AMATH 582 Final Project

Christopher Liu

March 19, 2020

Abstract

Recent deployments of ocean-bottom sensors provide us with new opportunities to study the characteristics of tsunamis. One such dataset that is now available from ocean-bottom sensors is pressure data. From studying the changes in pressures at the ocean-bottom during tsunami events, we hope that we can better characterize tsunamis when they are close to their source. Through the use of time-frequency analysis and windowing, we will attempt to observe hydroacoustic resonance and changes in seafloor geometry before and after a tsunami event. We will show that while this seems to work for data collected near the tsunami source, more work will need to be done to extend this to data from distant sources and in different geologic settings.

1 Introduction and Overview

Tsunamis are large waves generated typically by seismic events such as earthquakes that are capable of causing catastrophic damage to coastal communities.. As as result, much effort has been put into trying to understand tsunami phenomenon by the greater scientific community. In recent years, networks of ocean-bottom sensors have been able to measure data directly at the tsunami sources. Up until then, the scientific community lacked direct measurements and relied on measurements made by remote sensors [4]. Pressure data measured by gauges in these ocean-bottom sensor networks provide a unique opportunity to study study the characteristics of tsunamis and their corresponding earthquakes. In a paper published by Bolshakova et al. in 2011, they discovered a unique phenomenon in a tsunami source through time-frequency analysis of the pressure signal which they named hydroacoustic resonance[1]. Furthermore, by looking at changes in the static pressure before and after a tsunami event, we can also determine changes in the water depth after an earthquake event.

We will be applying continuous wavelet transforms (CWTs) and windowing to data measured by ocean-bottom pressure gauges from 2 events, the 2003 Tokachi-oki tsunami and the 2018 Anchorage earthquake. By performing a CWT on our pressure data, we will be able to produce spectrograms of our data and hopefully identify the hydroacoustic resonance discovered by Bolsakova et al. Since the pressure data plotted as is can be quite difficult to observe a trend in the mean static pressure over time. Windowing the data and calculating the mean will allow us to produce easy-to-interpret plots of how the static pressure changes with time during the tsunami event. Since, a spectral analysis of the 2003 Tokachi-oki tsunami event was carried out by Bolshakova et al. already, we will first develop a routine to produce spectrograms and compare them with the results from their study to validate our routine. We will then apply this to data from the 2018 Anchorage earthquake.

2 Theoretical Background

2.1 Tsunamis

Tsunamis are surface gravitational waves with long periods ranging from $10^2 - 10^4$s that are caused by a sharp, sudden displacement of the seafloor (typically due to an earthquake).They differ from most other natural disasters in that the waves are able to retain their destructive force over distances of thousands of kilometers. The wave heights are capable of reaching tens of meters as the tsunami reaches the coast [4]. By treating the water near the tsunami source as compressible, Bolshakova et. al observed low-frequency elastic oscillations of the water column close to the tsunami source [1].

$$\nu_0 = \frac{c}{4H}$$

where c is the sound velocity in water and H is the ocean depth. For an average depth of 4km, $\nu_0 \approx 0.1$ [1]. This frequency is unique to the tsunami itself and allows us to potentially detect a tsunami close to the earthquake when our signal would otherwise be inundated by seismic waves. The static pressure in the ocean measured by an ocean-bottom pressure gauge can be related to the ocean depth by the following,

$$p = \rho g H$$

where p is the pressure, ρ is the density of water, g is acceleration due to gravity and H is the height of the water column at the gauge.

2.2 Windowing

Windowing allows us to only look at a subset of our signal at a given time. In this case, we would like to determine the mean excess pressure as a function of time from the messy pressure signal. To get a smooth trend, we choose to use a Gaussian window given by

$$f(t_c) = e^{-a(t-t_c)^2}$$

where a is a width parameter and t_c is where we center the our window function.

2.3 Continuous Wavelet Transform

In previous assignments, we highlighted the drawbacks of Fourier analysis as it fails to capture when specific frequencies occur in time. To overcome this, we introduce the continuous wavelet transform (CWT). The CWT windows our signal to localize it in time [3]. By shifting our window in time, we are then able to capture at what time specific frequencies are present. Furthermore, since our window size affects the range of frequencies we can capture, we also scale the width of our window to capture low and high frequency data. The CWT is given by [3],

$$W_\psi[f](a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} f(t)\bar{\psi}(\frac{t-b}{a})dt$$

where b determines where in time the window is centered and a determines how wide our window is and $\bar{\psi}$ is the conjugate of our wavelet. The Fourier transform of our wavelet is then given by,

$$\hat{\psi}_{a,b} = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} e^{-i\omega t}\psi(\frac{t-b}{a})dt$$

2.3.1 Morlet Wavelet

To carry out the CWT, we use the Morlet wavelet as described in Kristekova et al. 2006

$$\psi(t) = \pi^{-1/4} exp(i\omega_0 t) exp(-t^2/2)$$

where ω_0 is a parameter of our choosing that has trade-offs between time and frequency resolutions.

3 Algorithm Implementation and Development

3.1 Pre-Processing

There are 2 main pre-processing steps that are done before we can perform any analysis. First, we need to determine the range of time needed for our pressure data and also the sample rate. The time range should include data measured before the event and a significant time after as we are interested in measuring the static pressure when there is no tsunami or earthquake. We are using data from 4 sensors (2 for each event). PG1 and PG2 recorded data from the 2003 tsunami and LDH and BDH recorded data from the 2018 Anchorage Earthquake. For PG1 and PG2, the sample rate is 10Hz and the starting time was 17416 seconds from the start of the data set which was loaded from a text file. For LDH and BDH, using the ObsPy package to download the data, the sample rates were 1Hz and 40Hz respectively and a time range of 30 seconds before and 20 minutes after the event was used [6].

Next, in order to center our pressure data and remove the ambient pressure due to normal tidal action and the water column, we subtract the mean from our data. Since we include a lot of data before and after the event, we observe that our pressure signal is now centered at around 0. Finally, we convert our y-axis units from pressure to excess pressure in meters of water by the following,

$$H = p/(\rho g)$$

where p is the pressure measured by the gauge, ρ is the density of water, g is acceleration due to gravity and H is the height of the water column at the gauge.

3.2 Analysis

We window our signal by convolving our signal with a Gaussian function as described in Section 2.2. The resulting vector then becomes our windowed signal. Taking the sum of the windowed vector, we get a weighted average of our windowed data which corresponds to the center of the window. In order, to get the unweighted average, we then divide by the sum of the weights which in this case is the Gaussian function. To calculate how the excess pressure changes with time, we shift this window across the signal at some increment. We can change the increment spacing and also the width of the Gaussian to produce different results.

To produce the spectrograms, we will perform a CWT on our pressure data. In order to do so, we will use the `cwt` function in the ObsPy package for Python which utilizes the Morlet wavelet as described in section 2.3.1. Note that this function takes the sample rate as a parameter which is why we need to determine it in our pre-processing. We can also vary the parameter ω_0 which will give us different resolutions in time and frequency. Once the CWT has been performed, we can create a spectrogram by using the output of our CWT as weights to our pseudocolor plot. For the spectrograms in this analysis, we normalize our weights so that the maximum value is 1.

4 Computational Results

4.1 2003 Tokachi-oki Tsunami

PG1 and PG2 are located extremely close to the tsunami source (less than 100km) which are located off the coast of Japan [4]. PG1 is closer to the source and the distance between PG1 and PG2 is approximately 68 km [1].

4.1.1 PG1

Plotting the spectrogram of our pressure data for PG1 (Figure 1) along with the corresponding ν_0 (in this case $\nu_0 = 0$ [1]), we see that there is indeed an observable hydroacoustic resonance that is detectable long after the earthquake. At the beginning of the signal, we observe a large range of frequencies that correspond to the earthquake signal.

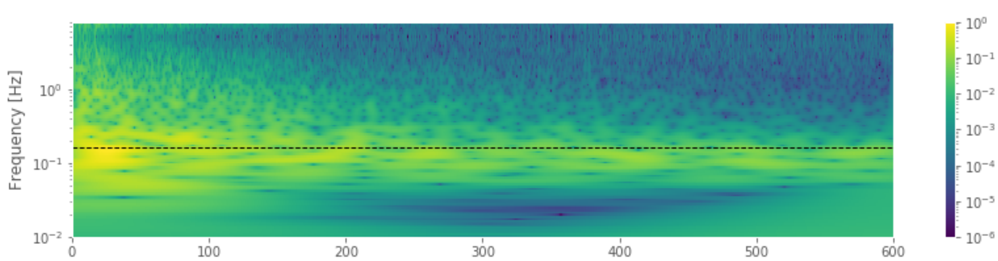

Figure 1: Spectrogram of PG1 (X-Axis is time in seconds)

Plotting our average excess pressure (Figure 2), we can also observe how the ocean depth is changing before and after the earthquake. There is a clear, abrupt decrease in the ocean depth which corresponds to uplift at the gauge by around 50-70cm.

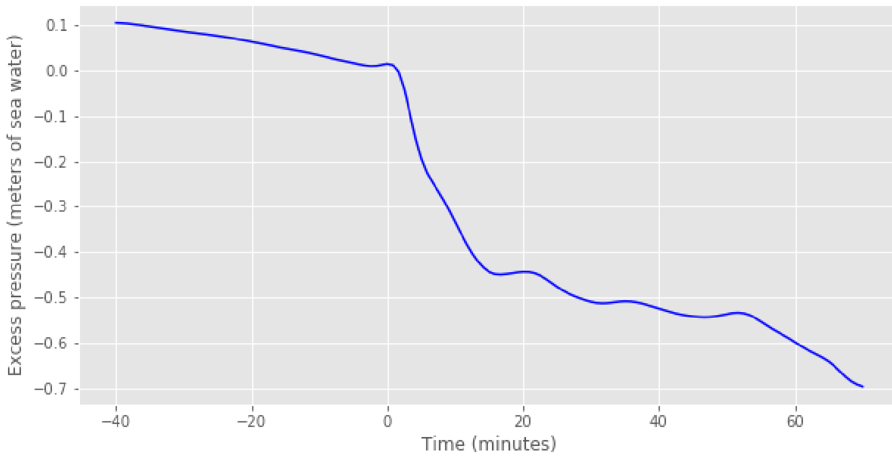

Figure 2: Mean excess pressure in meters for PG1

4.1.2 PG2

Moving onto PG2 (Figure 3), we once again observe a band of frequencies corresponding to ν_0 similar to PG1 that propagates long after the earthquake.

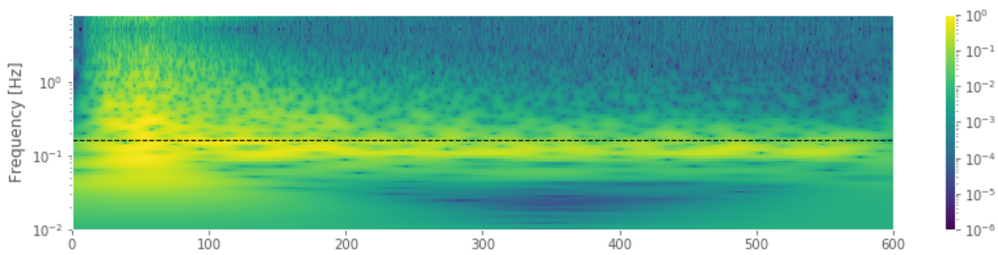

Figure 3: Spectrogram of PG2 (X-Axis is time in seconds)

The change in excess pressure also has a similar shape to that in PG1 (Figure 4). The overall change after the earthquake however, is about half of that in PG1. This is not unexpected as PG2 is farther away from the earthquake source than PG1.

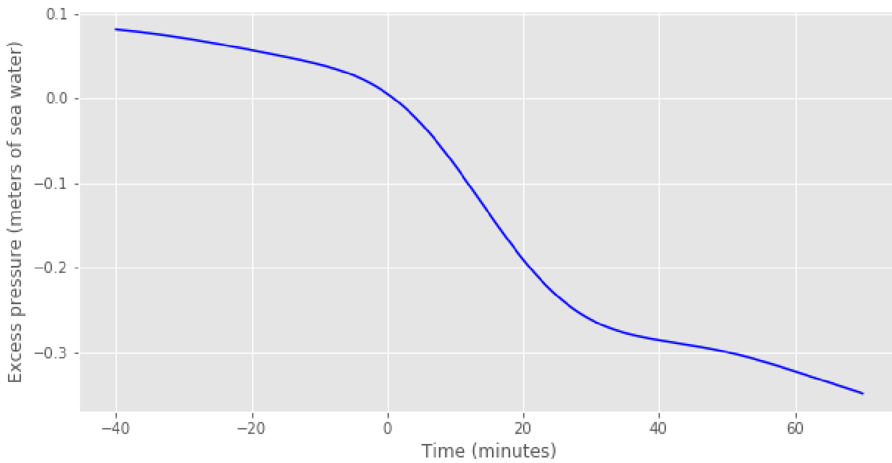

Figure 4: Mean excess pressure in meters for PG2

4.2 2018 Anchorage Earthquake

The 2018 Anchorage Earthquake differs from the 2003 Tokachi-oki Tsunami in that its epicenter is on land as opposed to offshore. Furthermore, the ocean-bottom sensors we are using are located significantly farther from the earthquake source than in the previous section (approximately 1000km) [5]. As a result, it is not clear what water depth we should use for calculating ν_0 or if a tsunami was produced at all.

4.2.1 LDH

Plotting our spectrogram for our low-sample rate sensor (1Hz) first, we observe that there is significantly less frequency content due to the distance between the sensor and the earthquake source. The red-dotted line here corresponds to $f = 0.1$ which his ν_0 for an average depth of 4km. Since we are still close to the coast, we would expect our hydroacoustic frequency to be greater than 0.1. Nonetheless, we still observe a band of low frequencies that propagate long after the earthquake (which reaches the sensor at about 230 seconds). We also notice a repeated pattern above 1Hz which may have to do with the sample rate of the sensor.

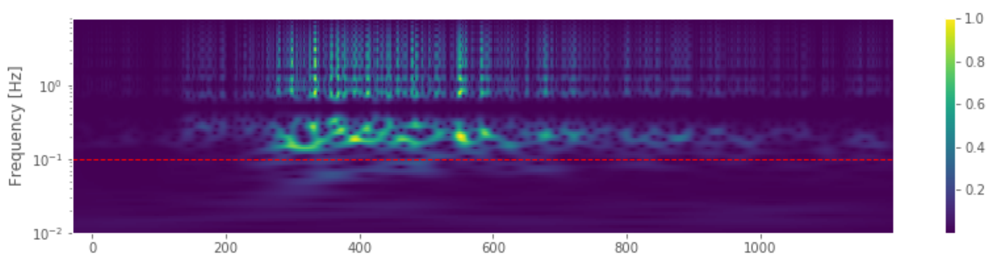

Figure 5: Spectrogram of LDH (X-Axis is time in seconds)

Turning our attention to the change in depth ((Figure 5)), we see that there is little change in the order of centimeters. As the sensor is far away from the epicenter, we expect little change in the seafloor depth.

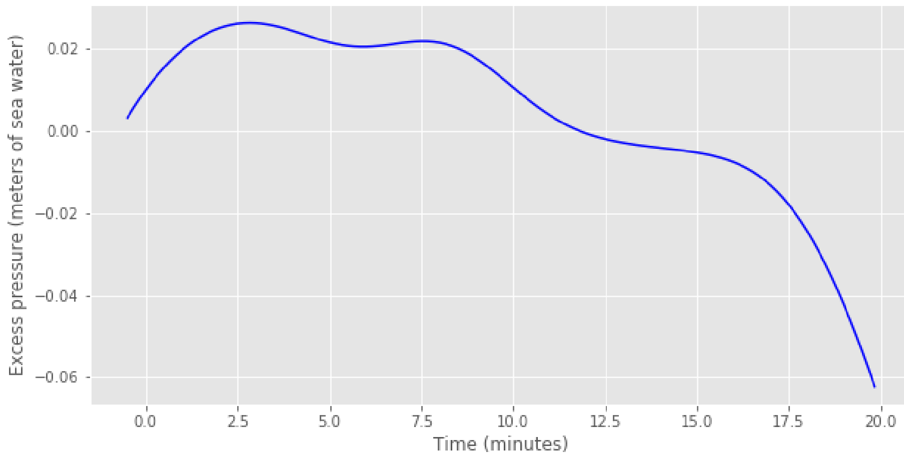

Figure 6: Mean excess pressure in meters for LDH

4.2.2 BDH

BDH, our high sample-rate sensor, produces similar results (Figure 7). A band of low frequency can be observed long after the earthquake has occurred. Interestingly enough, our higher sample-rate data removes this pattern above 1Hz we observed for the LDH data.

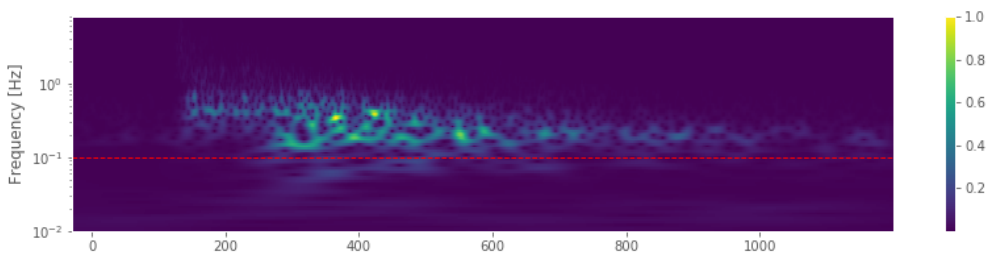

Figure 7: Spectrogram of BDH (X-Axis is time in seconds)

The change in depth (Figure 7) is also similar to that in LDH as both sensors are in the same location.

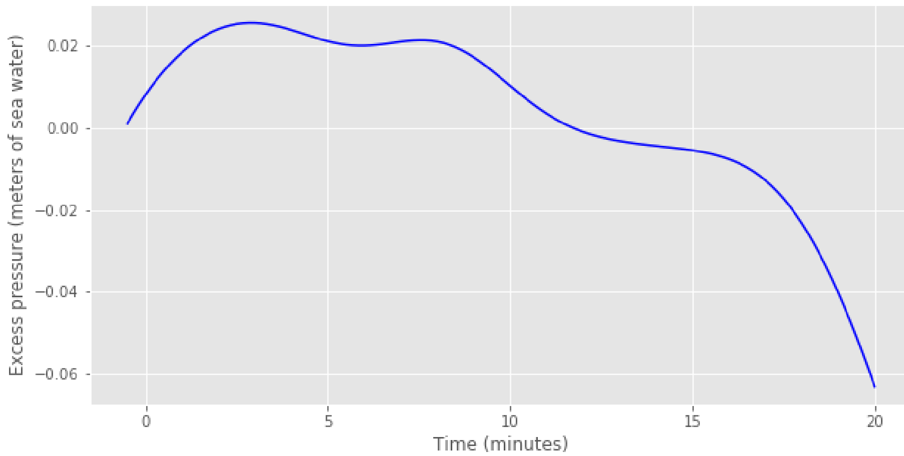

Figure 8: Mean excess pressure in meters for BDH

5 Summary and Conclusions

Comparing our results from the 2003 Tokachi-oki event and the 2018 Anchorage earthquake, it is not clear if we can directly extend the concepts derived from Bolshakova et al. to onshore earthquakes where the corresponding sensors are located very far away from the source. Nonetheless, we are still able to observe potential low-frequency oscillations that correspond to a tsunami. We were also able to show how the static pressure measurements from these pressure gauges can be used a proxy for ocean depth. In the case of PG1 and PG2, we were able to observe changes in the depth that seem to meet our expectations.

6 References

[1] Bolshakova, A., et al. "Hydroacoustic Effects in the 2003 Tokachi-Oki Tsunami Source." Russian Journal of Earth Sciences, vol. 12, no. 2, 2011, pp. 1–14., doi:10.2205/2011es000509.

[2] Kristekova, M., et al. "Misfit Criteria for Quantitative Comparison of Seismograms." Bulletin of the Seismological Society of America, vol. 96, no. 5, 2006, pp. 1836–1850., doi:10.1785/0120060012.

[3] Kutz, Jose Nathan. Data-Driven Modeling & Scientific Computation. Methods for Complex Systems & Big Data. Oxford University Press, 2013.

[4] Levin, Boris W., and Mikhail A. Nosov. Physics of Tsunamis. Springer International Publishing, 2016.

[5] "M 7.1 - 14km NNW of Anchorage, Alaska." USGS Earthquake Hazards Program, 2018, earthquake.usgs.gov/earthquakes/even

[6] "XO : WS74 (2018-07-18 - 2019-09-04)." IRIS, ds.iris.edu/mda/XO/WS74/?starttime=2018-07-21&endtime=2019-08-29.

Appendix A Python Functions

Below are the key Python functions implemented.

- `loadtxt('filename')` loads data from specified file name
- `where(condition,[x,y])` returns elements in x or y that meet the condition
- `obspy.signal.tf_misfit.cwt(st, dt,` ω_0`, fmin, fmax,)` performs the continuous wavelet transform on a signal st where dt is the time between two samples, ω_0 is our resolution parameter and fmin, fmax are the min and max frequencies.
- `np.meshgrid` returns coordinate matrices from the specific coordinate vectors

- `np.logspace` returns evenly spaced numbers on a logscale
- `pcolormesh(x,y,weight)` produces a pseudocolor plot with x vs y with the corresponding weight
- `arange` returns evenly space values on a given interval
- `client.get_waveforms(etwork, station, location, channel, starttime, endtime)` downloads the traces from a specific network, station, location, channel, and time.

Appendix B Python Code

For JupyterNotebook with output, refer to listing in Github
https://github.com/chrismhl/AMATH582/tree/master/Final%20Project

```python
from __future__ import print_function
from pylab import *
from scipy import signal
import matplotlib.pyplot as plt
import matplotlib.colors as colors
import numpy

import obspy
from obspy.imaging.cm import obspy_sequential, pqlx #colormaps
from obspy.signal.tf_misfit import cwt

from obspy.clients.fdsn import Client
client = Client("IRIS")

#2003 tokachi-oki
#pg1
d = loadtxt('kpg1.tp.txt')
t0 = 17416.
t = (d[:,0] - t0) # seconds post event
dt = 0.1

kmean = where(logical_and(t>-3, t<0))[0]
pmean = mean(d[kmean,1])
print('pmean = %g, mean(d[:,1]) = %g' % (pmean,mean(d[:,1])))

p = (d[:,1] - pmean) / (9.81*1025)  # excess pressure in m sea water

figure(1, figsize=(10,4))
clf()
plot(t/60.,p,'b',linewidth=0.5)
xlim(-5,50)
ylim(-30,30)
grid(True)
xlabel('minutes after earthquake')
ylabel('Excess pressure (meters of sea water)')

figure(2, figsize=(10,4))
clf()
plot(t/60.,p,'b',linewidth=1.0)

grid(True)
xlabel('minutes after earthquake')
ylabel('Excess pressure (meters of sea water)')
```

```


plt.style.use('ggplot')
figure(1, figsize=(12,4))
clf()
plot(t[173560:177161]/60.,p[173560:177161],'b',linewidth=0.5)
xlim(-1,5)
ylim(-30,30)
grid(True)
xlabel('minutes after earthquake')
ylabel('Excess pressure (meters of sea water)')

savefig('pg1sig_png')

#truncating the data for the spectrogram
p_trunc = p[174160:180161]
t_trunc = t[174160:180161]

f_min = 0.01
f_max = 8
w0 = 10 #parameter for the wavelet, tradeoff between time and frequency resolution

scalogram = cwt(p_trunc, dt, w0, f_min, f_max)

fig = plt.figure(figsize=(14,3))
ax = fig.add_subplot(111)

x, y = np.meshgrid(t_trunc,np.logspace(np.log10(f_min), np.log10(f_max), scalogram.
    shape[0]))
scalonorm = np.abs(scalogram)/np.amax(abs(scalogram)) #normalize scalogram

p1 = ax.pcolormesh(x, y, np.abs(scalonorm), cmap=obspy_sequential, norm=colors.
    LogNorm(vmin=1e-6, vmax=1))

#dotted lines corresponding to the frequencies
plt.axhline(y=0.165,color='black',ls='--',lw=1)
#plt.axhline(y=0.138,color='black',ls='--',lw=1)

ax.set_xlabel("Time after 17416 [s]")
ax.set_ylabel("Frequency [Hz]")
ax.set_yscale('log')
ax.set_ylim(f_min, f_max)
plt.colorbar(p1)
savefig('pg1spec.png')

# plot running average over window
j = where(logical_and(t>=-40*60,t<=70*60))[0]
tt = t[j]
pp = p[j]
# depth = 2283m = 1.522 sec, dt= 0.1 sec, 1.522*10=15.22

width = 10**-4 #width of gaussian

#gaussian
tts = tt-min(tt)
t_incr = arange(min(tts),max(tts)+1,50)
```

```
pw = zeros(len(t_incr))

for jj in range(0,len(t_incr)):
    window = np.exp(-width*np.power((tts-t_incr[jj]),2));
    windowed = np.multiply(pp,window)

    pw[jj] = sum(windowed)/sum(window)

plt.figure(figsize=(10,5))
plot((t_incr+min(tt))/60., pw,'b')

xlabel('Time (minutes)')
ylabel('Excess pressure (meters of sea water)')
savefig('pg1height.png')

#pg2
d2 = loadtxt('kpg2.tp.txt')
t20 = 17416.
t2 = (d2[:,0] - t20) # seconds post event
dt = 0.1

kmean = where(logical_and(t2>-3, t2<0))[0]
pmean = mean(d2[kmean,1])
print('pmean = %g, mean(d[:,1]) = %g' % (pmean,mean(d2[:,1])))
p2 = (d2[:,1] - pmean) / (9.81*1025)  # excess pressure in m sea water

figure(1, figsize=(10,4))
clf()
plot(t2/60.,p2,'b',linewidth=0.5)
xlim(-5,50)
ylim(-30,30)
grid(True)
xlabel('minutes after earthquake')
ylabel('Excess pressure (meters of sea water)')

figure(2, figsize=(10,4))
clf()
plot(t2/60.,p2,'b',linewidth=1.0)

grid(True)
xlabel('minutes after earthquake')
ylabel('Excess pressure (meters of sea water)')

plt.style.use('ggplot')
figure(1, figsize=(12,4))
clf()
plot(t2[173560:177161]/60.,p2[173560:177161],'b',linewidth=0.5)
xlim(-1,5)
ylim(-30,30)
grid(True)
xlabel('minutes after earthquake')
ylabel('Excess pressure (meters of sea water)')

savefig('pg2sig_png')

#truncating the data for the spectrogram
```

```
p_trunc2 = p2[174160:180161]
t_trunc2 = t2[174160:180161]

f_min = 0.01
f_max = 8
w0 = 10 #parameter for the wavelet, tradeoff between time and frequency resolution

scalogram = cwt(p_trunc2, dt, w0, f_min, f_max)

fig = plt.figure(figsize=(14,3))
ax = fig.add_subplot(111)

x, y = np.meshgrid(t_trunc2,np.logspace(np.log10(f_min), np.log10(f_max), scalogram.shape[0]))
scalonorm = np.abs(scalogram)/np.amax(abs(scalogram)) #normalize scalogram

p1 = ax.pcolormesh(x, y, np.abs(scalonorm), cmap=obspy_sequential, norm=colors.LogNorm(1e-6, vmax=1))

#dotted lines corresponding to the frequencies
plt.axhline(y=0.165,color='black',ls='--',lw=1)
#plt.axhline(y=0.138,color='black',ls='--',lw=1)

ax.set_xlabel("Time after 17416 [s]")
ax.set_ylabel("Frequency [Hz]")
ax.set_yscale('log')
ax.set_ylim(f_min, f_max)
plt.colorbar(p1)

savefig('pg2spec.png')

# plot running average over window
j = where(logical_and(t2>=-40*60,t2<=70*60))[0]
tt = t2[j]
pp = p2[j]
# depth = 2283m = 1.522 sec, dt= 0.1 sec, 1.522*10=15.22

width = 10**-6 #width of gaussian

#gaussian
tts = tt-min(tt)
t_incr = arange(min(tts),max(tts)+1,50)
pw = zeros(len(t_incr))

for jj in range(0,len(t_incr)):
    window = np.exp(-width*np.power((tts-t_incr[jj]),2));
    windowed = np.multiply(pp,window)

    pw[jj] = sum(windowed)/sum(window)

plt.figure(figsize=(10,5))
plot((t_incr+min(tt))/60., pw,'b')

xlabel('Time (minutes)')
ylabel('Excess pressure (meters of sea water)')
```

```
savefig('pg2height.png')

#2018 anchorage

t1 = obspy.UTCDateTime("2018-05-01T00:00:00")
t2 = obspy.UTCDateTime("2019-08-01T00:00:00")
catalog = client.get_events(starttime=t1, endtime=t2,
                            minlatitude=52,
                            maxlatitude=62,
                            minlongitude=-170,
                            maxlongitude=-130,
                            minmagnitude=6.)
print(catalog)
catalog.plot();

t = obspy.UTCDateTime("2018-11-30T17:29:29.33Z")
s = client.get_waveforms("XO", "WS74", "*", "?DH",t - 30, t + 20 * 60)
print(s)

s.plot()

bdh = s[0]
ldh = s[1]

#bdh

print(bdh.stats)

bdh_p = (bdh.data - mean(bdh.data))/(9.81*1000)
bdh_t = arange(49200)/40
dt = 0.025

plt.style.use('ggplot')
plt.rcParams['figure.figsize'] = 12, 4

plot((bdh_t-30)/60,bdh_p)
xlabel('Time(Minutes)')
ylabel('Excess pressure (meters of sea water)')

savefig('bdhsig_png')

plt.style.use('ggplot')
plt.rcParams['figure.figsize'] = 12, 4

plot((bdh_t-30)/60,bdh_p)
xlabel('Time(Minutes)')
ylabel('Excess pressure (meters of sea water)')

savefig('bdhsig_png')

f_min = 0.01
f_max = 8
w0 = 10 #parameter for the wavelet, tradeoff between time and frequency resolution

scalogram = cwt(bdh_p, dt, w0, f_min, f_max)

```

```python
fig = plt.figure(figsize=(14,3))
ax = fig.add_subplot(111)

x, y = np.meshgrid(bdh_t-30,np.logspace(np.log10(f_min), np.log10(f_max), scalogram.shape[0]))
scalonorm = np.abs(scalogram)/np.amax(abs(scalogram)) #normalize scalogram

p1 = ax.pcolormesh(x, y, np.abs(scalonorm), cmap=obspy_sequential)

#dotted lines corresponding to the frequencies
plt.axhline(y=0.1,color='red',ls='--',lw=1)

ax.set_xlabel("Time [s]")
ax.set_ylabel("Frequency [Hz]")
ax.set_yscale('log')
ax.set_ylim(f_min, f_max)
plt.colorbar(p1)

savefig('bdhspec.png')

# plot running average over window
width = 10**-4.5 #width of gaussian

#gaussian
tts = bdh_t
t_incr = arange(min(tts),max(tts)+1,10)
pw = zeros(len(t_incr))

for jj in range(0,len(t_incr)):
    window = np.exp(-width*np.power((tts-t_incr[jj]),2));
    windowed = np.multiply(bdh_p,window)

    pw[jj] = sum(windowed)/sum(window)

plt.figure(figsize=(10,5))
plot((t_incr-30)/60., pw, 'b')

xlabel('Time (minutes)')
ylabel('Excess pressure (meters of sea water)')

savefig('bdhheight.png')

#LDH
print(ldh.stats)

ldh_p = (ldh.data - mean(ldh.data))/(9.81*1000)
ldh_t = arange(1230)
dt = 1

plot((ldh_t-30)/60,ldh_p)
xlabel('Time(Minutes)')
ylabel('Excess pressure (meters of sea water)')

savefig('ldhsig_png')

f_min = 0.01
```

```
f_max = 8
w0 = 10 #parameter for the wavelet, tradeoff between time and frequency resolution

scalogram = cwt(ldh_p, dt, w0, f_min, f_max)

fig = plt.figure(figsize=(14,3))
ax = fig.add_subplot(111)

x, y = np.meshgrid(ldh_t -30,np.logspace(np.log10(f_min), np.log10(f_max), scalogram.shape[0]))
scalonorm = np.abs(scalogram)/np.amax(abs(scalogram)) #normalize scalogram

p1 = ax.pcolormesh(x, y, np.abs(scalonorm), cmap=obspy_sequential)

#dotted lines corresponding to the frequencies
plt.axhline(y=0.1,color='red',ls='--',lw=1)

ax.set_xlabel("Time [s]")
ax.set_ylabel("Frequency [Hz]")
ax.set_yscale('log')
ax.set_ylim(f_min, f_max)
plt.colorbar(p1)

savefig('ldhspec.png')


# plot running average over window
width = 10**-4.5 #width of gaussian

#gaussian
tts = ldh_t
t_incr = arange(min(tts),max(tts)+1,10)
pw = zeros(len(t_incr))

for jj in range(0,len(t_incr)):
    window = np.exp(-width*np.power((tts-t_incr[jj]),2));
    windowed = np.multiply(ldh_p,window)

    pw[jj] = sum(windowed)/sum(window)

plt.figure(figsize=(10,5))
plot((t_incr -30)/60., pw, 'b')

xlabel('Time (minutes)')
ylabel('Excess pressure (meters of sea water)')

savefig('ldhheight.png')
```