A. M.G.; Wassermann, J.; Schreiber, U. (2010): Can we estimate local Love wave dispersion properties from collocated amplitude measurements of translations and rotations? In: *Geophysical Research Letters*, Jg. 37.

Lee, W. H.K. (2009): A Glossary for Rotational Seismology. In: *Bulletin Of The Seismological Society Of America*, Jg. 99, H. 2B, p. 1082–1090.

Welch, P. D. (1967): The Use of Fast Fourier Transform for the Estimation of Power Spectra: A Method Based on Time Averaging Over Short, Modified Periodograms. In: *IEEE Transactions on Audio Electroacoustics*, H. Vol. AU-15, p. 70–73.

Internet sources

Article: Korrelationskoeffizient. In: Wikipedia, Die freie Enzyklopädie (2010). URL: <http://de.wikipedia.org/wiki/Korrelationskoeffizient>. Last visited: 2010-08-21 15:34 UTC.

Article: Vincenty's formulae. In: Wikipedia, The Free Encyclopedia (2010). URL: http://en.wikipedia.org/wiki/Vincenty's_formulae/. Last visited 2010-08-21 15:37 UTC.

Documentation: Python v2.6.5 Documentation (2010). URL: <http://docs.python.org/release/2.6.5/>. Last updated 2010-03-19. Last visited: 2010-08-17 17:54 UTC.

Documentation: Numpy and Scipy Documentation (2010). URL: <http://docs.scipy.org/doc/>. Last visited 2010-08-20 15:12 UTC.

Documentation: Python framework for processing seismological data (2010). URL: <http://svn.geophysik.uni-muenchen.de/trac/obspy/wiki/>. Last visit 2010-08-21 19:57 UTC

Documentation: Matplotlib User's Guide (2010). URL: <http://matplotlib.sourceforge.net/users/index.html>. Last visited 2010-08-12 12:04 UTC.

Manual: Portable Very-Broad-Band-Tri-Axial- Seismometer – STS-2 Manual (1995). URL: http://www.passcal.nmt.edu/webfm_send/488. Last visited: 2010-07-15 13:43 UTC.

Manual: Geocentric Datum of Australia Technical Manual (2009) URL: <http://www.icsm.gov.au/gda/gdatm/gdav2.3.pdf>. Last visited 2010-08-15 12:23 UTC.

Source: TauP web service (2010). URL: <http://www.orfeus-eu.org/wSDL/taup/taup.wSDL>. Last visited 2010-08-21 19:57 UTC.

Appendix

A. Source Code of the program

```
#Fix if you want to execute the script on Intranet machines

#import sys

#sys.path.append("/home/SOFTWARE/OBS/obspy/obspy/branches/symlink/")

from obspy.core import UTCDateTime

from obspy.arlink.client import Client

import numpy as np

import matplotlib.pyplot as plt

from obspy.signal import cornFreq2Paz, seisSim, gps2DistAzimuth, rotate_NE_RT, low-pass, bandpass

from obspy.signal.util import xcorr

from obspy.imaging.spectrogram import spectrogram

print "Setting global event variables"

#Setting important variables from user

#path = raw_input("Enter path to save data: ")

event = raw_input("Enter event name: ")

UTCTime = UTCDateTime(raw_input("Enter event time (no data before 2010-04-14): ").strip())

depth = float(input("Enter event depth: "))

lat2 = float(input("Enter event latitude: "))
```

```

#Check lat2 input
if lat2 > 90 or lat2 < -90:
    msg = "Latitude out of bounds! (-90 <= lat2 <=90)"
    raise ValueError(msg)
lon2 = float(input("Enter event longitude: "))

#Just a visual feedback for inputed data
#print "Path to save data: " , path
#print "Event name: " , event
print "Event time: " , UTCTime
print "Coordinates of event: " , lat2, lon2

#Computes backward azimuths between source and station
ba = gps2DistAzimuth(49.144, 12.878, lat2, lon2)

#Extracts distance parameter and converts to km
dist = float(ba[0]*1e-3)

print "Event distance (in km) from Wettzell, Germany: ", round(dist, 1)

#Creating dir with given path
#import os
#os.mkdir(path)

#Getting Data from ArcLink
client = Client('erde.geophysik.uni-muenchen.de', 18001)
#client = Client_Seishub()

```

```

t=UTCTime

#import pdb;pdb.set_trace()

#Calculation of time window with fixed S'S'df phase
twin=(3260*2)+dist/10

#Getting all four components and save them to disk as MSED for further actions
print "retrieving waveforms..."

bhn = client.getWaveform("BW", "WETR", "", "BHN", t - 300, t + twin)
bhe = client.getWaveform("BW", "WETR", "", "BHE", t - 300, t + twin)
bhz = client.getWaveform("BW", "WETR", "", "BHZ", t - 300, t + twin)
bjz = client.getWaveform("BW", "RLAS", "", "BJZ", t - 300, t + twin)
print "done"

#Setting path to created folder
#os.chdir(path)

print "writing MSED files ..."
bhn.write('BW.WETR.BHN.mseed', format = 'MSED')
bhe.write('BW.WETR.BHE.mseed', format = 'MSED')
bhz.write('BW.WETR.BHZ.mseed', format = 'MSED')
bjz.write('BW.RLAS.BJZ.mseed', format = 'MSED')
print "done"

#Adding all traces to one stream and plot 4C-Plot
fc = bhn + bhe + bhz + bjz
fc.plot(outfile = '4C-plot.ps')

```



```
print "unrotated 4C plot done"
```

```
#Retrieve PAZ for all STS-2 channels
```

```
print "retrieving PAZ..."
```

```
pazbhn = client.getPAZ("BW", "WETR", "", "BHN", t, t)
```

```
pazbhe = client.getPAZ("BW", "WETR", "", "BHE", t, t)
```

```
pazbhz = client.getPAZ("BW", "WETR", "", "BHZ", t, t)
```

```
pazbjz = client.getPAZ("BW", "RLAS", "", "BJZ", t, t)
```

```
print "done"
```

```
pazbhn = pazbhn.values()[0]
```

```
pazbhe = pazbhe.values()[0]
```

```
pazbhz = pazbhz.values()[0]
```

```
pazbjz = pazbjz.values()[0]
```

```
print "instrument correction ..."
```

```
#Subtracting mean from time series
```

```
bhn[0].data = bhn[0].data - bhn[0].data.mean()
```

```
bhe[0].data = bhe[0].data - bhe[0].data.mean()
```

```
bhz[0].data = bhz[0].data - bhz[0].data.mean()
```

```
bjz[0].data = bjz[0].data - bjz[0].data.mean()
```

```
#Apply bandpass from 0.01-40Hz
```

```
resbhn = bandpass(bhn[0], 0.005, 40, corners=4)
```

```
resbhe = bandpass(bhe[0], 0.005, 40, corners=4)
```

```
resbhz = bandpass(bhz[0], 0.005, 40, corners=4)
```

```
resbjz = bandpass(bjz[0], 0.005, 40, corners=4)
```

#0.005Hz instrument

```
one_hertz = cornFreq2Paz(.005)
```

#Correct for frequency response of the instrument

```
resbhn = seisSim(bhn[0].data.astype('float64'), bhn[0].stats.sampling_rate,
                 pazbhn, inst_sim=one_hertz)
```

```
resbhe = seisSim(bhe[0].data.astype('float64'), bhe[0].stats.sampling_rate,
                 pazbhe, inst_sim=one_hertz)
```

```
resbhZ = seisSim(bhz[0].data.astype('float64'), bhz[0].stats.sampling_rate,
                 pazbhZ, inst_sim=one_hertz)
```

```
resbjz = seisSim(bjz[0].data.astype('float64'), bjz[0].stats.sampling_rate,
                 pazbjz, inst_sim=one_hertz,
                 no_inverse_filtering=True) #important because of empty PAZ, so just
resampling data
```

#Correct for overall sensitivity, nm/s

```
resbhn = (resbhn/1e4)/pazbhn['sensitivity']#resbhn *= 1e9/pazbhn['sensitivity'] #
```

```
resbhe = (resbhe/1e4)/pazbhe['sensitivity']#resbhe *= 1e9/pazbhe['sensitivity'] #
```

```
resbhZ = (resbhZ/1e4)/pazbhZ['sensitivity']#resbhZ *= 1e9/pazbhZ['sensitivity'] #
```

```
resbjz = (resbjz/1e8)/pazbjz['sensitivity']#resbjz *= 1e9/pazbjz['sensitivity'] #
```

#Plot the seismograms (original and corrected data)

```
secbhn = np.arange(len(resbhn))/bhn[0].stats.sampling_rate
```

```
secbhe = np.arange(len(resbhe))/bhe[0].stats.sampling_rate
```

```
secbhZ = np.arange(len(resbhZ))/bhZ[0].stats.sampling_rate
```

```
secbjz = np.arange(len(resbjz))/bjz[0].stats.sampling_rate
```

```

plt.figure()
plt.subplot(211)
plt.plot(secbhn,bhn[0].data, 'k')
plt.title("WETR.BHN instrument correction")
plt.ylabel('STS-2')
plt.subplot(212)
plt.plot(secbhn,resbhn, 'k')
plt.xlabel('time [s]')
plt.ylabel('0.005 Hz Corner Frequency')
plt.suptitle('%s' % (event))
plt.savefig('BHN_instrument-correction.ps')

```

```

plt.figure()
plt.subplot(211)
plt.plot(secbhe,bhe[0].data, 'k')
plt.title("WETR.BHE instrument correction")
plt.ylabel('STS-2')
plt.subplot(212)
plt.plot(secbhe,resbhe, 'k')
plt.xlabel('time [s]')
plt.ylabel('0.005 Hz Corner Frequency')
plt.suptitle('%s' % (event))
plt.savefig('BHE_instrument-correction.ps')

```

```

plt.figure()
plt.subplot(211)
plt.plot(secbhz,bhz[0], 'k')

```

```

plt.title("WETR.BHZ instrument correction")
plt.ylabel('STS-2')
plt.subplot(212)
plt.plot(secbhz,resbhz, 'k')
plt.xlabel('time [s]')
plt.ylabel('0.005 Hz Corner Frequency')
plt.suptitle('%s' % (event))
plt.savefig('BHZ_instrument-correction.ps')

```

```

plt.figure()
plt.subplot(211)
plt.plot(secbjz,bjz[0].data, 'k')
plt.title("RLAS.BJZ instrument correction")
plt.ylabel('RLAS')
plt.subplot(212)
plt.plot(secbjz,resbjz, 'k')
plt.xlabel('time [s]')
plt.ylabel('0.005 Hz Corner Frequency')
plt.suptitle('%s' % (event))
plt.savefig('BJZ_instrument-correction.ps')

```

```

print "instrument response corrected plots done"

```

```

####delete to free memory####

```

```

del bhn, bhe, bhz, bjz, resbhz

```

```

resbhe.resize((len(resbhn), ), refcheck=False)

```

```
#Rotate horizontal component and superposition of rotation rate & transversal acceleration
```

```
rot = rotate_NE_RT(resbhn, resbhe, ba[1])
```

```
###delete to free memory###
```

```
del resbhn, resbhe
```

```
#Differentiate rotated horizontal component to retrieve trans. acc.
```

```
ta = np.diff(rot[1], n=1, axis=-1)
```

```
#Fix for the amplitude axis for perfect superposition of both signals
```

```
ta *= -1e1
```

```
#trans. acc. & rotation rate plot
```

```
secta = np.arange(len(ta))/20
```

```
#superposition plot of rot. rate & trans. acc.
```

```
fig = plt.figure()
```

```
ax1 = fig.add_subplot(111)
```

```
ax1.plot(secta, ta, 'r')
```

```
ax1.set_ylim(ymin=min(ta),ymax=max(ta))
```

```
ax1.set_title('Superposition of trans. acc. & rot. rate')
```

```
ax1.set_ylabel('transverse acc. [ $\text{m/s}^2$ ]', color='red')
```

```
ax1.set_xlabel('time [s]')
```

```
ax2 = ax1.twinx()
```

```
ax2.plot(secbjz, resbjz, 'k')
```

```
ax2.set_ylim(ymin=min(ta),ymax=max(ta))
```

```

ax2.set_ylabel('rotation rate [$rad/s$])
ax2.set_xlabel('time [s]')
plt.suptitle('%s' % (event))
plt.savefig('superposition_TA_RR.ps')

#trans. acc. & rotation rate plot
plt.figure()
plt.subplot(211)
plt.plot(secta, ta, 'r')
plt.title("transverse acceleration & rotation rate" % (t))
plt.ylabel("transverse acc. [$m/s^2$]")
plt.xlabel("Time [s]")
plt.subplot(212)
plt.plot(secbjz, resbjz, 'k')
plt.xlabel("time [s]")
plt.ylabel("rotation rate [$rad/s$]")
plt.suptitle('%s' % (event))
plt.savefig('vs_TA_RR.ps')

print "trans. acc. vs rot. rate plot done"

#Spectrograms of rot. rate & trans. acc.

#Spectrogram of rot. rate
#fig = plt.figure()
#ax1 = fig.add_axes([0.1, 0.75, 0.7, 0.2])
#ax1.plot(secbjz, resbjz, 'k')

```

```

#ax1.set_title('Spectrogram of rotational component')

#ax1.set_ylabel('nm/s/Hz')

#ax2 = fig.add_axes([0.1, 0.1, 0.7, 0.60])

#ax3 = fig.add_axes([0.83,0.1,0.03,0.6])

#ax, spec = spectrogram(resbjz, 20, mult=8.0, log=True, axis=ax2)

#ax1.set_xlim(secbjz[0], secbjz[-1])

#ax2.set_xlim(secbjz[0], secbjz[-1])

#ax3.set_xlabel('Sqrt')

#fig.colorbar(spec, cax=ax3)

#plt.savefig(outfile='SpectrogramRR.ps',format="ps")

#plt.show()

```

```

#Spectrogram of trans. acc.

#fig = plt.figure()

#ax1 = fig.add_axes([0.1, 0.75, 0.7, 0.2])

#ax1.plot(secta, ta,'k')

#ax1.set_title('Spectrogram of transversal component')

#ax1.set_ylabel('nm/s/Hz')

#ax2 = fig.add_axes([0.1, 0.1, 0.7, 0.60])

#ax3 = fig.add_axes([0.83,0.1,0.03,0.6])

#ax, spec = spectrogram(ta, 20, mult=8.0, log=True, axis=ax2)

#ax1.set_xlim(secta[0], secta[-1])

#ax2.set_xlim(secta[0], secta[-1])

#ax3.set_xlabel('Sqrt')

#fig.colorbar(spec, cax=ax3)

#plt.savefig(outfile='SpectrogramTA.ps',format="ps")

```

```

#cross correlation coefficient of rot. rate & trans. acc.

#a, b = xcorr(ta, resbjz, dist)

#print 'Max correlation coefficient: ', round(b, 2)


#Resizing ndarray resbjz to correlate against ta
xresbjz = resbjz.copy()
xresbjz.resize((len(ta), ), refcheck=False)


#### Delete to free memory ####
del resbjz


#cross correlation along time axis
#discrete, linear correlation of two 1-dim sequences


i=0
k=0

shiftlen = 20

winlen = 30*20 #samples weil direkt auf array (30s * 20Hz)
xcorr_array = np.empty(len(ta)/shiftlen)


while True:
    if i + shiftlen > len(xresbjz):
        break

    window1 = xresbjz[i : i + winlen]
    window2 = ta[i : i + winlen]

    xcorr_array[k] = np.corrcoef(window1,window2)[1,0]

```



```

k=k+1

i=i + shiftlen

#setting new y range for plot
xsecbjz = np.arange(len(xresbjz))/20

#plotting the figure
fig = plt.figure()
ax1 = fig.add_subplot(211)
ax1.plot(secta, ta, 'r')
ax1.set_ylim(ymin=min(ta),ymax=max(ta))
ax1.set_title('Superposition of trans. acc. & rot. rate')
ax1.set_ylabel('transverse acc. [ $\text{m/s}^2$ ]', color='red')
ax2 = ax1.twinx()
ax2.plot(xsecbjz, xresbjz, 'k')
ax2.set_ylim(ymin=min(ta),ymax=max(ta))
ax2.set_ylabel('rotation rate [ $\text{rad/s}$ ']')
ax2.set_xlabel('time [s]')
ax3 = fig.add_subplot(212, sharex=ax1)
ax3.plot(abs(xcorr_array))
ax3.set_title('Cross-corr. coeff. for every 30s')
ax3.set_ylabel('Max Cross-corr. norm.')
ax3.set_xlabel('time [s]')
ax3.set_xlim((0,len(xcorr_array)))
plt.suptitle('%s' % (event))
plt.savefig('crosscorr_ta-rr.ps')
print "trans_acc_rot_rat_vs_cc_norm plot done"

```

```

print "spectrogram plot of TA ..."
spectrogram(ta, 20, log=False, outfile='SpectrogramTA.ps', format='PS', mult=8.0)
print "spectrogram plot done"
print "spectrogram plot of RR ..."
spectrogram(xresbjz, 20, log=False, outfile='SpectrogramRR.ps', format='PS', mult=8.0)
print "spectrogram plots done"

```

B. Source code of ps2pdf script

```

from pyPdf import PdfFileWriter, PdfFileReader

output = PdfFileWriter()

input1 = PdfFileReader(file("4C-plot.ps", "rb"))
input2 = PdfFileReader(file("BHN_instrument-correction.ps", "rb"))
input3 = PdfFileReader(file("BHE_instrument-correction.ps", "rb"))
input4 = PdfFileReader(file("BHZ_instrument-correction.ps", "rb"))
input5 = PdfFileReader(file("BJZ_instrument-correction.ps", "rb"))
input6 = PdfFileReader(file("vs_TA_RR.ps", "rb"))
input7 = PdfFileReader(file("superposition_TA_RR.ps", "rb"))
input8 = PdfFileReader(file("crosscorr_ta-rr.ps", "rb"))
input9 = PdfFileReader(file("SpectrogramTA.ps", "rb"))
input10 = PdfFileReader(file("SpectrogramRR.ps", "rb"))

# add all pages from inputs to output document, unchanged
output.addPage(input1.getPage(0))
output.addPage(input2.getPage(1))
output.addPage(input3.getPage(2))

```

```
output.addPage(input4.getPage(3))  
output.addPage(input5.getPage(4))  
output.addPage(input6.getPage(5))  
output.addPage(input7.getPage(6))  
output.addPage(input8.getPage(7))  
output.addPage(input9.getPage(8))  
output.addPage(input10.getPage(9))
```

```
# print how many pages the document has:
```

```
print "document1.pdf has %s pages." % input1.getNumPages()
```

```
# finally, write "output" to event.pdf
```

```
outputStream = file("event.pdf", "wb")
```

```
output.write(outputStream)
```

```
outputStream.close()
```