Sea and land breeze simulation at IJmuiden

Project 1 - Morphodynamics of wave-dominated coasts

Paulien Koster, Jelle Verburg & Koen van der Zee

February 14, 2024

Pre-processing

The signal's duration, D, is 60 minutes since D is given by the number of time steps divided by the frequency: $D = 14400/4\text{Hz} = 3600$ s $= 60$ minutes.

With this dataset, we can calculate from the pressure the water depth. With this, the average water depth, h_{mean}, at the sensor is: 3.774

We can also plot this water depth for the different time steps, they are shown in the following figure:

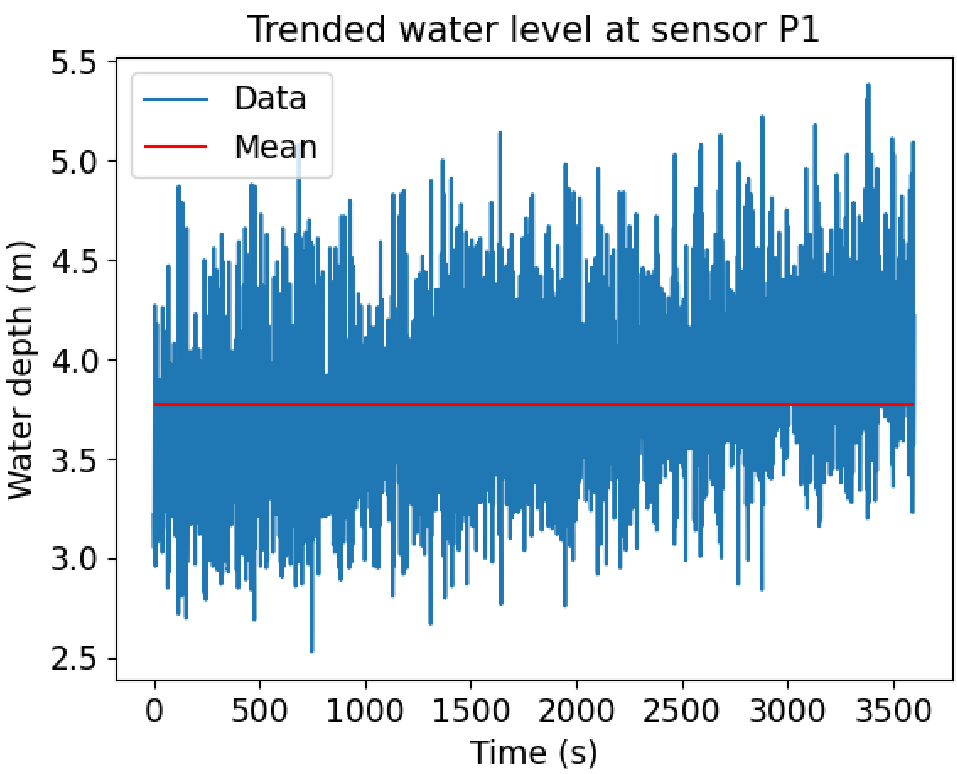

Figure 1: Evolution of h as function of time

In this figure, we see an increase in water depth over time. We expect that this happens because of an incoming tide. Because the time is only 60 minutes, we cannot see the whole tidal wave, only a part of it which is only increasing in this case. Now, we can also consider the detrended water depth and together with that the water depth minus the mean water depth. They are shown in the following figures:

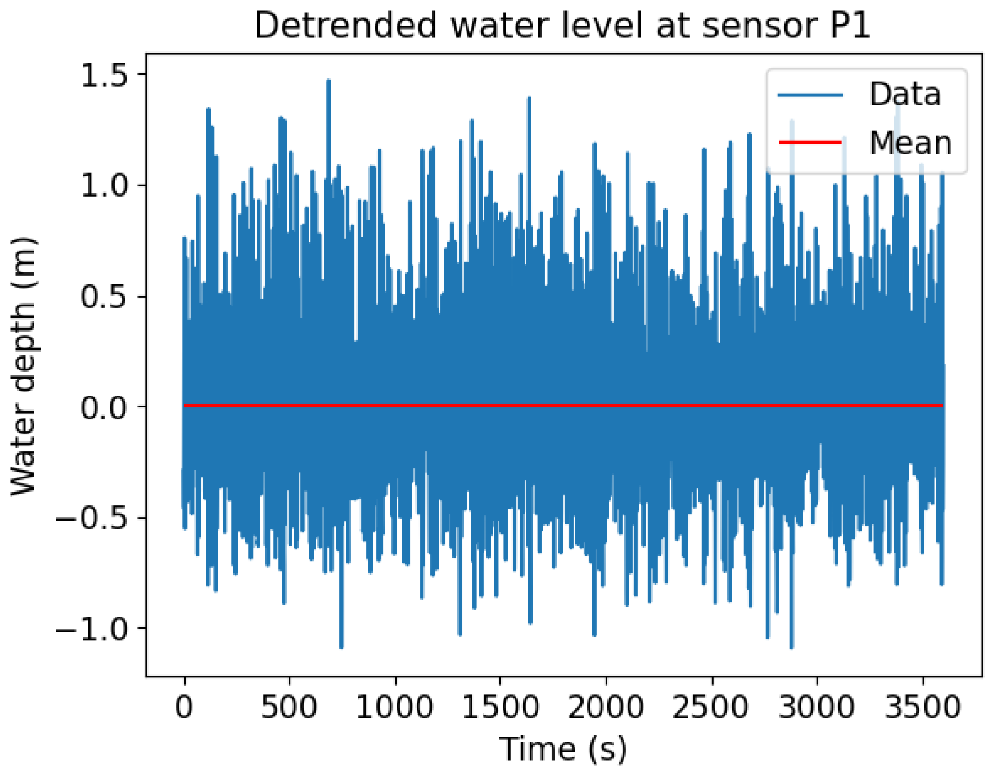

Figure 2: Detrended water level.

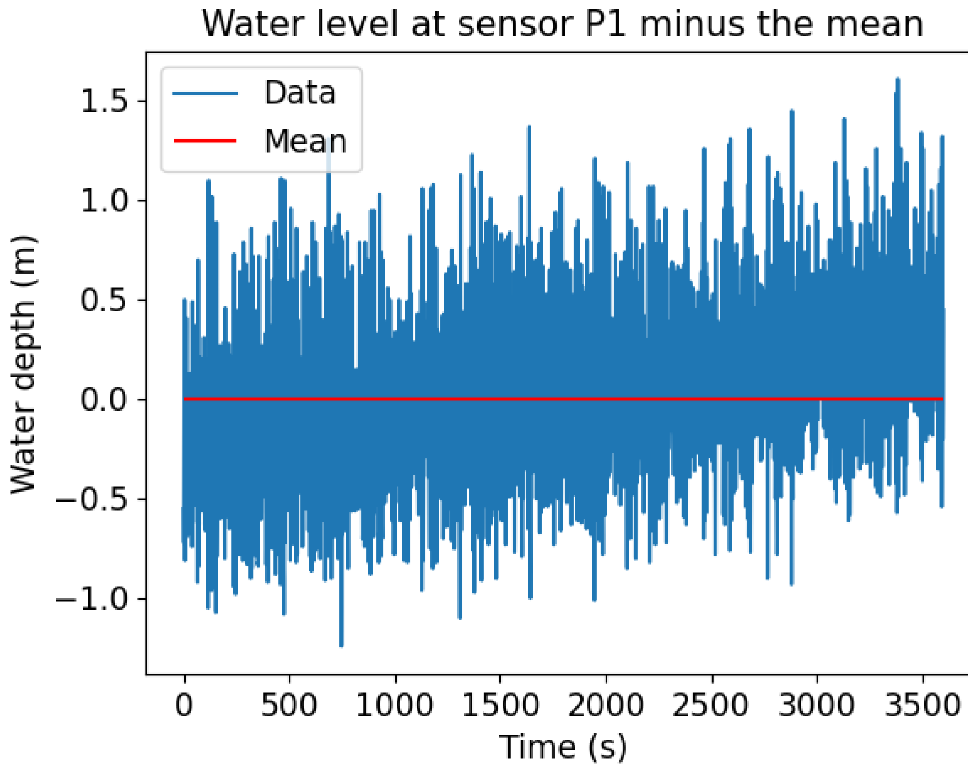

Figure 3: Water level compared with mean water level.

We filter the linear increase in mean water level with a detrend-function, which results in Figure 2. Now, the signal oscillates around zero and thus the trend is filtered out of the signal. In Figure 3, which depicts $h - h_{mean}$, the signal does not oscillate around zero. The trend is not filtered out of the signal.

Wave statistics

1.2.1 Wave statistics at low tide

In this section, we only consider the low and the high tide and we plot them for 3 different locations. They are shown in the following plot:

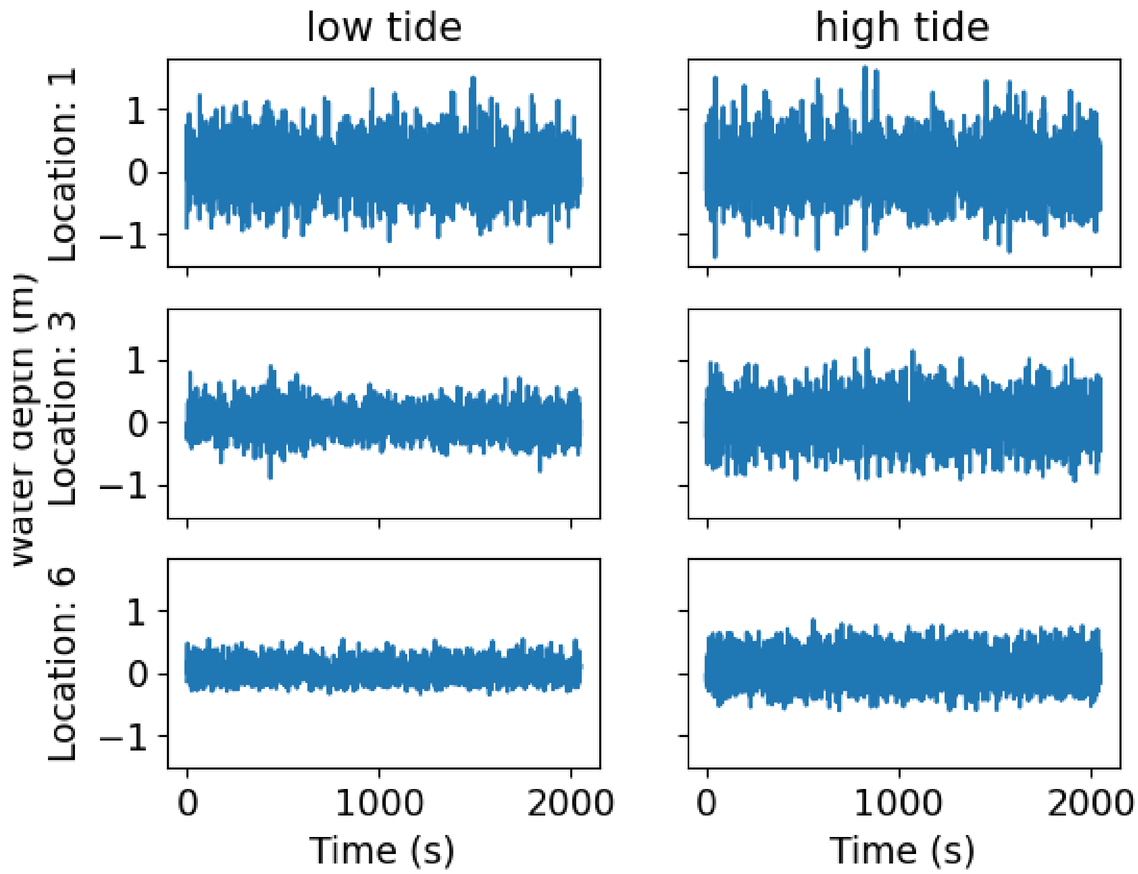

Figure 4: Water depths (m) for low tide (left column) and high tide (right column) for the location, from top to bottom, P1, P3, and P6.

At P1 the wave height for the time-series is visually estimated at 1.5m at low tide and 1.6m at high tide.

For P3 the wave height is about 1.0m at low tide and 1.3m at high tide.

Looking at the time-series at P6 we estimate that the wave height is 0.6m at low tide and 1.0m at high tide.

As we can see, in the figure, for all sensors, we have a higher deviation of zero for high tides than for low tides. Also for higher sensors, we have lower wave heights. If we compare this to the bedrock height, we can see that if we have a higher mean water depth, we have higher waves. This is both for the high tides as for the deeper bedrock elevation.

1.2.2 Wave statistics for full data set

First of all, we make a new script which we can import with the functions of the manual. After that, we put our attention to only the dataset of the low tides. With this, we can calculate the mean wave height, the significant wave height, and the root mean square wave height. The values we got are summarized in the following table:

Sensor	**Mean**	**Significant**	**RMS**
P1	0.90975889	1.44367489	1.02092234
P3	0.5332715	0.80421773	0.58695961
P4	0.44894705	0.67178023	0.4917019
P5	0.36073143	0.53830683	0.39394349
P6	0.33314973	0.49521039	0.3621391

Table 1: the low tide H_{mean}, $H_{Significant}$, and H_{RMS} for all six location.

We can also plot those values together with the bedrock elevation. They are given in the following plot.

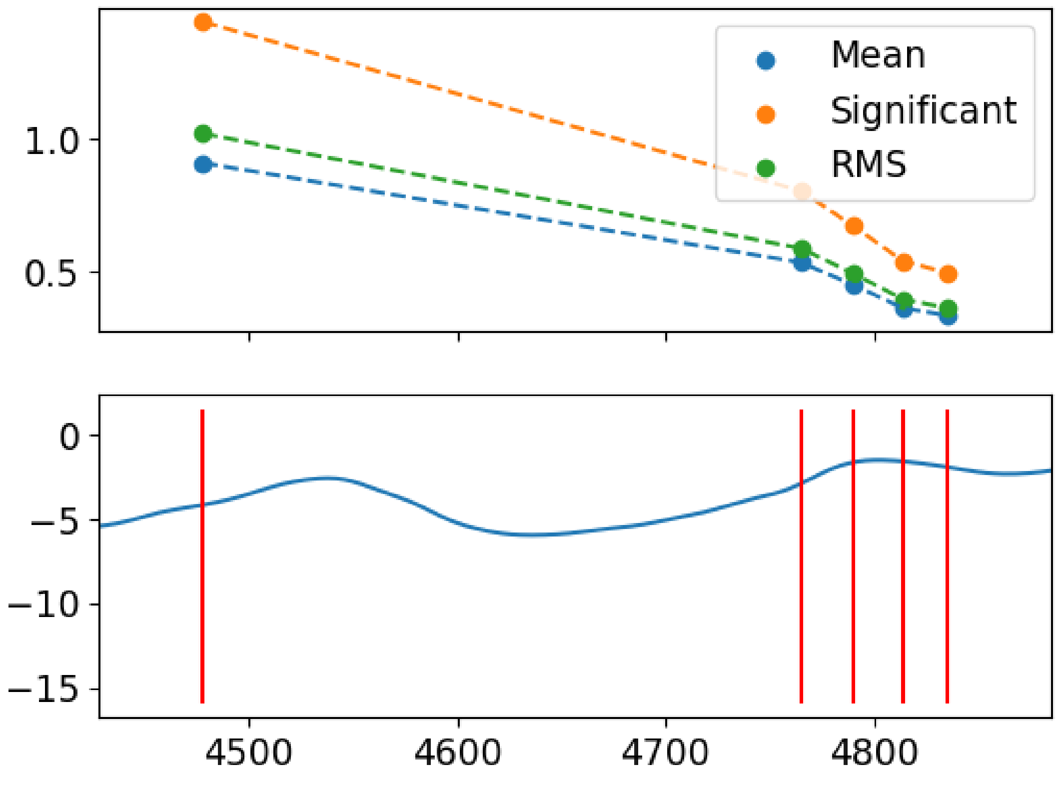

Figure 5: Top: Values from Table 1 plotted for every position. Bottom: Bedrock elevation.

Here, we observe that there is an almost linear relationship between the three different functions. This is as expected.

Full dataset statistics

After this, we put our attention to the full datasets. Thus, we do the same calculations for also the mid and the high tides. In the following plots, we plotted the significant wave height and the root mean square wave height.

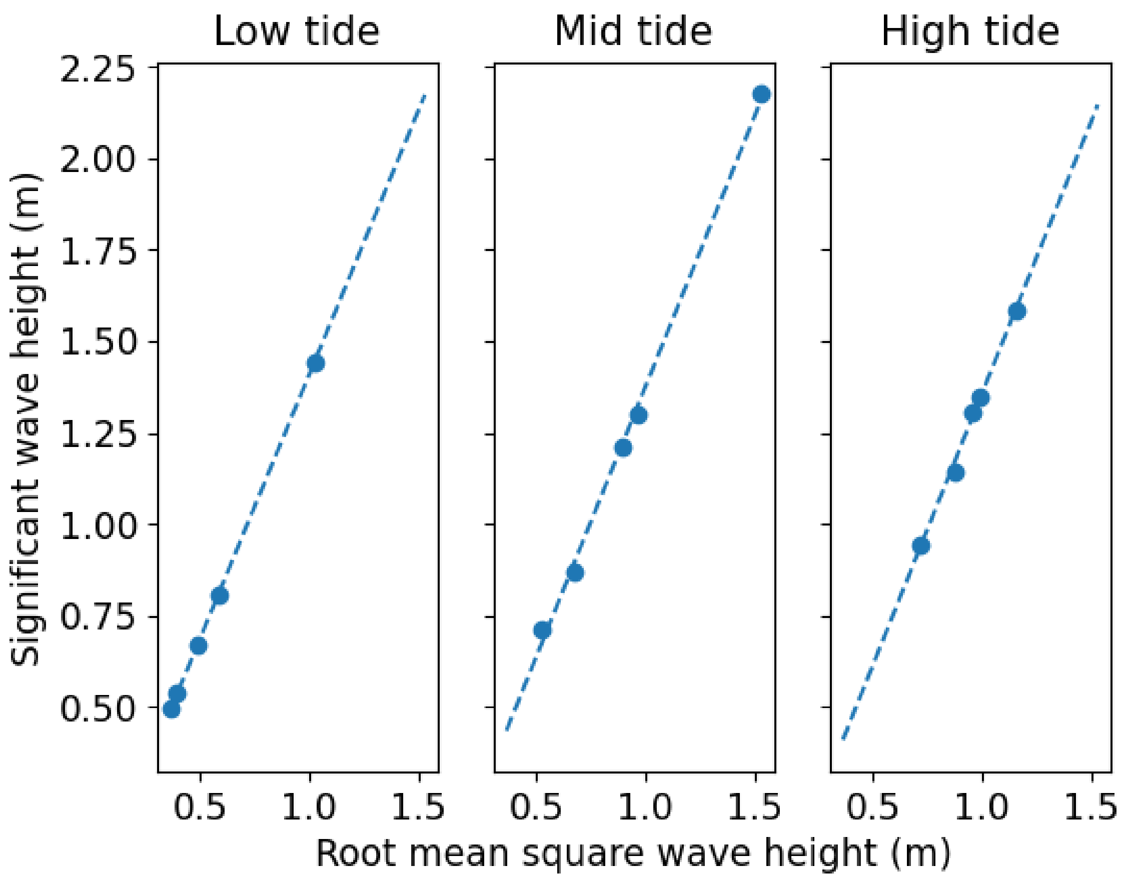

Figure 6: The relation between $H_{Significant}$ and H_{RMS} for the low tide, mid tide, and high tide.

Here, we get slopes of 1.44337195, 1.47736809, 1.48673988 which are close to $\sqrt{2} = 1.4142135623730951$. However, we can do it even better by taking all the tides together.

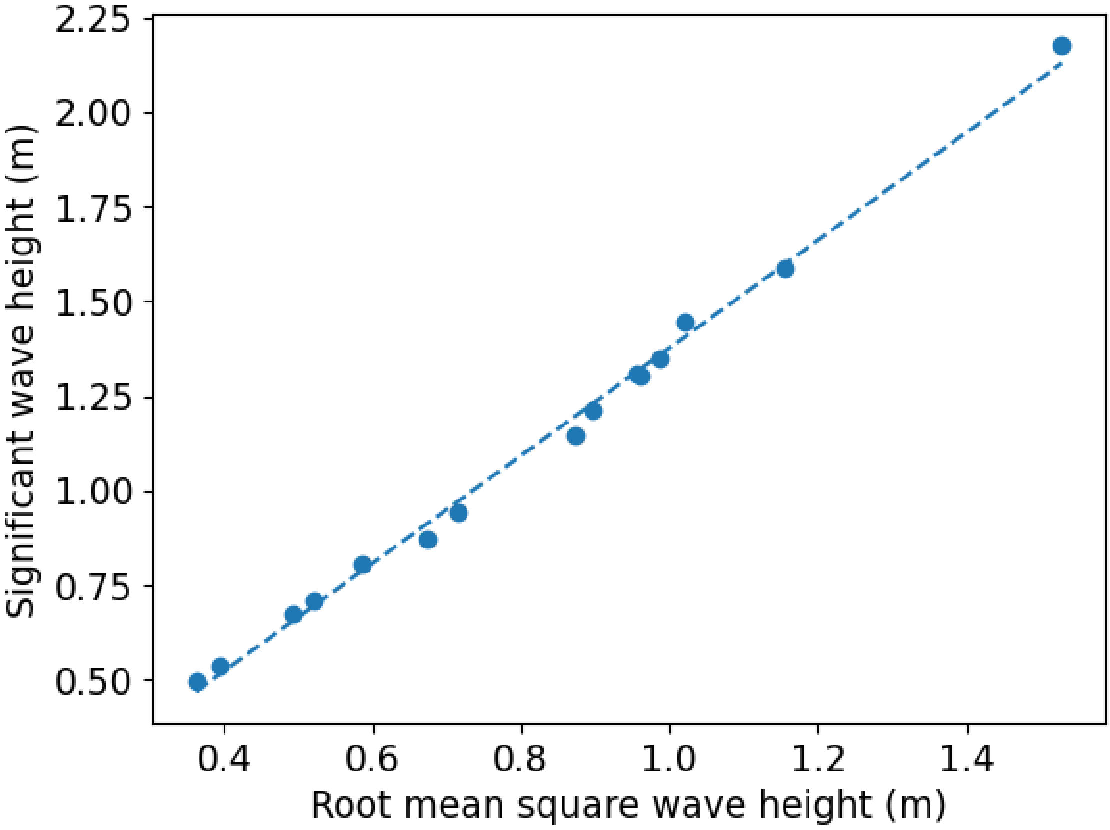

Figure 7: The relation between $H_{Significant}$ and H_{RMS} for all tides combined.

Then, we get a slope of 1.423397506665318 which is even closer to $\sqrt{2} = 1.4142135623730951$. This means that the relation goes very well in this case.
Now, we make the same plot for the Root mean square wave height and the mean wave height and we find:

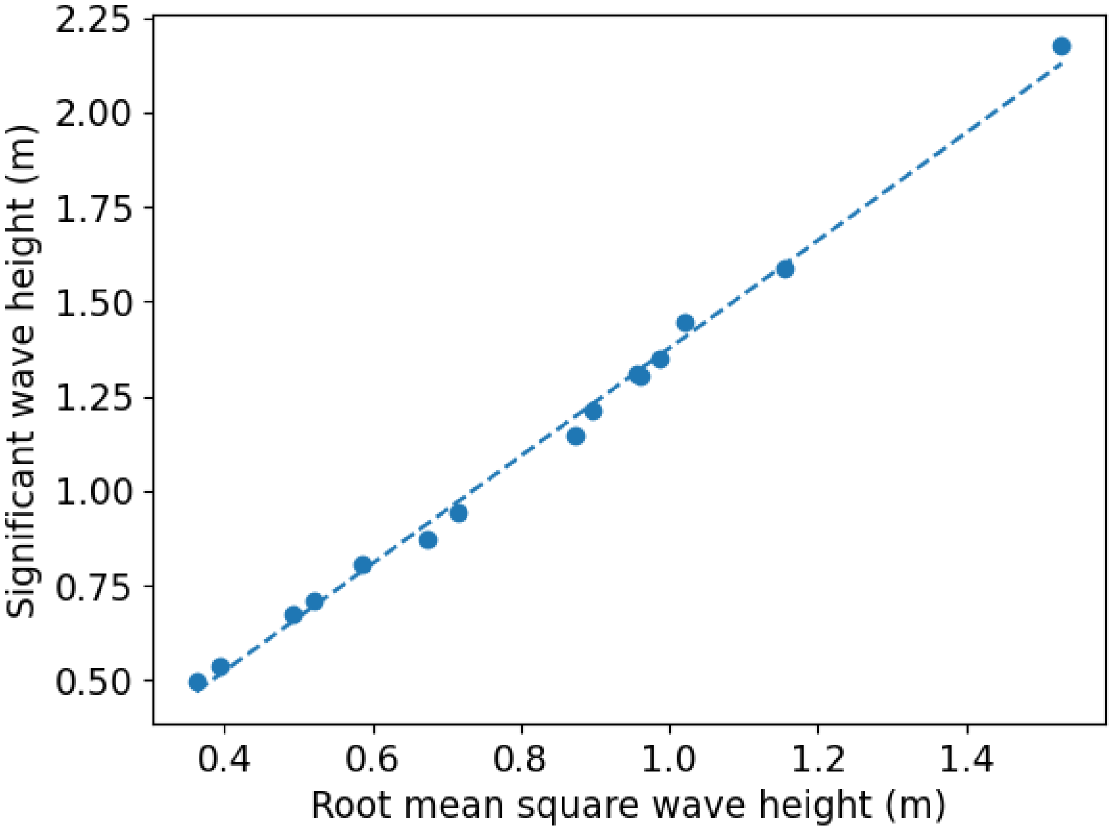

Figure 8: Caption

Here, the slope of the trend line becomes 0.8913804959830316 which is close to the expected value 0.89. In the end, we save the found values with the code of the manual such that we can use these values in the future.

Scripts

Exercise 1

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy

df = np.loadtxt("CalibP1.txt") #Load dataset

print("length of the dataset: ", np.size(df)/4/60, "minutes")

params = {'legend.fontsize': 13.5, #fix values for the fontsizes etc.
          'axes.labelsize': 13.5,
          'axes.titlesize':15,
          'xtick.labelsize':13.5,
          'ytick.labelsize':13.5,
          'legend.title_fontsize':13.5}
plt.rcParams.update(params)

#%% First plot of the dataset
plt.figure()
plt.plot(np.arange(0,60,1/4/60), df)

#%% Calculate from the dataset the pressure
a1 = 19.97370
b1 = -49.95959

p = a1*df+b1

plt.figure()
plt.plot(np.arange(0,60,1/4/60), p)

#%% Calculate from the pressure the water depth

rho = 1025
g = 9.81

ha = p/rho/g
hb = 1.45

h = ha+hb

print("Mean water depth: ", np.mean(h))

t = np.arange(0,60*60,1/4)

#%% Plot the water depth
```

```python
plt.figure()
plt.plot(t, h, label = "Data")
plt.xlabel("Time (s)")
plt.ylabel("Water depth (m)")
plt.hlines(np.mean(h), t[0], t[-1], color = "red", label = "Mean")
plt.title("Trended water level at sensor P1")
plt.legend()
# plt.savefig("Figures/Exercise_1a.png")

#%% Plot the water depth zoomed in on a small part
plt.figure()
plt.plot(t, h, label = "Data")
plt.xlim(60*20,60*21)
plt.ylim(3,4.5)
plt.xlabel("Time (s)")
plt.ylabel("Water depth (m)")
plt.hlines(np.mean(h), t[0], t[-1], color = "red", label = "Mean")
plt.title("Trended water level at sensor P1")
plt.legend()
# plt.savefig("Figures/Exercise_1a_zoom.png")

#%% Plot the detrended water depth
h_det = scipy.signal.detrend(h)

plt.figure()
plt.plot(t, h_det, label = "Data")
plt.xlabel("Time (s)")
plt.ylabel("Water depth (m)")
plt.hlines(np.mean(h_det), t[0], t[-1], color = "red", label = "Mean")
plt.title("Detrended water level at sensor P1")
plt.legend()
# plt.savefig("Figures/Exercise_1b.png")

#%% Plot the detrended water depth zoomed in on a small part
plt.figure()
plt.plot(t, h_det, label = "Data")
plt.xlim(60*20,60*21)
plt.ylim(-0.75,1)
plt.xlabel("Time (s)")
plt.ylabel("Water depth (m)")
plt.hlines(np.mean(h_det), t[0], t[-1], color = "red", label = "Mean")
plt.title("Detrended water level at sensor P1")
plt.legend()
# plt.savefig("Figures/Exercise_1b_zoom.png")

#%% Plot the water depth minus the mean water depth
plt.figure()
```

```python
plt.plot(t, h - np.mean(h), label = "Data")
plt.xlabel("Time (s)")
plt.ylabel("Water depth (m)")
plt.hlines(np.mean(h - np.mean(h)), t[0], t[-1], color = "red",
    label = "Mean")
plt.title("Water level at sensor P1 minus the mean")
plt.legend()
# plt.savefig("Figures/Exercise_1c.png")

#%% Plot the water depth minus the mean water depth  zoomed in on
    a small part
plt.figure()
plt.plot(t, h - np.mean(h), label = "Data")
plt.xlim(60*20,60*21)
plt.ylim(-0.75,0.75)
plt.xlabel("Time (s)")
plt.ylabel("Water depth (m)")
plt.hlines(np.mean(h - np.mean(h)), t[0], t[-1], color = "red",
    label = "Mean")
plt.title("Water level at sensor P1 minus the mean")
plt.legend()
# plt.savefig("Figures/Exercise_1c_zoom.png")
```

Exercise 2

The significant height script:

```python
'''
In this script, we caculate from a dataset the zerocrossing for the exercise.
Also, from the wave heights which come from that zerocross function, we can calculate the significant height and the root mean square height.
Those functions are given in the manual and in the lectures
'''


from zero_crossing import zero_crossing
import numpy as np

def zerocross(array):
    return zero_crossing(array, 2)[:,0]

def significant_height(cross):

    if int(np.ceil(2/3*np.size(cross))) == np.size(cross):
        return np.sort(cross)[-1]
    else:
        return np.mean(np.sort(cross)[int(np.ceil(2/3*np.size(cross))):])

def rms_height(cross):
    return np.sqrt(np.mean(cross**2))
```

The script to calculate and plot everything of exercise 2:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy
import pickle

params = {'legend.fontsize': 13.5, #fix values for the fontsizes etc.
          'axes.labelsize': 13.5,
          'axes.titlesize':15,
          'xtick.labelsize':13.5,
          'ytick.labelsize':13.5,
          'legend.title_fontsize':13.5}
plt.rcParams.update(params)

#%% Load data and calculate the time array

low = np.loadtxt("lowTide.txt")
mid = np.loadtxt("midTide.txt")
high = np.loadtxt("highTide.txt")

bed =  np.loadtxt("prof1018.txt")

t = np.arange(0, np.size(low[:, 0])/2,1/2)

#%% Plot the 6 plots for the different tides (low and high) and the different locations
array = np.array([0,1,4]) #the location is python array
array_name = np.array([1,3,6]) #the real name of the locations
low_high = np.array(["low", "high"]) #For the low and the high tide

fig, axs = plt.subplots(np.size(array),2, sharex = True, sharey = True)

for i in range(np.size(array)): #Plot
    axs[i,0].plot(t, low[:,array[i]])
    axs[i,1].plot(t, high[:,array[i]])

for i in range(2): #Set the labels and title in the right positions.
    for j in range(3):
        if j == 2:
            axs[j,i].set(xlabel = "Time (s)")
        if i ==0 and j ==1:
            axs[j,i].set(ylabel = f"Water depth (m) \n Location: {array_name[j]}")
        if i ==0 and j != 1:
```

```python
            axs[j,i].set(ylabel = f"Location: {array_name[j]}")
        if j == 0:
            axs[j,i].set(title = f"{low_high[i]} tide")

# plt.savefig("Figures/Exercise_1-2a.png")

#%% Find the significant height and the rms height for the different low tides. Here is the signicant_height script used. At the first location
test = np.array([1,1,2,2,3,3])

from significant_height import significant_height
from significant_height import zerocross

from significant_height import rms_height

print(significant_height(zerocross(low[:,0])))

print(rms_height(zerocross(low[:,0])))


#%% For all the locations
Hm_tot = np.zeros(np.shape(low[1]))
H13_tot = np.zeros(np.shape(low[1]))
Hrms_tot = np.zeros(np.shape(low[1]))

for i in range(np.shape(low)[1]):
        Hm_tot[i] = np.mean(zerocross(low[:,i]))
        H13_tot[i] = significant_height(zerocross(low[:,i]))
        Hrms_tot[i] = rms_height(zerocross(low[:,i]))

#%% Plot this together with the bed profile
loc = np.array([4478, 4765, 4790, 4814, 4835]) #Locations sensors
fig, axs = plt.subplots(2,1, sharex = True)

axs[0].scatter(loc, Hm_tot, label = "Mean")
axs[0].scatter(loc, H13_tot, label = "Significant")
axs[0].scatter(loc, Hrms_tot, label = "RMS")

axs[0].plot(loc, Hm_tot,'--')
axs[0].plot(loc, H13_tot, '--')
axs[0].plot(loc, Hrms_tot, '--')

axs[0].legend()

axs[1].plot(bed[:,0], bed[:,1])
plt.xlim(loc[0]-50, loc[-1]+50)

for el in loc:
```

```
    plt.vlines(el, np.min(bed[:,1]), np.max(bed[:,1]), color = 'red')

# plt.savefig("Figures/Exercise222a.png")

#%% Calculate also for the other tides
Hm_tot = np.zeros((3,np.size(low[1])))
H13_tot = np.zeros((3,np.size(low[1])))
Hrms_tot = np.zeros((3,np.size(low[1])))
data = np.array([low, mid, high])

for j in range(np.shape(data)[0]):
    for i in range(np.shape(low)[1]):
        Hm_tot[j,i] = np.mean(zerocross(data[j,:,i]))
        H13_tot[j,i] = significant_height(zerocross(data[j,:,i]))
        Hrms_tot[j,i] = rms_height(zerocross(data[j,:,i]))

#%% Plot those as well in tree different subplots with a trendline
title = np.array(["Low tide", "Mid tide", "High tide"])

fig, axs = plt.subplots(1,3, sharey = True, sharex = True)
xas = np.array([np.min(Hrms_tot), np.max(Hrms_tot)])
for i in range(np.shape(data)[0]):
    axs[i].scatter(Hrms_tot[i], H13_tot[i])
    fit = np.polyfit(Hrms_tot[i], H13_tot[i], 1)
    print(fit)
    axs[i].plot(xas, xas*fit[0]+fit[1], '--')
    axs[i].set_title(title[i])
axs[1].set_xlabel("Root mean square wave height (m)")
axs[0].set_ylabel("Significant wave height (m)")

plt.savefig("Figures/Exercise222b.png")

#%% As one plot with a trendline

plt.figure()
plt.scatter(Hrms_tot, H13_tot)
fit = np.polyfit(Hrms_tot.reshape(15), H13_tot.reshape(15), 1)
print(fit)
plt.plot(xas, xas*fit[0]+fit[1], '--')
plt.xlabel("Root mean square wave height (m)")
plt.ylabel("Significant wave height (m)")

print("Slope linear fit: ", fit[0]) #print values
print(r"2^(1/2): ", np.sqrt(2))

plt.savefig("Figures/Exercise222c.png")
#%% Also for the root mean squared error
plt.figure()
```

```python
plt.scatter(Hrms_tot, Hm_tot)
fit = np.polyfit(Hrms_tot.reshape(15), Hm_tot.reshape(15), 1)
print(fit)
plt.plot(xas, xas*fit[0]+fit[1], '--')

plt.xlabel("Root mean square wave height (m)")
plt.ylabel("Mean wave height (m)")

plt.savefig("Figures/Exercise222d.png")

print("Slope linear fit: ", fit[0])
print(r"Expected: ", 0.89)

#%% Store the found values in a new file.
stats = {'Hm_tot': Hm_tot, 'Hrms_tot': Hrms_tot, 'H13_tot': H13_tot}
path = "C:/Users/koenc/Desktop/School/UU/Jaar 6/Periode 3/Morphodynamics of wave dominated coasts/Python opdrachten/"

with open(path + 'StatisticsEgmond.pickle', 'wb') as f:
    pickle.dump(stats, f)
```