CEE 4350 Coastal Engineering

Problem Set 4

Zoe Maisel

Problem 1

Wave data in the form of a time series of surface elevation, η, was provided. Zero-crossings were calculated using a for-loop to find where the sign for η changed. It was found that there were 139 zero crossings, N_z. Then, the maximum of η was found between each peak. The entire dataset was analyzed.

The code above shows the calculation for the wave height and period metrics using the following equations:

$$N = N_z - 1$$

$$H_{\frac{1}{3}} = \frac{3}{N}\sum_{1}^{\frac{N}{3}} H_i$$

$$H_{rms} = \frac{1}{N}\sqrt{(\sum_{1}^{\frac{N}{3}} H_i^2)}$$

$$T_{\frac{1}{3}} = \frac{3}{N}\sum_{1}^{\frac{N}{3}} T_i$$

$$T_c = \frac{T_R}{N_z}$$

Significant wave height and wave period are reported in the following table. | | | | — | – | | $H_{\frac{1}{3}}$ (cm) | 4.88| | H_{rms} (cm) | 0.244 | | $T_{\frac{1}{3}}$ (s) | 1.08 | | T_c (s)| 0.719 |

A histogram of the wave heights was made using the wave data provided for analysis.

The supporting R code is shown below.

```
setwd("~/github/Coastal_Engineering")
library(zoo)

# Load in data from csv file
# Time (s) eta (cm?)
wavedata = read.csv("waverecord_HW4.csv")
wavedata$time_s = wavedata[,1]
```

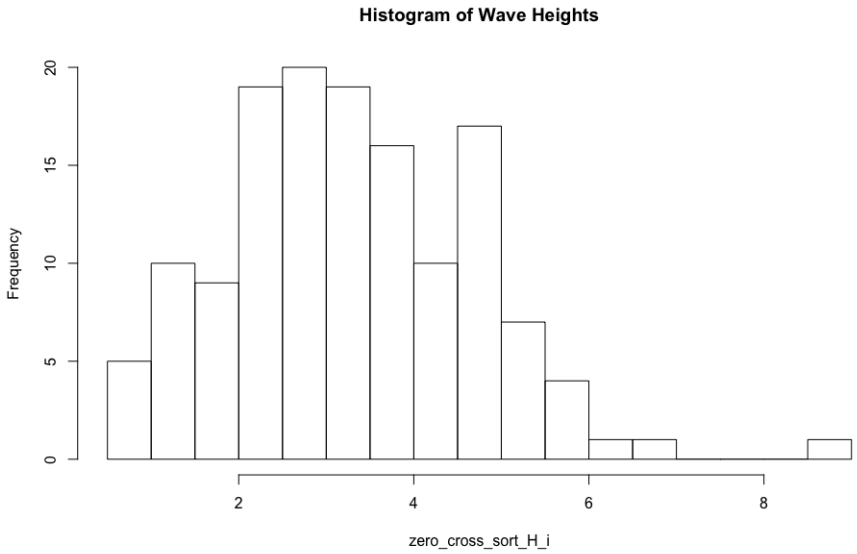


Figure 1:

```
wavedata$eta_cm = wavedata[,2]

Nz = 139
N = Nz - 1
N3 = N/3

zero_cross = data.frame(matrix(nrow = Nz, ncol = 0))
num_samples = 1000

# Find the time of zero crossings
# Find the max and min wave height between zero crossings
j = 0
m = 0
for (i in 1:num_samples)
{
  if ((wavedata$eta_cm[i] < 0) & (wavedata$eta_cm[i+1] > 0))
  {
    j = j+1
    zero_cross$time_s[j] = (wavedata$time_s[i] + wavedata$time_s[i+1])/2
    zero_cross$H_i[j] = max(wavedata$eta_cm[(m:i)]) - min(wavedata$eta_cm[(m:i)])
    zero_cross$T_i[j] = max(wavedata$time_s[(m:i)]) - min(wavedata$time_s[(m:i)]) # wave pe
    m = i
  }
```

```
}

# Sort the zero-crossings
zero_cross_sort_H_i = sort(zero_cross$H_i, decreasing = TRUE)
zero_cross_sort_T_i = sort(zero_cross$T_i, decreasing = TRUE)

# Calculate H_1/3
H_onethird = 3/N*sum(zero_cross_sort_H_i[1:N3])
# Calculate H_rms
H_rms = 1/N*sqrt(sum((zero_cross_sort_H_i[1:N3])^2))

# Calculate T_1/3 using zero crossings
T_onethird = 3/N*sum(zero_cross_sort_T_i[1:N3])

# Calculate T_c
# Assume that there are the same number maximum peaks as there are zero crossings
# On average, the period between the peaks will be the same,
# so they can be found by dividing the total time series by the number of zero crossings
T_c = max(wavedata$time_s)/Nz

# Make a histogram of wave heights
hist(zero_cross_sort_H_i, breaks = 20, main = "Histogram of Wave Heights")
```

Problem 2

The Pierson-Moskowitz frequency for wave energy spectrums was given as

$$S_{\eta\eta} = 0.205 H_{\frac{1}{3}}^2 T_{\frac{1}{3}}^{-4} f^{-5} exp[-0.75(T_{\frac{1}{3}} f)^{-4}], (m^2 s)$$

$S_{\eta\eta}$ was calculated using $H_{\frac{1}{3}}$ and $T_{\frac{1}{3}}$ found is Problem 1 as 4.88 cm and 1.08 s, respectively. A vector for f was made from 0.01 to 5 s with a step size of 0.01.

$S_{\eta\eta}$ was plotted against f and is shown in the following spectral plot.

The area under the curve, AUC, for $S_{\eta\eta}$ was calculated to find the following relationships.

$$a_{rms} = \sqrt{(2 * AUC)}$$
$$H_{rms} = 2 * a_{rms}$$
$$H_{max} = \sqrt{2} * H_{rms}$$
$$H_{\frac{1}{10}} = 20 * \frac{\sqrt{2} * H_{rms} * sin(\frac{\pi}{20})}{\pi}$$

$$H_{\frac{1}{3}} = 6 * \frac{\sqrt{2} * H_{rms} * sin(\frac{\pi}{6})}{\pi}$$

AUC	1.63
H_{rms} (cm)	3.609
H_{max} (cm)	5.103
$H_{\frac{1}{10}}$ (cm)	5.082
$H_{\frac{1}{3}}$ (cm)	4.87

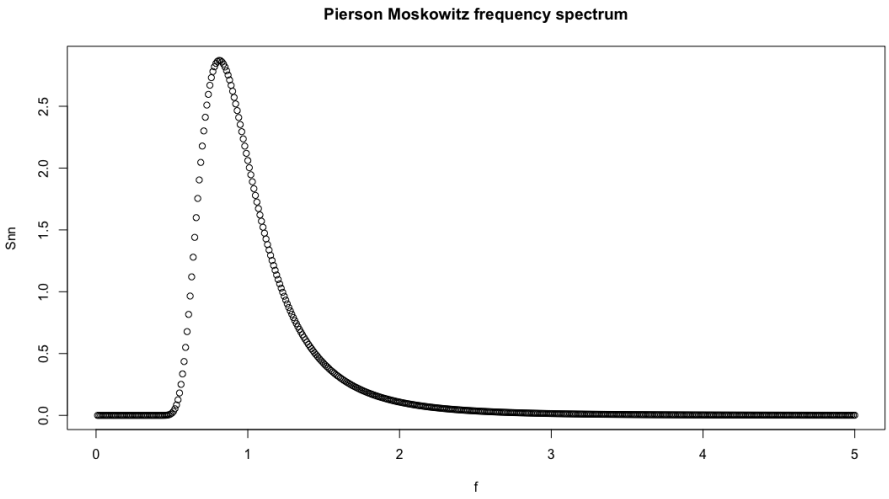
The maximum of $S_{\eta\eta}$ was determined to find the most energetic component of the total energy spectrum. The maximum $S_{\eta\eta} = 2.87$ and was found at $f = 0.81$.

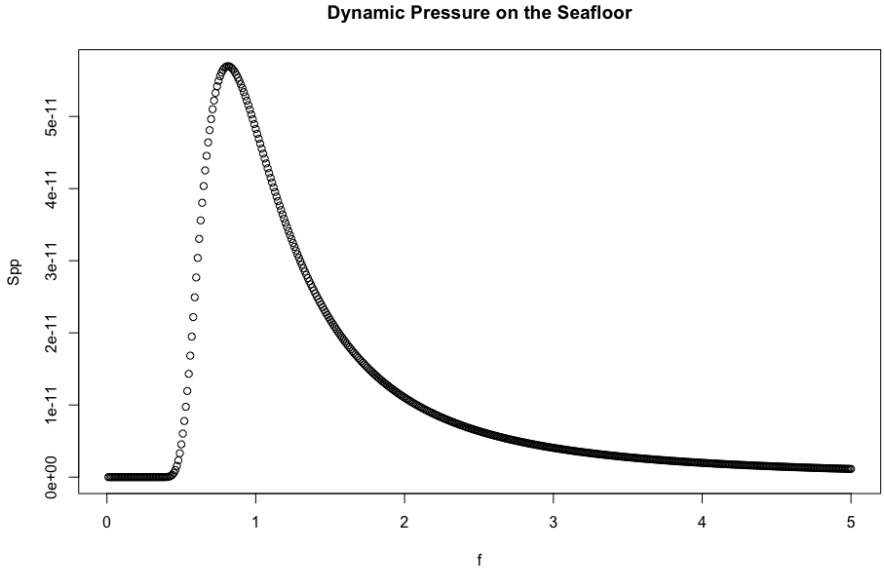
The dynamic pressure, S_{pp}, was calculated by

$$S_{pp} = \frac{\sqrt{(}S_{\eta\eta}) * g * rho}{cosh(kh)}$$

using $h = 10\ m$ and $k = 3.4\ m^{-1}$.

The following python and R code was used for analysis.






```
from aide_design.play import*
g = pc.gravity

def wavenumber(T, h):
  """Return the wavenumber of wave using period and water height from bed."""
  k = 10  # this is a guess to find what k is
  diff = (((2*np.pi)/T)**2)-(g.magnitude * k * np.tanh(k*h))
  while diff<0:
      LHS = ((2*np.pi)/T)**2
      RHS = g.magnitude * k * np.tanh(k*h)
      diff = LHS - RHS
      k = k - 0.0001
  return k

k = wavenumber(1.083, 10)
```

```
# Pierson Moskowitz frequency spectrum
f = seq(0.01, 5, by = 0.01)
Snn = 0.205 * H_onethird^2 * T_onethird^(-4) * f^(-5) * exp(-0.75 * (T_onethird * f)^(-4))
plot(f, Snn, main = "Pierson Moskowitz frequency spectrum")

# Calculate the area under the curve
id = order(f)
AUC = sum(diff(f[id])*rollmean(Snn[id],2))

# Calculate the variance to check that AUC is reasonable (it is)
```

```
# var = var(wavedata$eta_cm)

a_rms = sqrt(2*AUC)
H_rms2 = 2 * a_rms
H_max2 = sqrt(2)*H_rms2
H_onetenth2 = 20*sqrt(2)*H_rms2*sin(pi/20)/pi
H_onethird2 = 6*sqrt(2)*H_rms2*sin(pi/6)/pi

# Find maximum Snn to find the most energetic component of the total energy spectrum
Snn_max = max(Snn)

# Need to find the index of the max to find the wave frequency that it occurs at
i=0
for (i in 1:length(Snn))
{
  if (Snn[i] == Snn_max)
  {
    f_max = f[i]
  }
}

# Find dynamic pressure on seafloor
g = 9.81 # m/s
rho = 1000 # kg/m^3
h = 10 # m
k = 3.4 # calculated from T_onethird and h using a while loop and function in Python
Spp = sqrt(Snn)*g*rho/cosh(k*h)

plot(f, Spp, main = "Dynamic Pressure on the Seafloor")
```

Problem 3

A sloping bottom of a beach is described by the function

$$h(x) = s(x + \delta x_o exp[-\frac{(x - x_o)^2}{\lambda}])$$

Three different situations were: i) $s = 1/50$, $\delta = 0$ ii) $s = 1/50$, $\delta = 0.5$, $x_o =$ -1000 m, $\lambda = 5000$ m iii) $s = 1/50$, $\delta =$ -0.5, $x_o =$ -1000 m, $\lambda = 5000$ m

To find the waveray and wave amplitude, a for loop was written to evaluate the following equation at every distance from shore:

$$\Delta y = \frac{\kappa}{\sqrt{(k^2 + \kappa^2)}} \Delta x$$

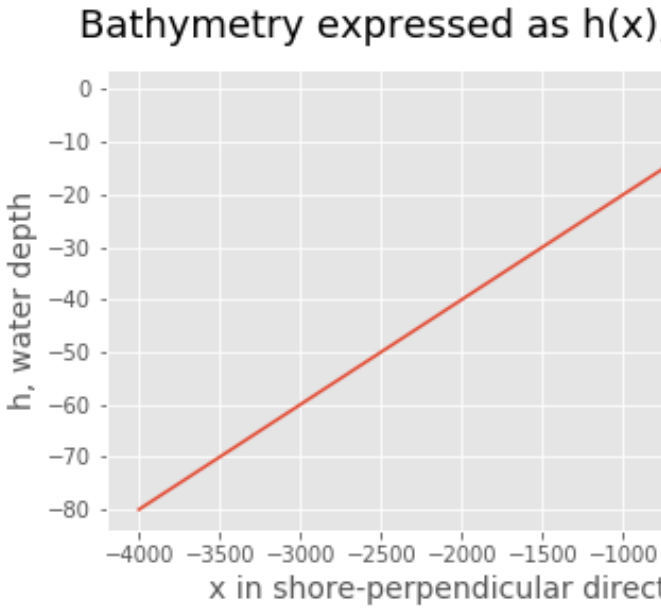
$$\alpha = tan^{-1}(\frac{\Delta y}{\Delta x})$$

$$C_g = \sqrt{(gh)}$$

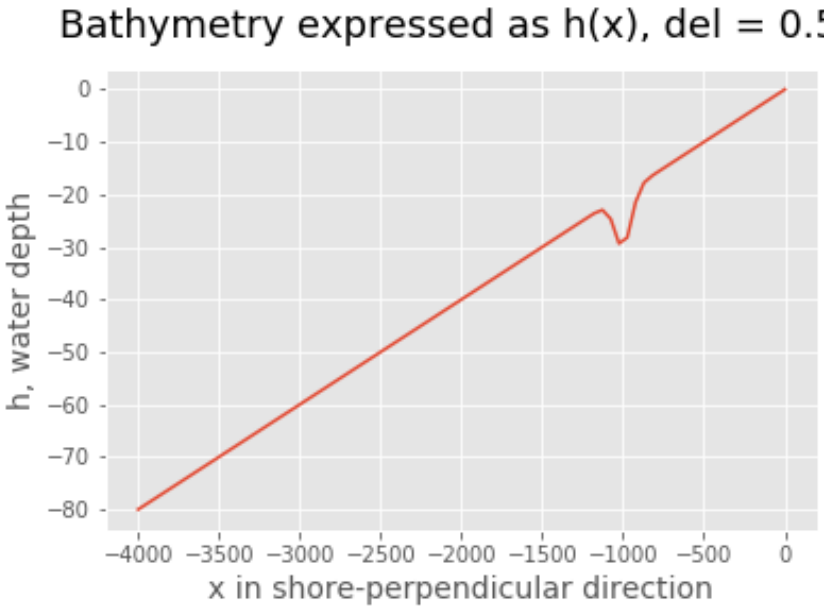
$$b = cos(\alpha)$$

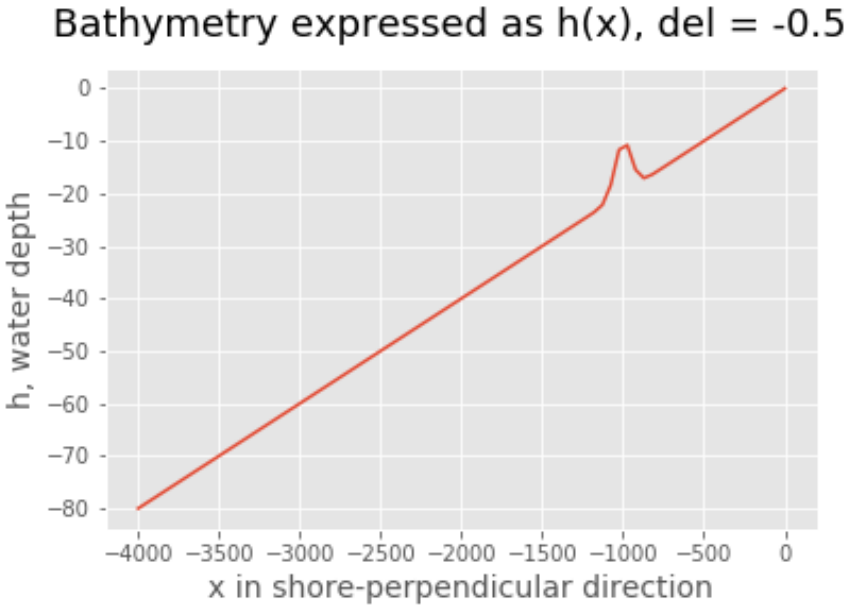
$$a = a_o\sqrt{(\frac{C_{go}}{C_g})}\sqrt{(\frac{b_o}{b})}$$

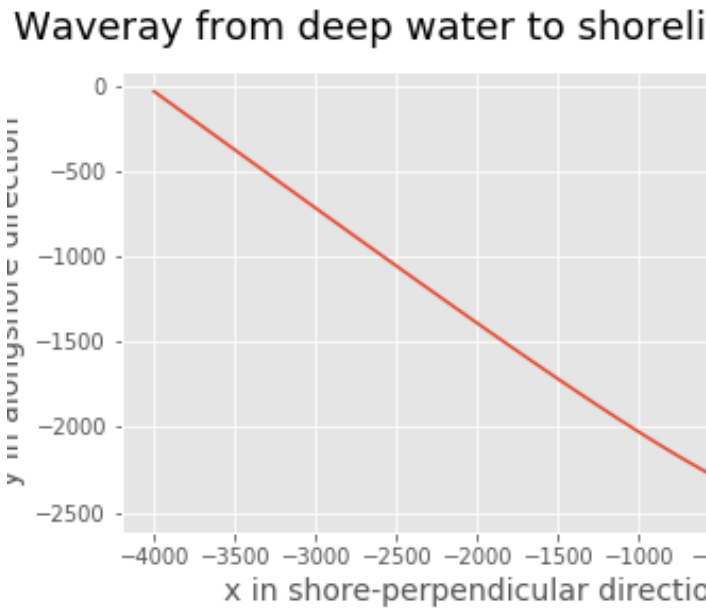
X ranges from -4000 to 0 m and will calculate different values for h and k which are used in calculating y and a. Iteratively looping through all of the x-values produces vectors for y and a which have been plotted for the three conditions.



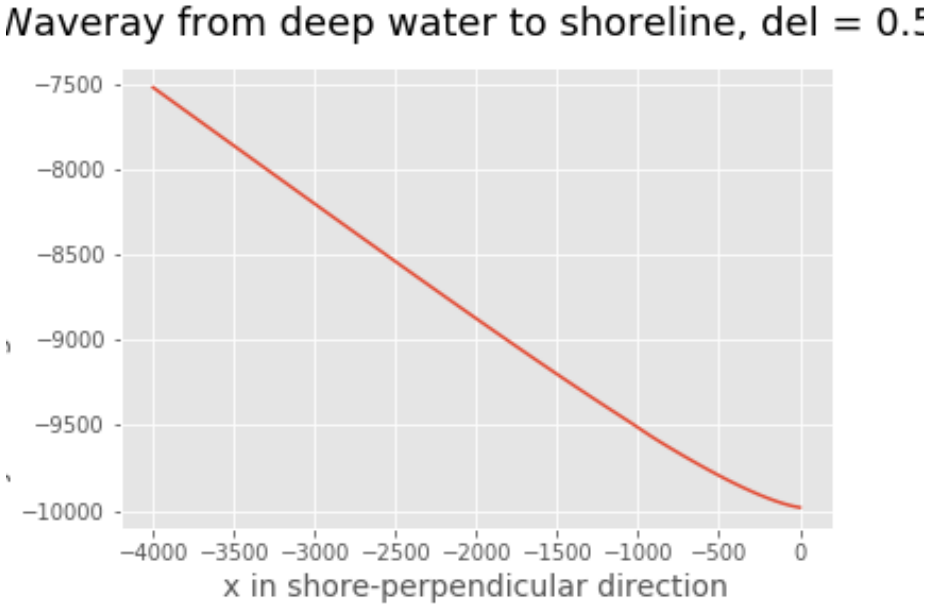
Bathymetry for the three conditions are shown below:

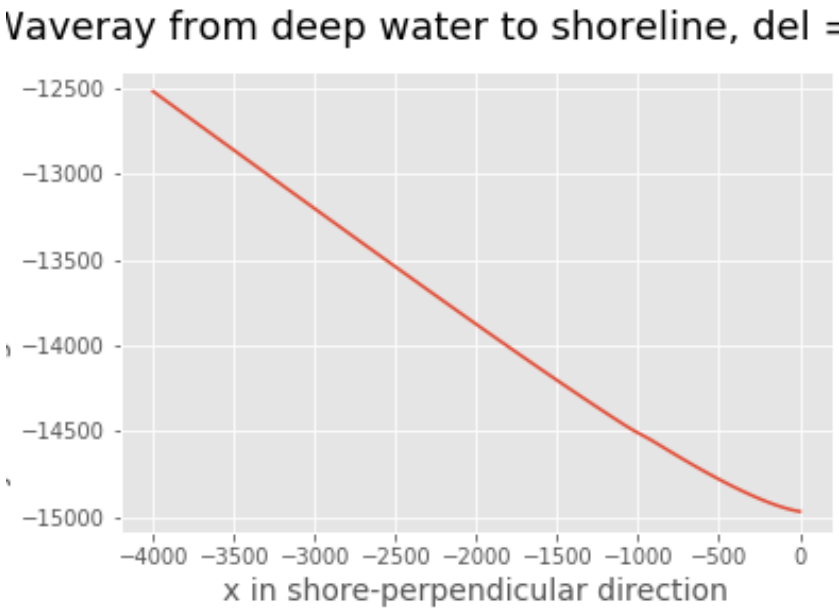

Bathymetry expressed as h(x), del = 0.5
h, water depth
0
−10
−20
−30
−40
−50
−60
−70
−80
−4000 −3500 −3000 −2500 −2000 −1500 −1000 −500 0
x in shore-perpendicular direction

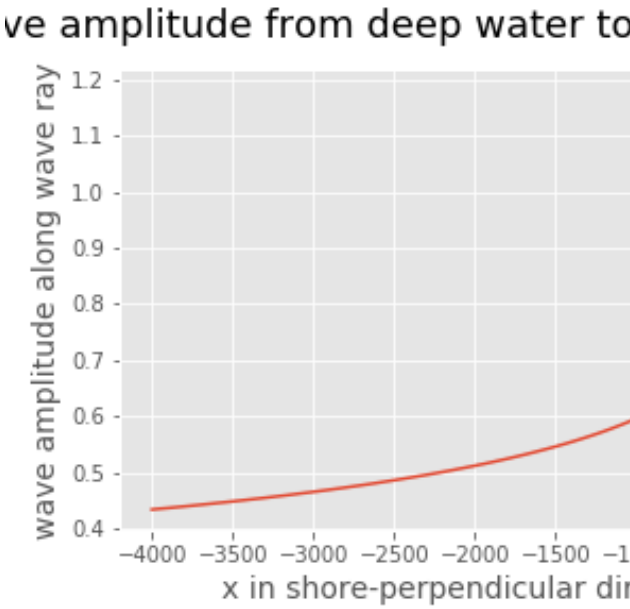

Bathymetry expressed as h(x), del = -0.5
h, water depth
0
−10
−20
−30
−40
−50
−60
−70
−80
−4000 −3500 −3000 −2500 −2000 −1500 −1000 −500 0
x in shore-perpendicular direction



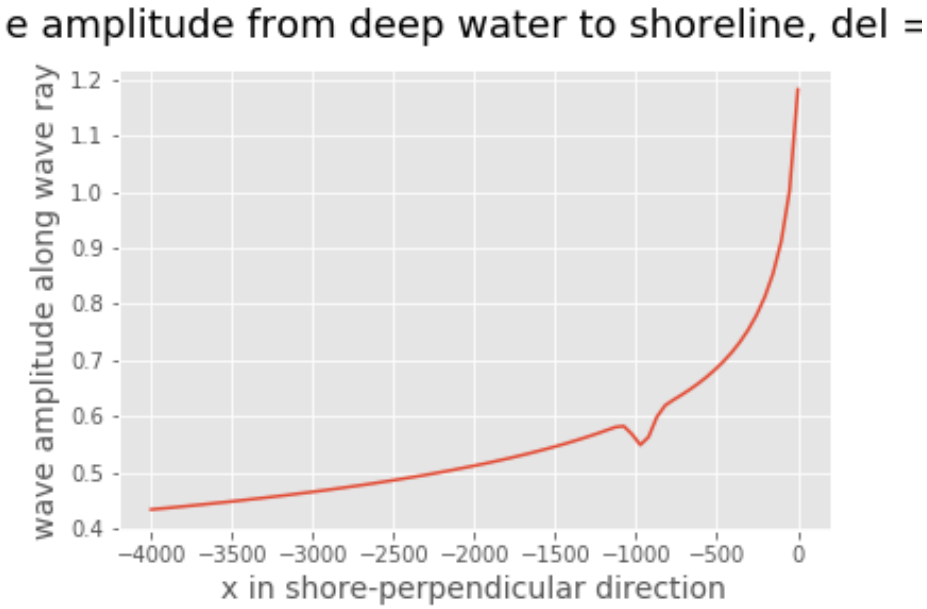
Waverays for the three conditions are shown below:

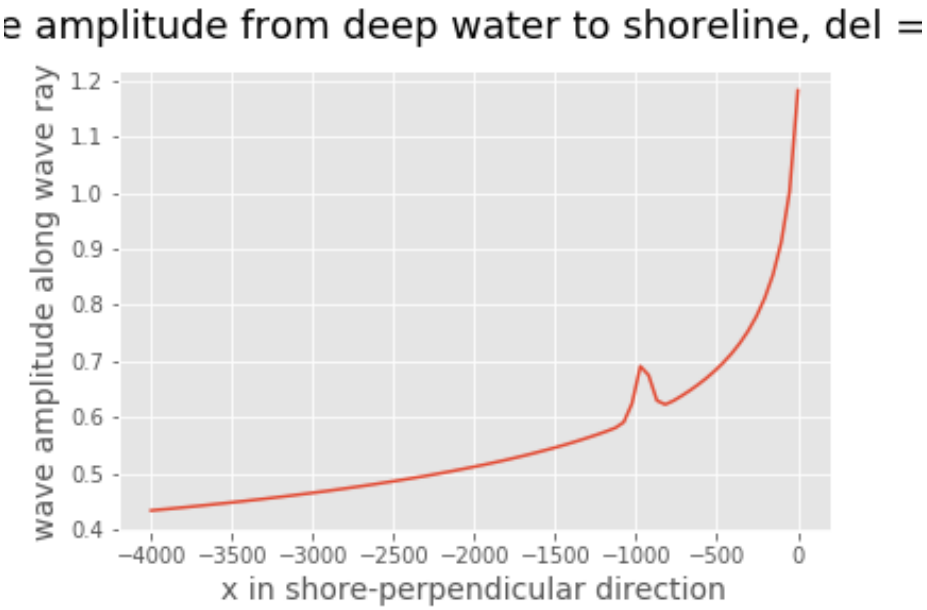






Wave amplitude for the three conditions are shown below:





The following python code was used for analysis.

```
from aide_design.play import*
import pandas as pd
g = pc.gravity
import timeit
```

```
def wavenumber(T, h):
  """Return the wavenumber of wave using period and water height from bed."""
  k = 10  # this is a guess to find what k is
  diff = (((2*np.pi)/T)**2)-(g.magnitude * k * np.tanh(k*h))
  while diff<0:
      LHS = ((2*np.pi)/T)**2
      RHS = g.magnitude * k * np.tanh(k*h)
      diff = LHS - RHS
      k = k - 0.0001
  return k

def h_x(x, _del, xo):
  """Return the water height from the bed to determine bathymetry."""
  h_x = slope*(x + _del * xo * np.exp((-(x-xo)**2)/lmbda))
  return h_x

del_x = 50
slope = 1/50
del_zero = 0
del_pos = 0.5
del_neg = -0.5
lmbda = 5000 #m
T = 10 #s
xo = -1000 #m

length = abs(4000/del_x)
print(length)
length_init = int(length - 1)
# Need to initalize all the values
del_y = np.ones(length_init)
del_y[0] = 0

y = np.ones(length_init)
y[0] = 0 # arbitrary coordinate assignment

alpha = np.zeros(length_init)
alpha[0] = 30 # deg

x = np.linspace(-4000, 0, num = 80)
x[0] = -4000 #m

h = np.ones(length_init)
h[0] = abs(h_x(x[0], del_zero, xo))

k = np.ones(length_init)
k[0] = wavenumber(T, abs(h_x(x[0], del_zero, xo)))
```

```
cg = np.zeros(length_init)
ho = abs(h_x(x[0], del_zero, xo))
cg[0] = np.sqrt(g.magnitude * ho)
cgo = np.sqrt(g.magnitude * ho)

b = np.zeros(length_init)
bo = np.cos(alpha[0])

a = np.zeros(length_init)
a[0] = 1 #m
ao = 1 #m

xplot = np.linspace(-4000, 1, num = 79)
plt.plot(xplot, h_x(xplot, del_zero, xo))
plt.xlabel('x in shore-perpendicular direction', fontsize=14)
plt.ylabel('h, water depth', fontsize=14)
plt.suptitle('Bathymetry expressed as h(x), del = 0', fontsize=18)
plt.savefig('bathydelzero.png')
plt.show()

xplot = np.linspace(-4000, 1, num = 79)
plt.plot(xplot, h_x(xplot, del_pos, xo))
plt.xlabel('x in shore-perpendicular direction', fontsize=14)
plt.ylabel('h, water depth', fontsize=14)
plt.suptitle('Bathymetry expressed as h(x), del = 0.5', fontsize=18)
plt.savefig('bathydelpos.png')
plt.show()

xplot = np.linspace(-4000, 1, num = 79)
plt.plot(xplot, h_x(xplot, del_neg, xo))
plt.xlabel('x in shore-perpendicular direction', fontsize=14)
plt.ylabel('h, water depth', fontsize=14)
plt.suptitle('Bathymetry expressed as h(x), del = -0.5', fontsize=18)
plt.savefig('bathydelneg.png')
plt.show()

# Calculate Kappa using initial k and alpha values
kappa = k[0] * np.sin(alpha[0])
print(kappa)

def bathymetry_wave(_del):
    for i in range(0, length_init):
        #start = time.time()
        #h[i] = abs(h_x(x[i], del_zero, xo))
        k[i] = wavenumber(T, abs(h_x(x[i], _del, xo)))
```

```
        del_y[i] = kappa / (np.sqrt(k[i]**2 + kappa**2)) * del_x
        y[i] = y[i-1] + del_y[i]
        alpha[i] = np.arctan(del_y[i] / del_x)
        cg[i] = np.sqrt(g.magnitude * abs(h_x(x[i], _del, xo)))
        b[i] = np.cos(alpha[i])
        a[i] = ao * np.sqrt(cgo/cg[i]) * np.sqrt(bo/b[i])
        #stop = time.time()
        #print(stop-start)
    return k, del_y, y, alpha, cg, b, a

xplot = np.linspace(-4000, 1, num = 79)

# del = 0 condition

y_zero = bathymetry_wave(del_zero)[2]
plt.plot(xplot, y_zero)
plt.xlabel('x in shore-perpendicular direction', fontsize=14)
plt.ylabel('y in alongshore direction', fontsize=14)
plt.suptitle('Waveray from deep water to shoreline, del = 0', fontsize=18)
plt.savefig('waveraydelzero.png')
plt.show()


a_zero = bathymetry_wave(del_zero)[6]
plt.plot(xplot, a_zero)
plt.xlabel('x in shore-perpendicular direction', fontsize=14)
plt.ylabel('wave amplitude along wave ray', fontsize=14)
plt.suptitle('Wave amplitude from deep water to shoreline, del = 0', fontsize=18)
plt.savefig('waveampdelzero.png')
plt.show()

# del = 0.5 condition
y_pos = bathymetry_wave(del_pos)[2]
plt.plot(xplot, y_pos)
plt.xlabel('x in shore-perpendicular direction', fontsize=14)
plt.ylabel('y in alongshore direction', fontsize=14)
plt.suptitle('Waveray from deep water to shoreline, del = 0.5', fontsize=18)
plt.savefig('waveraydelpos.png')
plt.show()

a_pos = bathymetry_wave(del_pos)[6]
plt.plot(xplot, a_pos)
plt.xlabel('x in shore-perpendicular direction', fontsize=14)
plt.ylabel('wave amplitude along wave ray', fontsize=14)
plt.suptitle('Wave amplitude from deep water to shoreline, del = 0.5', fontsize=18)
plt.savefig('waveampdelpos.png')
```

```
plt.show()

# del = -0.5 condition
y_neg = bathymetry_wave(del_neg)[2]
plt.plot(xplot, y_neg)
plt.xlabel('x in shore-perpendicular direction', fontsize=14)
plt.ylabel('y in alongshore direction', fontsize=14)
plt.suptitle('Waveray from deep water to shoreline, del = -0.5', fontsize=18)
plt.savefig('waveraydelneg.png')
plt.show()

a_neg = bathymetry_wave(del_neg)[6]
plt.plot(xplot, a_neg)
plt.xlabel('x in shore-perpendicular direction', fontsize=14)
plt.ylabel('wave amplitude along wave ray', fontsize=14)
plt.suptitle('Wave amplitude from deep water to shoreline, del = -0.5', fontsize=18)
plt.savefig('waveampdelneg.png')
plt.show()

# To convert the document from markdown to pdf
pandoc Problem_Set_4.md -o Maisel_Problem_Set_4.pdf
```