Quiz 2 Solution

Instructor: Trani

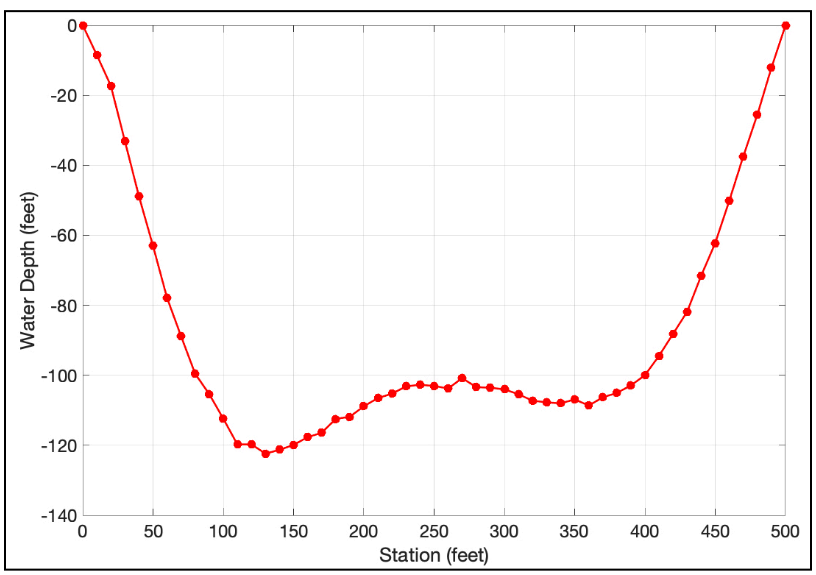
Problem 1

a) Script to plot the channel data.

```
% Script to Problem 1 - Quiz 2 - CEE 3804

% Read the channel data

load ('channelData.m')

% Data in the channel data file is of the type:

% Data for one channel depth
% Data measured using

% Column 1 = station (feet)
% Column 2 = water depth (feet)

%   0.000      0.0000
%   10.0000   -8.5101
%   20.0000  -17.3085
%   30.0000  -33.1293

% Plot the channel data

station = channelData(:,1);
waterDepth = channelData(:,2);

plot(station,waterDepth,'o--r')
xlabel('Station (feet)','fontsize',24)
ylabel('Water Depth (feet)','fontsize',24)
grid
```

Figure 1. Script to Read and Plot the Channel Depth vs. Station.

The script produces the following plot:



Parts (b and c) Function to calculate wetted perimeter and hydraulic radius

```
% Function to calculate the hydraulic radius

function [hydRadius, area, wettedPerimeter] = calculateHydRadius(station,waterDepth)

% Estimates the wetted perimeter (Wp)
% Add the distances between adjacent points along the edge of the channel
% station units are (feet)
% water depth units are (feet)

noPoints = length(station);           % detect number of points in the data
wettedPerimeter = 0;                  % Initialize a counter to estimate wetted perimeter

for i=1:noPoints-1                                      % noPoints-1 segments to add up in the data
    deltaStation = station(i+1) - station (i);          % distance along stations between successive points
    deltaDepth  = abs(waterDepth(i+1) - waterDepth(i)); % distance between water depth successive points
    distance = sqrt(deltaStation .^2 + deltaDepth.^2);  % slant distance between successive points

    % Calculate the cummulative distance between all points (wetted
    % perimeter)

    wettedPerimeter = wettedPerimeter + distance;       % adds each segment to calculate wetted perimeter

end

% Find the area of the open channel section
% Use numerical integration using the Trapezoidal rule

area = - trapz(station,waterDepth);          % Use the trapezoidal rule function (negative sign to avoid negative area)

hydRadius = area / wettedPerimeter;          % hydraulic radius - negative sign is to avoid negative area
```

The code to call the function is shown below. The function produced three outputs: area, hydraulic radius and wetted perimeter (parts b and c)

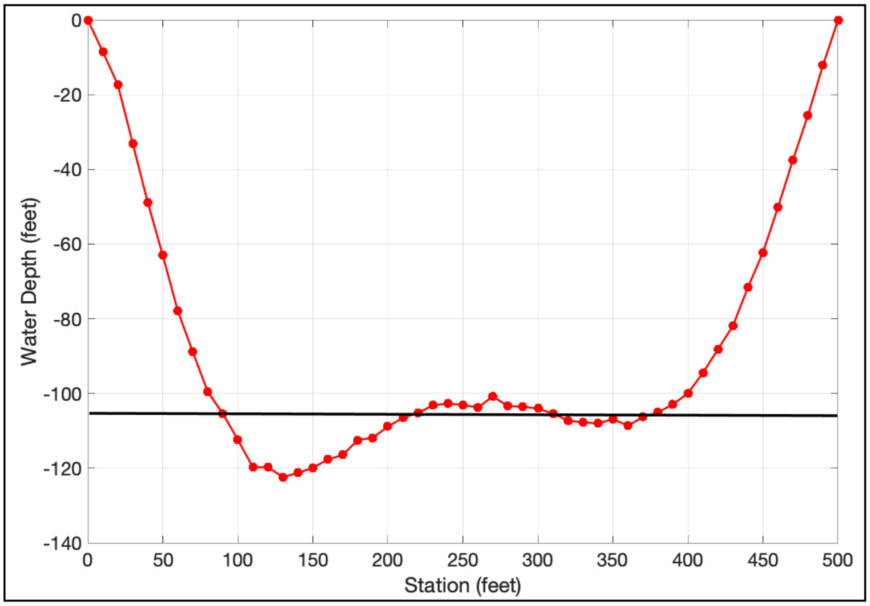
```
% Calculate the wettedPerimeter using the function created in Matlab

[hydRadius, area, wettedPerimeter] = calculateHydRadius(station,waterDepth);

disp(' ')
disp(['Wetted Perimeter (square feet)   ' , num2str(wettedPerimeter)])
disp(['Hydraulic Radius (feet)   ' , num2str(hydRadius)])
disp(['Area (square feet)   ' , num2str(area)])
```

The scrip and function produce the following results:

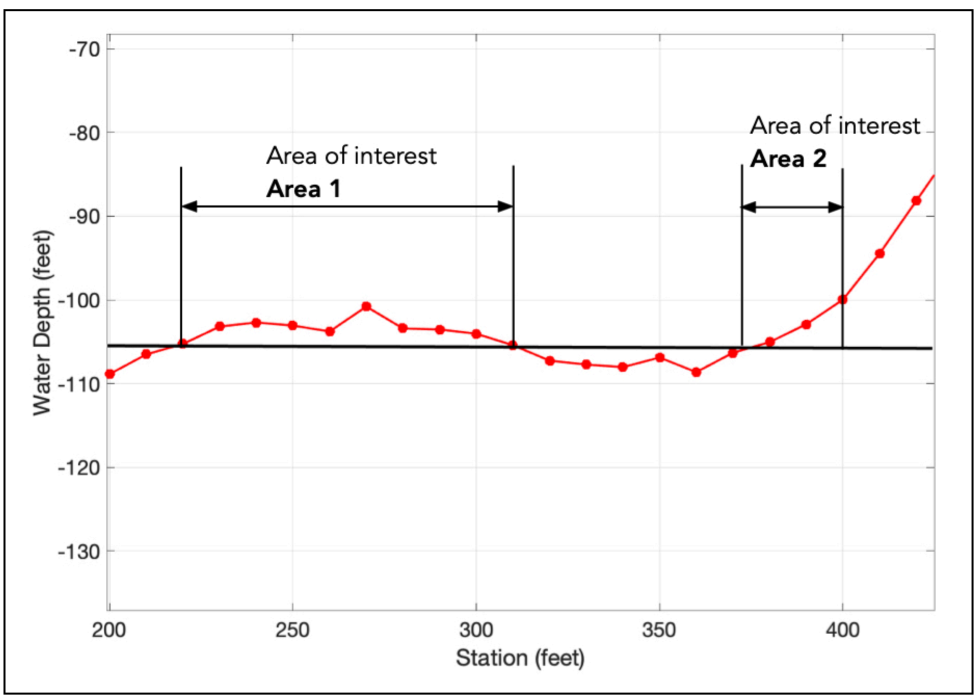
Wetted Perimeter (square feet)   602.594
Hydraulic Radius (feet)   73.7384
Area (square feet)   44434.311

Part (d)

Visualize the problem. I added a line at 106 feet to understand the magnitude of the material to be removed.



A little more detail is shown in the following plot. Note that we need to take material only in the central area of the channel where the water depth is less than 106 feet.



Code to identify stations whose water depth is more than -106 feet and with stations between 100 and 400 feet

```
% Part d

% Find stations below 106 feet and that lie in the central area
% I use three conditions: waterDepth > -106 feet ; station >= 100 feet and
% station <= 400 feet. The & sign is used to handle the AND condition

indicesOfLowStations       = find(waterDepth>-106 & station >= 100 & station <=400);
waterDepthLowStations    = waterDepth(indicesOfLowStations);
lowStations                = station(indicesOfLowStations);

figure
plot(lowStations ,waterDepthLowStations)
xlabel('Station (feet)','fontsize',24)
ylabel('Water Depth (feet)','fontsize',24)
grid
```

[indicesOfLowStations lowStations waterDepthLowStations]

```
23.0000  220.0000 -105.2547
24.0000  230.0000 -103.1904
25.0000  240.0000 -102.6910
26.0000  250.0000 -103.0718
27.0000  260.0000 -103.7714
28.0000  270.0000 -100.8011
29.0000  280.0000 -103.4098
30.0000  290.0000 -103.5368
31.0000  300.0000 -104.0457
32.0000  310.0000 -105.4174
39.0000  380.0000 -105.0440
40.0000  390.0000 -102.9111
41.0000  400.0000  -99.9631
```

Integrate in two intervals:

a) between stations 220 and 310 feet (10 elements of array)
b) between stations 380 and 400 feet (3 elements of array)

```
AreaOfInterest1 = - trapz(lowStations(1:10),waterDepthLowStations(1:10));     % area in sq. feet
AreaOfInterest2 = - trapz(lowStations(11:13),waterDepthLowStations(11:13));     % area in sq. feet

AreaUnder106ftLine_Area1 = 106*(lowStations(10)-lowStations(1));
AreaUnder106ftLine_Area2 = 106*(lowStations(13)-lowStations(11));

Area1 = AreaUnder106ftLine_Area1 - AreaOfInterest1;
Area2 = AreaUnder106ftLine_Area2 - AreaOfInterest2;

TotalArea = Area1 + Area2;

disp(' ')
disp(['Area 1 (square feet)   ' , num2str(Area1)])
disp(['Area 2 (square feet)   ' , num2str(Area2)])
disp(['Total Area (square feet)   ' , num2str(TotalArea)])
```

Area 1 (square feet) 241.4595
Area 2 (square feet) 65.8535
Total Area (square feet) 307.313

Problem 2

```
% Script to to calculate the bridge forces
clc

% Define the parameters of the problem

% wo = load (N/m)
% l = length (m)
% f = sag distance (m)

wo = 1e5;   % N/m
l = 300;     % meters
f = 20;       % meters

% Call the function to calculate the bridge forces

% H = Horizontal force (N)
% Vr = vertical reaction force (N)
% Tr = tension on the cable (N)

[H, Vr, Tr] = bridgeForces(wo,l,f);

% Display the results

disp(' ')
disp(['Horizontal force (N)   ' , num2str(H)])
disp(['Vertical Force (N)   ' , num2str(Vr)])
disp(['Tension (N)   ' , num2str(Tr)])
```

```
% Function to calculate the forces acting on the bridge

function [H, Vr, Tr] = bridgeForces(wo,l,f)

% wo = load (N/m)
% l = length (m)
% f = sag distance (m)

H     = wo .* l .^2 / (8 * f);                          % Horizontal force (N)
Vr    = wo .* l / 2;                                    % vertical reaction force (N)
Tr    = wo .* l / 2 * sqrt(1 + l .^2 / (16 * f .^2));   % tension on the cable (N)
```

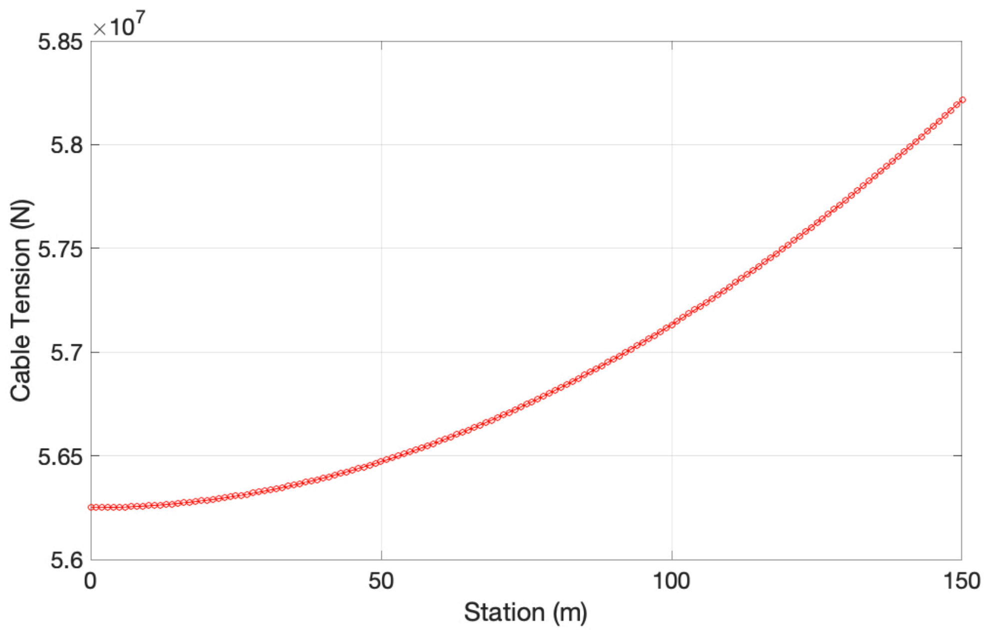
Horizontal force (N)   56,250,000

Vertical Force (N)   15,000,000

Tension (N)   58,215,655.111

```
x = 0:1:150;
% tension      = sqrt (H .^2 + wo .^2 .* x .^ 2);      % tension along the cable (N)
[tension] = tensionAlongSpan(H,wo,x);

plot(x,tension,'o-r')
xlabel('Station (m)','fontsize',24)
ylabel('Cable Tension (N)','fontsize',24)
grid
```

```
% Function to calculate the forces acting on the bridge

function [tension] = tensionAlongSpan(H,wo,x)

% wo = load (N/m)
% H = horizontal force (N)
% x = station along span (m)

tension      = sqrt (H .^2 + wo .^2 .* x .^ 2);      % tension along the cable (N)
```

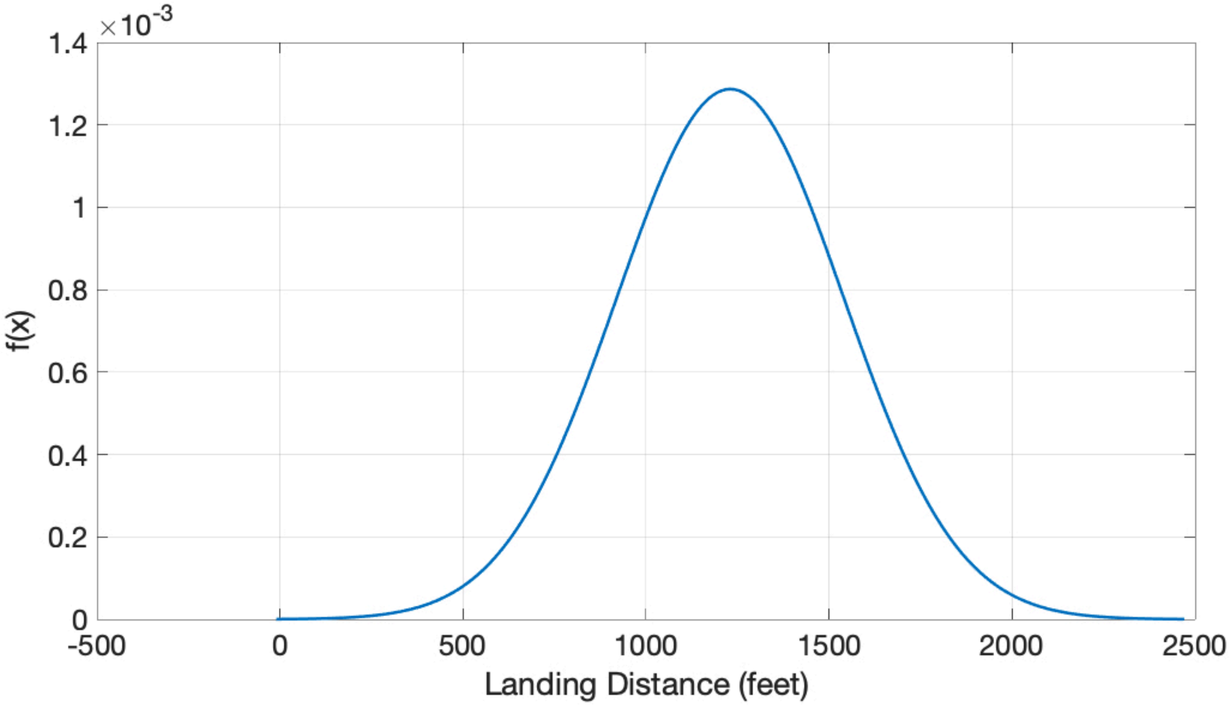


Problem 3

Function to calculate f(x) for normal distribution.

```
% Function to evaluate the normal distribution

% Passes back a single argument containing f(x) for the normal PDF function

function [fx] = normalDistribution(x, mu, sigma)

fx = 1./(sigma .* sqrt(2*pi)) .* exp(-1/2 .* ((x-mu)./sigma).^2 );
```

```
% Landing distance calculations using the normal distribution

% Define the parameter of the distribution (mu) and standard deviation (sigma)

mu      = 1230;  % mean of landing distance (feet)
sigma  = 310;    % stad. deviation of landing distance (feet)

% Define upper and lower bounds to get a nice plot of the Normal PDF

npoints = 200;
parameter = 4.0;
low = mu - parameter * sigma;
high = mu + parameter* sigma;
interval = (high - low) / npoints;

% Define random variable x

x=low:interval:high;

% Call the function of the random variable x (PDF function)

[fx] = normalDistribution(x, mu, sigma);

% Plot the random variable x versus the PDF function

plot(x,fx)
xlabel('Landing Distance (feet)','fontsize',24)
ylabel('f(x)','fontsize',24)
grid
```



Code to estimate the area under the curve between 365 and 2050 feet.

```
% Compute area under the curve f(x) from – 365 to 2050
% Change the values of the last two arguments if you
% want another range of values

landingDistances = 365:1:2050;        % Distances to be integrated
fx_landingdistances = normalDistribution(landingDistances, mu, sigma);

AreaUnderCurve=trapz(landingDistances,fx_landingdistances);

  disp(' ')
  disp([blanks(5),'Area under the Normal Distribution : '])
  disp(' ')
  disp([blanks(5),'     P(365 < x < 2050)=',num2str(AreaUnderCurve)])
  disp(' ')
```

Area under the Normal Distribution :

P(365 < x < 2050)=0.99328

Hence 1- 0.993 ~ 0.007 probability that landings happen outside the area.