ME560 Assignment #4

Group 14

Andrew Alferman
Nathan Schorn

12/3/16

Problem i:

Properties for water were selected at approximately 293K as follows:

Density (ρ) = 1000 kg/m^3

Viscosity (μ) = 1.0 * 10^{-3} Ns/m^2

The velocities selected were 0.01, 0.05, 1, and 5 m/s. Using the equation for Reynolds number,

$$Re = \frac{\rho V L}{\mu},$$

where L is the length of the plate and V is the velocity selected, the respective Reynolds numbers are 5.0*10^4, 2.5*10^5, 5.0*10^6, and 2.5 * 10^7 at the trailing edge of the plate. The first two of these values are laminar, and the last two are turbulent. The large size of the plate necessitates low velocities to have completely laminar flow across the entire plate.

The following equations were used to find the drag force of the plate (equations 12.20, 13.37, and 13.38 of Nuun):

To calculate drag force:

$$D_f = C_f \left(\frac{1}{2}\rho U_\infty^2 b L\right)$$

Where U_∞ is the velocity far from the plate (selected earlier) b is the width of the plate and L is the length of the plate. To calculate the laminar friction drag coefficient:

$$C_f = \frac{1.328}{Re_L^{\frac{1}{2}}}$$

To calculate the turbulent friction drag coefficient:

$$C_f = \frac{0.072}{Re_x^{\frac{1}{5}}}$$

To calculate the combined friction drag coefficient:

$$C_{f_c} = C_{f_{tt}} - \frac{A}{Re_l}$$

where:

$$A = Re_{crit}\left(C_{f_{tx}} - C_{f_{lx}}\right)$$

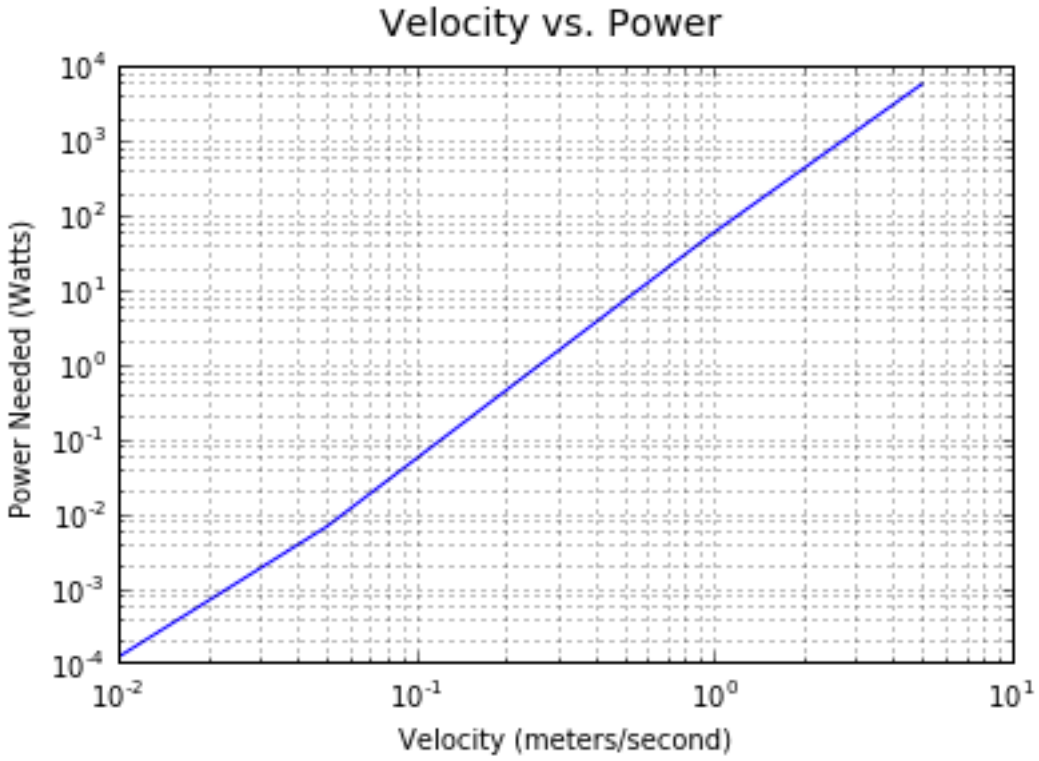
or combining the equations for laminar and turbulent friction drag coefficient, with a Re_{crit} of 5x10^5:

$$A = 1.328(Re_{crit})^{\frac{1}{2}}[0.056(Re_{crit})^{0.3} - 1.0] \approx 1750$$

Using the velocities given above, the drag forces were found to be 0.0119, 0.133, 58.8, and 1158 N, respectively. These values, and all values in this assignment, were calculated using the attached Python code.

Problem ii:

To find the power required to overcome drag, the forces found in problem i were simply multiplied by the velocity. The power required for each of the velocities (0.01, 0.05, 1, and 5 m/s) were 1.89*10^-4, 6.64*10^-3, 58.8, and 5790 W, respectively. A plot of the velocity vs power is as follows:



The plot demonstrates that the power needed is a function of the velocity cubed because the line has a slope on the log-log plot of 3.

Problem iii:

In order to meet the criteria of the problem of having two velocities before and two velocities after transition to the fully rough regimes, all for turbulent flow, four velocities were selected which were different than the velocities in problem i and ii. These velocities were 0.12, 0.2, 100.0, and 200 m/s, corresponding to Reynolds numbers of 6x10^5, 1x10^6, 5x10^8, and 1*10^9. The first two of these velocities needed to be low to maximize the differentiation between the different roughness values before transitioning into the fully rough regime. The second two velocities, though not physically realistic, were required to be extremely large (for travel through water, at least) in order for flow over the subject plate with length = 5m to enter the fully rough regime for a roughness value L/ε of 5x10^4. Entering this regime for the large

roughness value (corresponds to a relatively smooth plate) required a Reynolds number exceeding approximately 2*10^8, which normally can only be achieved (underwater) in practical applications with a flat plate much longer than 5m. For comparison, the Russian-built solid fuel rocket propelled supercavitating VA-111 Shkval torpedo, one of the fastest underwater vehicles ever constructed, may achieve a maximum speed in excess of 100 m/s, but almost certainly not approaching 200 m/s. The subject vehicle would likely be rapidly disassembled by the intense hydrodynamic forces in these conditions. Assuming this will not happen and that no additional effects due to cavitation or heating/vaporization of the fluid, etc. are present, we press on.

The force required to overcome drag was calculated identically to part i, however instead of solving for the coefficient of friction through the equations listed above, the coefficient was instead found by matching the Reynolds number and roughness value on the plot given. The roughness values were input into the attached Python code as follows:

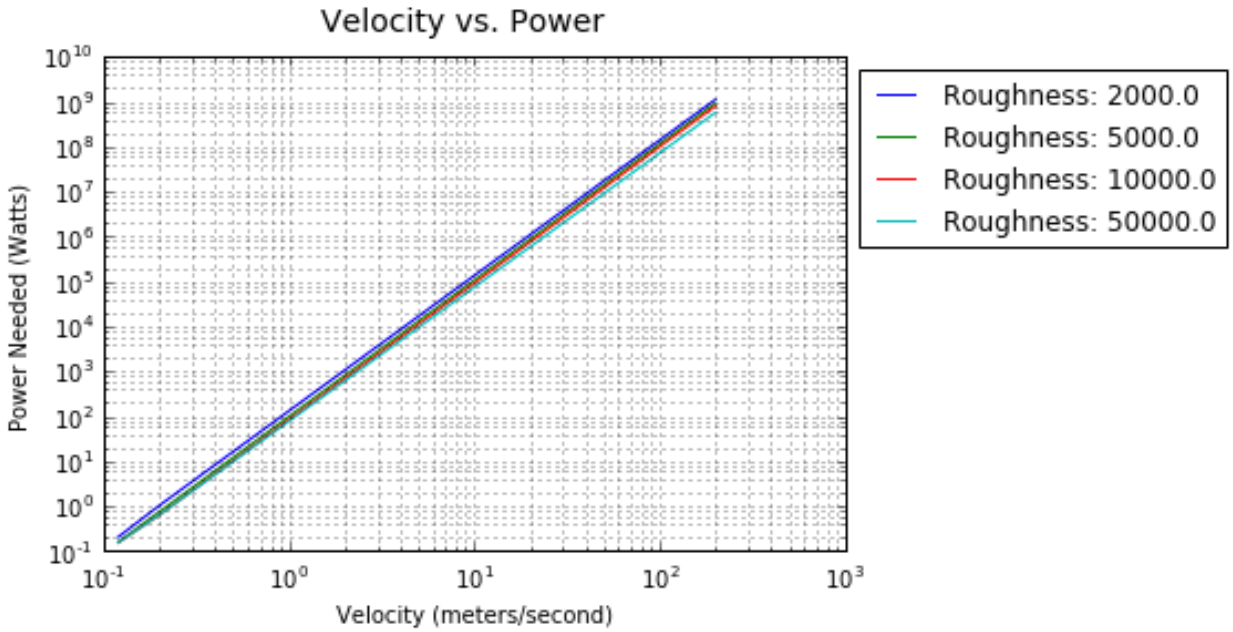
```
cds = [[0.0060, 0.0047, 0.0045, 0.0045], [0.0065, 0.0047, 0.004, 0.004],
    [0.007, 0.0058, 0.005, 0.0036], [0.007, 0.0058, 0.005, 0.0036]]
```

In this input, each list within the list represented all of the roughness values, in increasing order (and thus increasing smoothness), of a given velocity. For example, the 0.0065 value corresponded to a roughness value of 2000 and a Reynolds number of 1x10^6. Using the same methodology as problem ii, the following values for power were obtained, with the inside values having units of Watts:

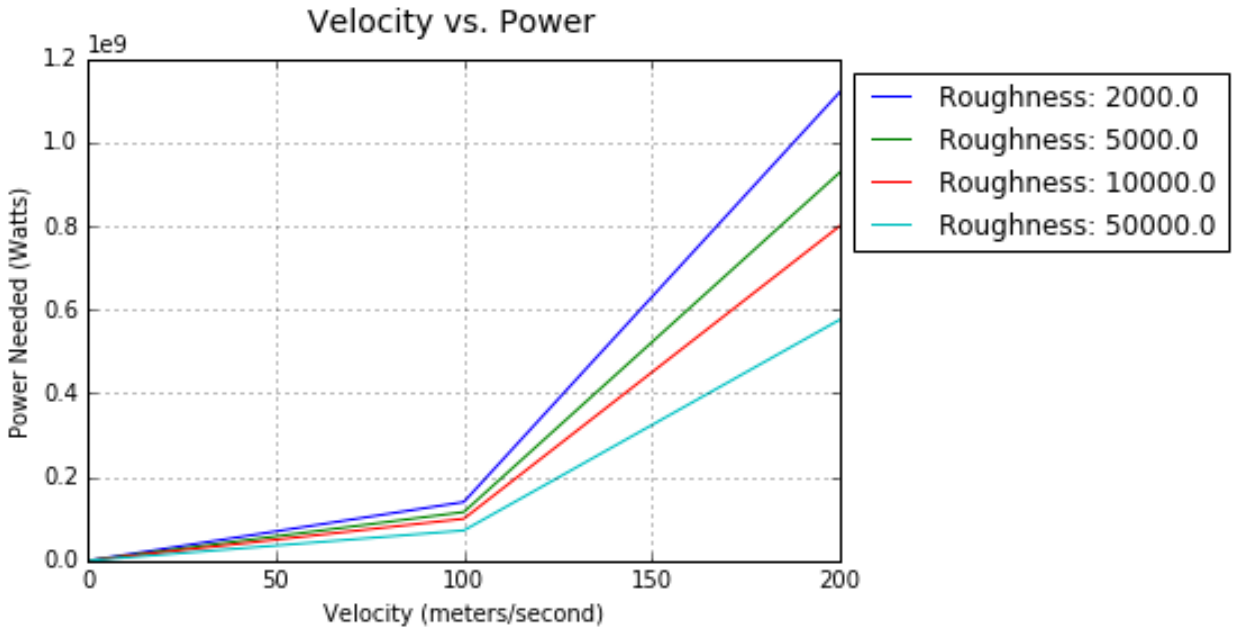
Power Required (Watts)

		Velocity (m/s)			
		0.12	0.20	100	200
Roughness Value	2000	0.2074	1.040	1.400x10^8	1.120x10^9
	5000	0.1624	0.7520	1.160x10^8	9.280x10^8
	1x10^4	0.1555	0.6400	1.000x10^8	8.000x10^8
	5x10^4	0.1555	0.6400	7.200x10^7	5.760x10^8

These values are also represented in the following log-log plot:



Unfortunately due to the nature of all of the force values having approximately the same slope on the log-log plot, the differences between each of the lines are difficult to see in this plot. Nonetheless, the velocity vs. power lines lie on this plot are three separate lines, as illustrated in the following linear plot of the same information:



This demonstrates that the different roughness values may increase the force and thus the required to overcome drag by a scalar factor, essentially by the ratio of a given coefficient to a different coefficient of friction if two different roughness values are

analyzed for a given Reynolds number. A smoother surface will result in less drag than a rougher surface. However, the power required to overcome drag still increases by the product of the velocity squared. It is noted that in the fully rough regime, the coefficient of drag is not a function of Reynolds number.

Problem iv:

In light of the enormous power values seen in problem iii, it is easy to conclude that the velocity should be lower to reduce power consumption. The following table shows the cost per 10 years to overcome the drag force for the subject plate in the given velocities:

Cost of 10 Year Operation ($)

		Velocity (m/s)			
		0.12	0.20	100	200
Roughness Value	2000	0.47	2.37	3.191x10^8	2.552x10^9
	5000	0.37	1.71	2.644x10^8	2.115x10^9
	1x10^4	0.35	1.46	2.279x10^8	1.823x10^9
	5x10^4	0.35	1.46	1.641x10^8	1.313x10^9

Note that at low velocities, the cost is the same between the two smoother plates. This is because the values of the coefficient of drag converge in Figure 7.6, and therefore it takes no more energy to power the plate for one value of smoothness versus another.

At higher velocities, the cost of smoothing the plate becomes insignificant compared to the operating cost, and even modest reductions can yield large savings. However, at low velocities, the operational cost is already low, and it is not worthwhile to smooth the plate. As the velocity increases, a balance point will be reached where the cost of smoothing the plate is equal to the cost savings of reduced drag force. By using the formulas shown in the writeup for problem i and iteratively guessing velocities, it was found that it would be worthwhile to smooth a plate with a roughness value of 5x10^4 when the velocity reaches approximately 5.80 m/s, a plate with a roughness value of 1x10^4 when the velocity reaches approximately 7.41m/s, a plate with a roughness value of 5000 when the velocity reaches approximately 8.56 m/s, and a plate with a roughness value of 2000 when the velocity reaches approximately 9.48 m/s. These values correspond to when the operational cost approaches the cost of the smoothing operation. The larger roughness values would be smoothed first because of the reduced cost in making the plate perfectly smooth versus a rougher plate; operational costs must be higher to justify the additional expenditure of smoothing a rougher surface.

v) $n = 8$, $C_1 = 8.74$, $m = \frac{n+1}{2}$, $C_2 = 2C_1^{\cdot 2m/m+1}$

$$C_f = 2\left(\frac{n}{(n+1)(n+2)}\right)\frac{d\delta}{dx}$$

$$C_f = \frac{C_2}{Re_\delta^{1/m}} \qquad \text{where} \qquad Re_\delta = (Re_x)\left(\frac{\delta}{x}\right)$$

equate and solve for $\delta(x)$

$$\frac{d\delta}{dx}\,2\left(\frac{n}{(n+1)(n+2)}\right) = \frac{C_2}{Re_\delta^{1/m}}$$

$$\frac{d\delta}{dx}\,2\,\frac{n}{(n+1)(n+2)} = \frac{C_2}{(Re_x)^{1/m}\left(\delta/x\right)^{1/m}}$$

$$\int_0^\delta \delta^{1/m}\,d\delta = \int_0^x \frac{(n+1)(n+2)\,C_2}{2n\left(\frac{\rho U}{\mu}\right)^{1/m}\left(\frac{x}{x}\right)^{1/m}}\,dx$$

$$\frac{1}{\frac{1}{m}+1}\,\delta^{\frac{1}{m}+1} = x\,\frac{(n+1)(n+2)\,C_2}{2n\left(\frac{\rho U}{\mu}\right)^{1/m}}$$

$$\delta(x) = x\left(\frac{(n+1)(n+2)\,C_2}{2n\left(\frac{\rho U}{\mu}\right)^{1/m}}\left(\frac{1}{m}+1\right)\right)^{\frac{m}{m+1}}$$

a) $\frac{\delta_1}{\delta} = \int_0^n \left(1-\frac{u}{U}\right) d\left(y/\delta\right) = \frac{1}{n+1} = \boxed{1/9}$

b) $\boxed{\delta(x) \text{ - see derivation}, \quad \delta_1 = \delta\,1/9}$

c) $$C_f = C_2\,\frac{1}{(Re_x)^{1/m}\left(\delta(x)/x\right)^{1/m}}$$

looking at only the denominator

$$(Re_x)^{1/m}\left(x^{\frac{m}{m+1}-1}\left(\frac{\rho U}{\nu}\right)^{-\frac{1}{m+1}}\left(\frac{(n+1)(n+2)\,C_{2m}}{2n(m+1)}\right)^{\frac{m}{m+1}}\right)^{1/m}$$

$$x^{\frac{m}{m+1}-1} = x^{\frac{m}{m+1}-\frac{m+1}{m+1}} = x^{-\frac{1}{m+1}}$$

$$(Re_x)^{1/m}\left((Re_x)^{-\frac{1}{m+1}}\left(\frac{(n+1)(n+2)\,E_{2m}}{2n(m+1)}\right)^{\frac{m}{m+1}}\right)^{1/m}$$

$$C_f = \frac{C_2}{(Re_x)^{\frac{1}{m}-\frac{1}{m(m+1)}}\left(\frac{(n+1)(n+2)\,C_{2m}}{2n(m+1)}\right)^{\frac{1}{m+1}}}$$

$$C_f = \boxed{\frac{0.0375}{Re_x^{2/11}}}$$

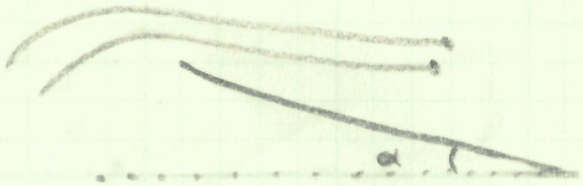
$$C_L = \frac{1}{L}\int_0^L C_f\,dx = \boxed{\frac{0.0458}{Re_x^{2/11}}}$$

$$D = C_L \tfrac{1}{2} \rho U^2 A$$

$$P = D \cdot U$$

vi)

$$\frac{C_f}{2} = \frac{d\delta_2}{dx} + \left(2 + \frac{\delta_1}{\delta_2}\right)\frac{\delta_2}{U}\frac{dU}{dx} \qquad U = U_0(1 - x\sin\alpha)$$



$$\frac{\delta_2}{\delta} = \int_0^1 \frac{u}{U}\left(1 - \frac{u}{U}\right) d(y/\delta) = \frac{n}{(n+1)(n+2)} \qquad n = 9$$

$$\frac{\delta_1}{\delta} = \int_0^1 \left(1 - \frac{u}{U}\right) d(y/\delta) = \frac{1}{n+1}$$

$$\delta_1 = \frac{\delta}{n+1}, \qquad \delta_2 = \frac{\delta n}{(n+1)(n+2)}$$

from part v

$$\delta(x) = \left(x \frac{(n+1)(n+2)C_2}{2n\left(\frac{\rho U}{\mu}\right)^{1/n}}\left(\frac{1}{n}+1\right)\right)^{n/n+1}$$

$$C_f = 2\left(\frac{d\delta}{dx}\frac{1}{n+1} + \left(2 + \left(\frac{n+2}{n}\right)\right)\frac{\delta(x)}{U}\frac{n}{(n+1)(n+2)}\frac{dU}{dx}\right)$$

Based on this relation as the U velocity increase it will still great more drag but with the angle of attack it should provide lift predendicular to the plate which will reduce the drag from a flat plate

Andrew Alferman
Nathan Schorn

ME560 Assignment #4

Problem (vii)

Given:

$$\frac{d(U^2\delta_2)}{dx} + \delta_1 U \frac{dU}{dx} = \frac{\tau_w}{\rho}$$

Show that:

$$\frac{c_f}{2} = \frac{d\delta_2}{dx} + \left(2 + \frac{\delta_1}{\delta_2}\right)\frac{\delta_2}{U}\frac{dU}{dx}$$

$c_f = \frac{\tau_0}{\frac{1}{2}\rho U_\infty^2}$; $\tau_0 \to \tau_w$ $U_\infty \to U$

$$\hookrightarrow \tau_w = \frac{c_f \rho U^2}{2} \rightarrow \frac{\tau_w}{\rho} = \frac{c_f U^2}{2}$$

$$\frac{dU^2\delta_2}{dx} = U^2\frac{d\delta_2}{dx} + \delta_2\frac{dU^2}{dx}$$ Using product rule

Substitute into original equation:

$$U^2\frac{d\delta_2}{dx} + \delta_2\frac{dU^2}{dx} + \delta_1 U\frac{dU}{dx} = \frac{c_f U^2}{2}$$

$$\frac{d\delta_2}{dx} + \frac{\delta_2}{U^2}\frac{dU^2}{dx} + \frac{\delta_1}{U}\frac{dU}{dx} = \frac{c_f}{2}$$

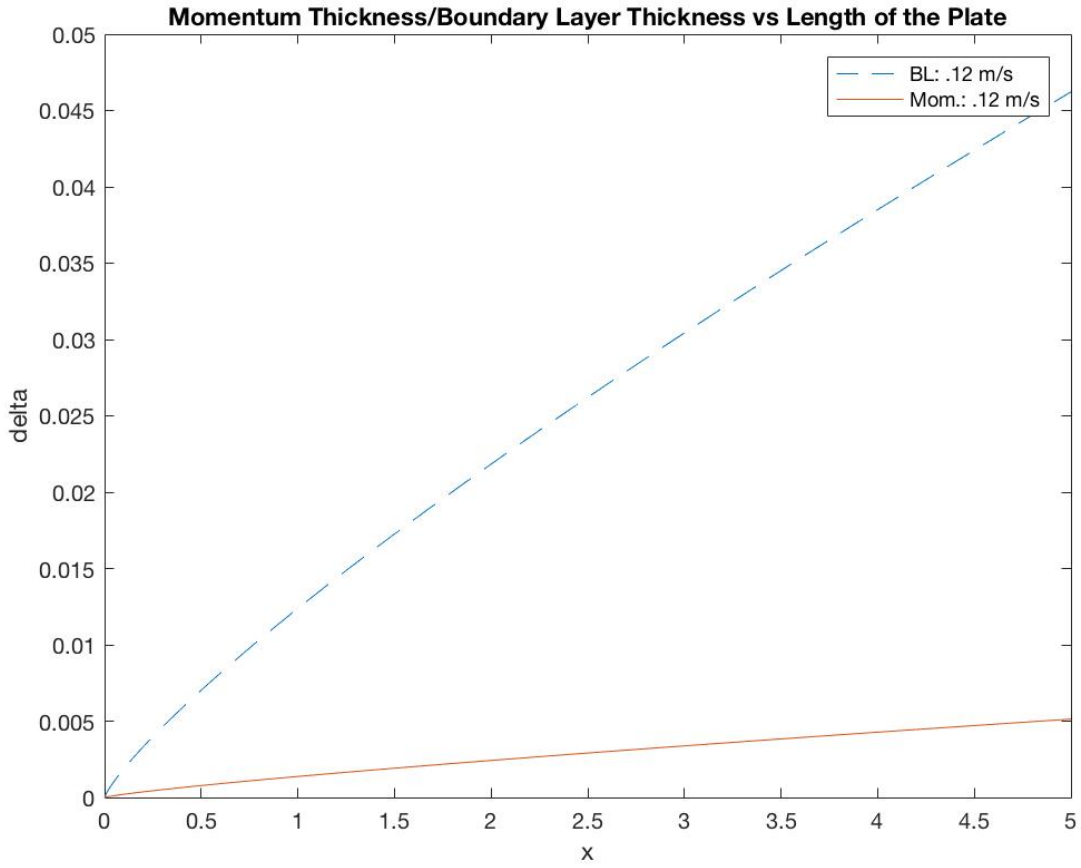
Factor out $\frac{\delta_2}{U}\frac{dU}{dx}$:

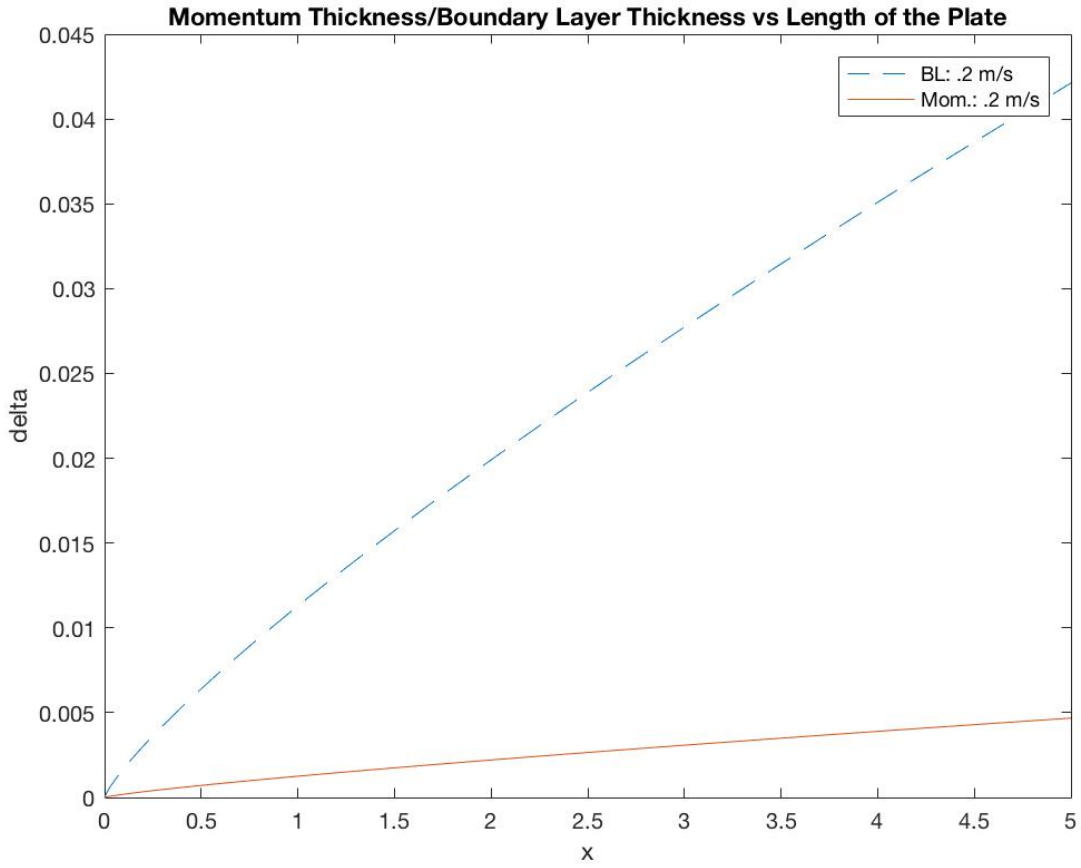
$$\frac{d\delta_2}{dx} + \left(\frac{1}{U}\frac{dU^2}{dx}\frac{dx}{dU} + \frac{\delta_1}{\delta_2}\right)\frac{\delta_2}{U}\frac{dU}{dx} = \frac{c_f}{2}$$

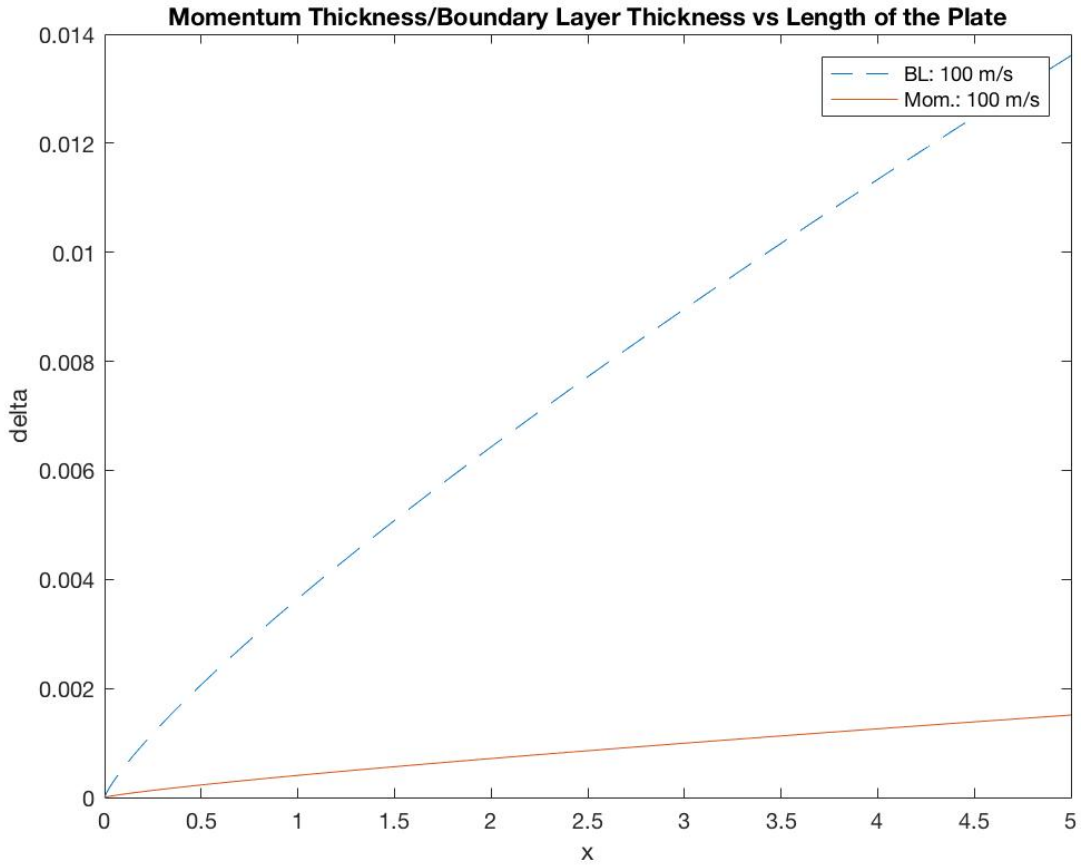
$$\frac{d\delta_2}{dx} + \left(\frac{1}{U}\frac{dU^2}{dU} + \frac{\delta_1}{\delta_2}\right)\frac{\delta_2}{U}\frac{dU}{dx} = \frac{c_f}{2}$$

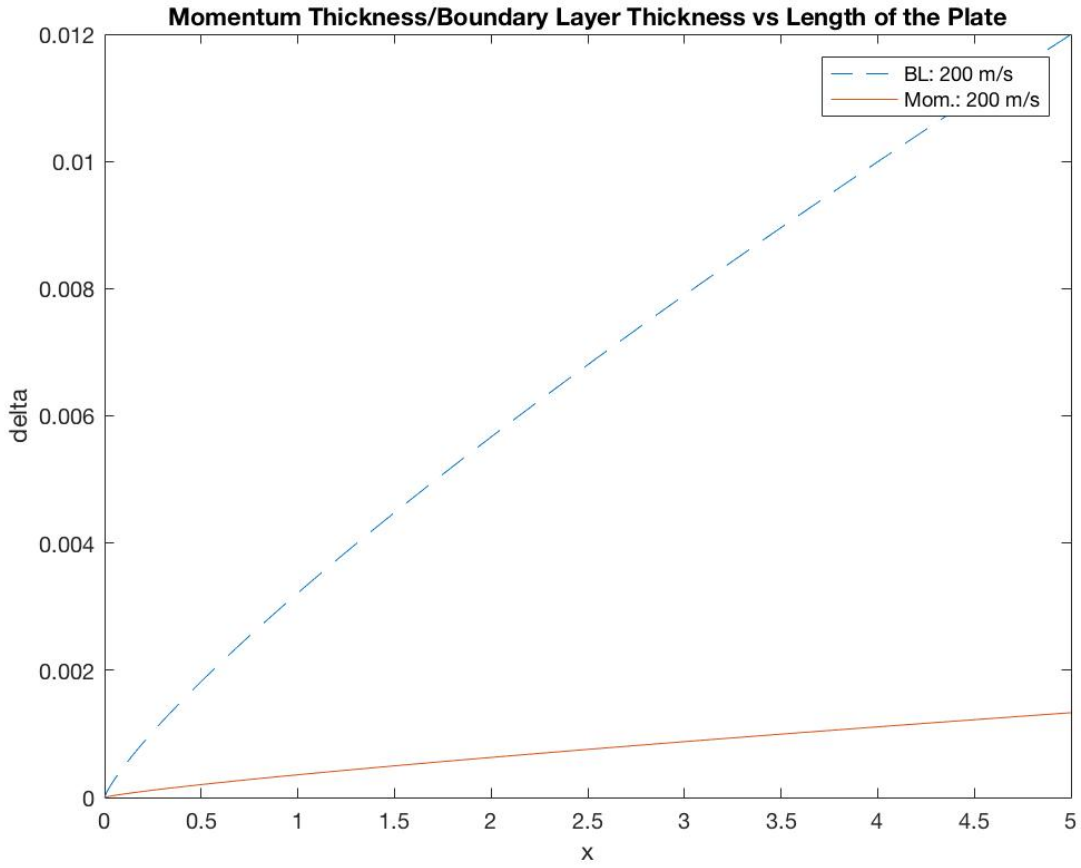
$$\frac{d\delta_2}{dx} + \left(\frac{1}{U}(2U) + \frac{\delta_1}{\delta_2}\right)\frac{\delta_2}{U}\frac{dU}{dx} = \frac{c_f}{2} \rightarrow \boxed{\frac{d\delta_2}{dx} + \left(2 + \frac{\delta_1}{\delta_2}\right)\frac{\delta_2}{U}\frac{dU}{dx} = \frac{c_f}{2}}$$

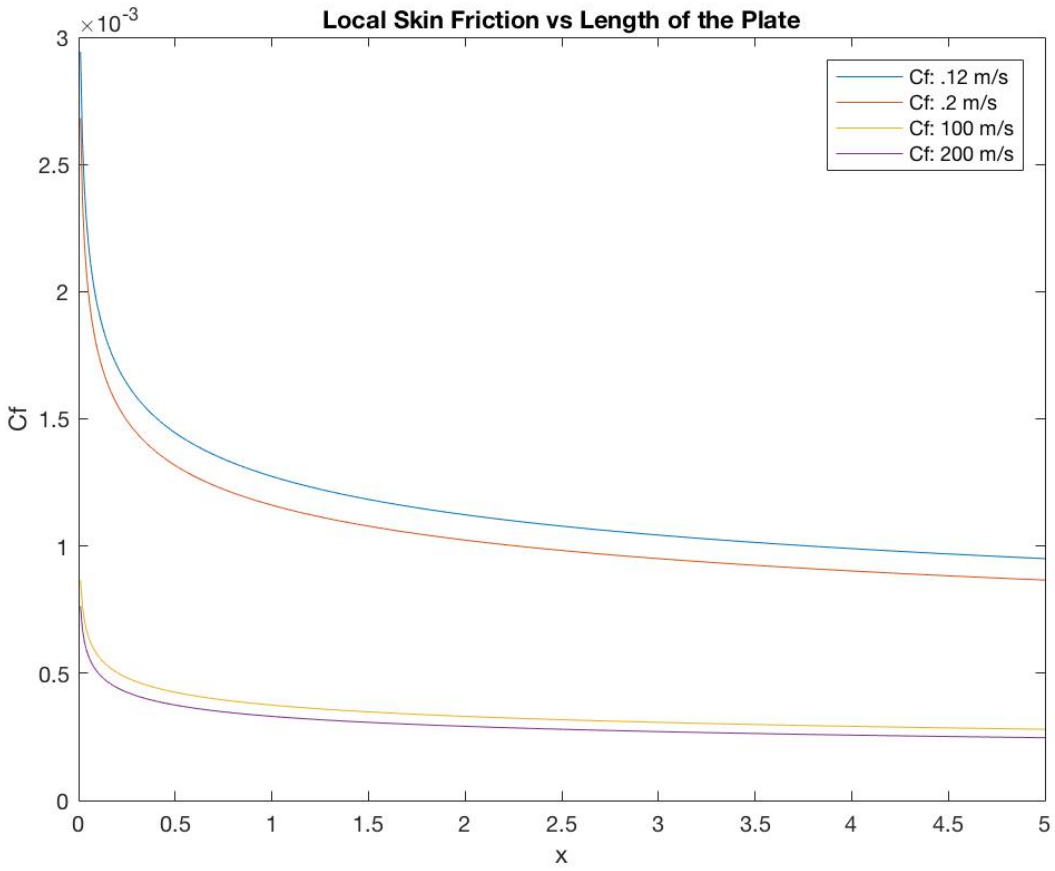
Plots for Problem v

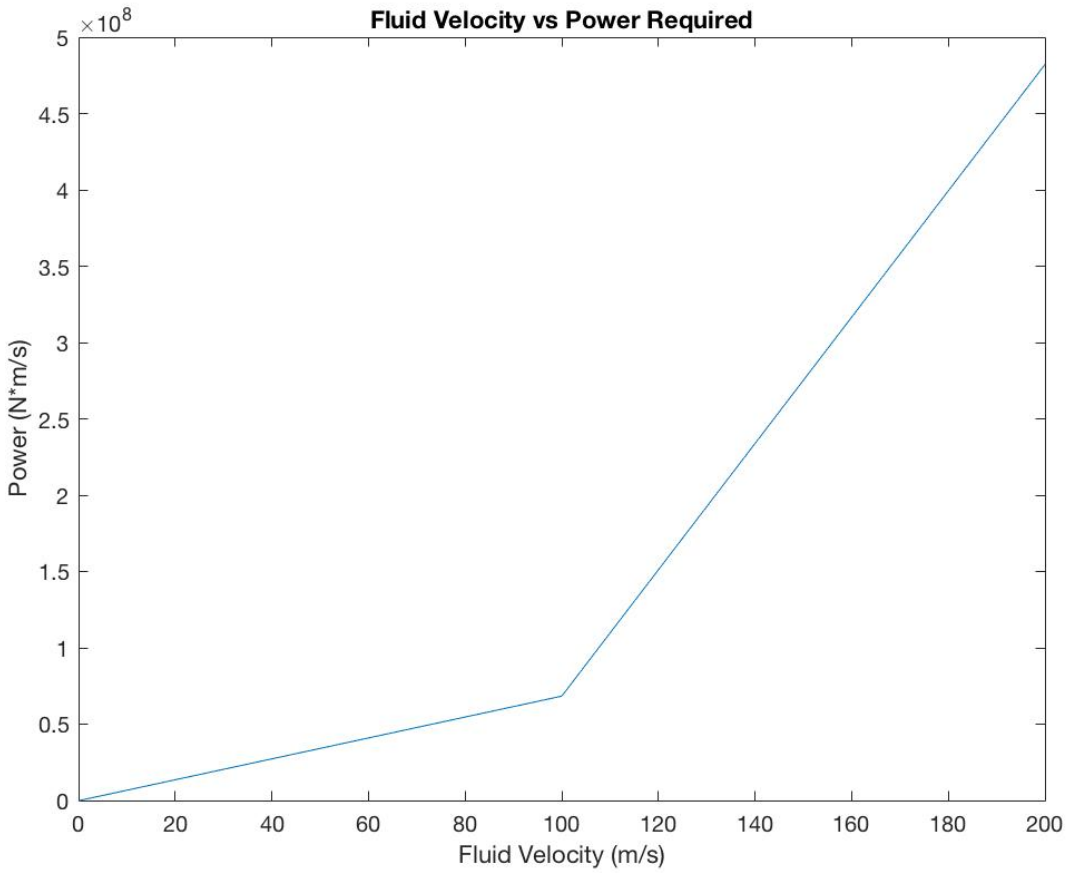

Momentum Thickness/Boundary Layer Thickness vs Length of the Plate
BL: .12 m/s
Mom.: .12 m/s
delta
x
0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5


Momentum Thickness/Boundary Layer Thickness vs Length of the Plate
BL: .2 m/s
Mom.: .2 m/s
delta
x
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5


Momentum Thickness/Boundary Layer Thickness vs Length of the Plate
BL: 100 m/s
Mom.: 100 m/s
delta
x
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5


Momentum Thickness/Boundary Layer Thickness vs Length of the Plate
BL: 200 m/s
Mom.: 200 m/s
delta
x
0.012
0.01
0.008
0.006
0.004
0.002
0
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5


Local Skin Friction vs Length of the Plate
×10^-3
Cf
x
Cf: .12 m/s
Cf: .2 m/s
Cf: 100 m/s
Cf: 200 m/s
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5


Fluid Velocity vs Power Required
×10^8
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
Power (N*m/s)
0
20
40
60
80
100
120
140
160
180
200
Fluid Velocity (m/s)

Code for Problems i – iv

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Dec  1 14:01:18 2016

@author: andrewalferman
"""

import numpy as np
import matplotlib.pyplot as plt

problem = 1

def reynoldsfun(properties, l, v):
    """Function that finds the reynolds number given fluid properties,
    length, and velocity"""
    rho, mu = properties
    return rho * v * l / mu


def laminarbigcf(reynolds):
    return 1.328 / np.sqrt(reynolds)


def turbbigcf(reynolds):
    return 0.072 / (reynolds**0.2)


def combinedcf(reynolds):
    Recrit = 5.e5
    A = 1.328 * (Recrit**0.5) * (0.056 * (Recrit**0.3) - 1.0)
    return turbbigcf(reynolds) - (A / reynolds)


def dragforce(dragparams):
    cf, rho, u_inf, b, L = dragparams
    return cf * 0.5 * rho * u_inf**2 * b * L


def critlength(v, properties):
    rho, mu = properties
    return (5.e5) * mu / (rho * v)


def findvel(reynolds, parameters, L):
    rho, mu = parameters
    return mu * reynolds / (rho * L)


# Note that all units are in meters, seconds, kg, etc.

# Initialize basic geometry of the problem
```

```
surflength = 5
surfwidth = 8
surfarea = surflength * surfwidth

# Fluid properties assumed at 20 degrees C
rho_water = 1000
mu_water = 1.e-3

reproperties = rho_water, mu_water

roughnesses = [2000., 5000., 1.e4, 5.e4]

hoursperday = 8

secondsperyear = hoursperday * 60 * 60 * 365. * 10

# Obtain velocity values
if problem == 1:
    #velocity = [0.01, 0.05, 1., 5.]
    velocity = [x*0.01 for x in range(1, 1000)]
else:
    # Picking some nice and easy values for the subsequent problems
    reynoldsvals = [6.e5, 1.e6, 5.e8, 1.e9]
    velocity = []
    for i in reynoldsvals:
        velocity.append(findvel(i, reproperties, surflength))
    # These values manually picked from the provided graph
    # Each row corresponds to a velocity, each column a roughness value
    cds = [[0.0060, 0.0047, 0.0045, 0.0045], [0.0065, 0.0047, 0.004, 0.004]
           [0.007, 0.0058, 0.005, 0.0036], [0.007, 0.0058, 0.005, 0.0036]]

# Initialize lists for saving values
etas, relist, drags, powers = [], [], [], []

# Calculate roughness values, just in case they are needed
roughnessvalues = [2000., 5000., 1.e4, 5.e4]
for e in roughnessvalues:
    etas.append(surflength/e)

# Calculate drag force for each velocity
initdragmatrix = True
findpower = True
for v in range(len(velocity)):
    # Shorthand for the velocity in this for loop
    velv = velocity[v]
    if problem == 1:
        # Re across at the trailing edge of the plate
        rev = reynoldsfun(reproperties, surflength, velv)
        relist.append(reynoldsfun(reproperties, surflength, velv))
        critl = critlength(velv, reproperties)
        if rev < 5.e5:
            cf = laminarbigcf(rev)
```

```
            else:
                cf = combinedcf(rev)
            dragparams = cf, rho_water, velv, surfwidth, surflength
            drags.append(dragforce(dragparams))
            powers.append(drags[v] * velv)
            if drags[v] * velv > (8775.77 * 4) and findpower == True:
                print('THE VELOCITY NEEDED IS:')
                print(velv)
                findpower = False
        else:
            if initdragmatrix == True:
                drags = np.zeros([len(velocity), len(roughnesses)])
                powers = np.zeros([len(velocity), len(roughnesses)])
                joulesperyear = np.zeros([len(velocity), len(roughnesses)])
                costperyear = np.zeros([len(velocity), len(roughnesses)])
                initdragmatrix = False
            for r in range(len(roughnesses)):
                cf = cds[v][r]
                dragparams = cf, rho_water, velv, surfwidth, surflength
                drags = np.array(drags)
                drags[r][v] = dragforce(dragparams)
                powers = np.array(powers)
                powers[r][v] = drags[r][v] * velv
                joulesperyear = np.array(joulesperyear)
                joulesperyear[r][v] = powers[r][v] * secondsperyear
                costperyear[r][v] = joulesperyear[r][v] * 2.168e-8

# Turn all the lists into arrays and format for nice printouts
relist = np.array(relist)
#etas = np.array(etas)
drags = np.array(drags)
powers = np.array(powers)
np.set_printoptions(formatter={'float': lambda x: format(x, '6.3E')})

#relist = ['%.2f' % elem for elem in relist]
#etas = ['%.2f' % elem for elem in etas]
#drags = ['%.2f' % elem for elem in drags]
#powers = ['%.2f' % elem for elem in powers]

"""
print('Velocities:')
print(velocity)
print('Reynolds Values:')
if problem == 1:
    print(relist)
else:
    print(reynoldsvals)
print('Drag Forces:')
print(drags)
print('Power Required:')
print(powers)
"""
```

```
if problem == 1:
    fig = plt.figure()
    ax = plt.gca()
    ax.set_xscale('log')
    ax.set_yscale('log')
    fig.suptitle('Velocity vs. Power', fontsize = 14)
    plt.xlabel('Velocity (meters/second)')
    plt.ylabel('Power Needed (Watts)')
    plt.plot(velocity, powers)
    plt.grid(b=True, which='both')
    plt.show()
else:
    print('Cost per Year:')
    print(costperyear)
    fig = plt.figure()
    ax = plt.gca()
    ax.set_xscale('log')
    ax.set_yscale('log')
    fig.suptitle('Velocity vs. Power', fontsize = 14)
    plt.xlabel('Velocity (meters/second)')
    plt.ylabel('Power Needed (Watts)')
    for r in range(len(roughnesses)):
        plt.plot(velocity, powers[r],
                 label='Roughness: {}'.format(roughnesses[r]))
    plt.grid(b=True, which='both')
    plt.legend(bbox_to_anchor=(1, 1), loc=2)
    plt.show()
```

Code for Problem v

```
clear
clc
close all

U= [.12 .2 100 200];
L= 5;
W= 8;
x=[0:.01:L];
n= 8;
m= (n+1)/2;
C1= 8.74;
C2= 2*C1^(-2*n/(n+1));
rho= 1000;
nu= 1e-6;

for i=1:length(U)
    delta= (x*(n+1)*(n+2)*C2/(2*n*(rho*U(i)/nu)^(1/m))*(1/m+1)).^(m/(m+1));
    deltam= delta/9;
    Rex= rho*U(i)*x/nu;
    Cf= .0375./Rex.^(2/11);
    Re(i)= rho*U(i)*L/nu;
    Cd(i)= .0458/Re(i)^(2/11);
    D(i)= Cd(i)*5*rho*U(i)^2*L*W;
    P(i)= D(i)*U(i);
    figure (i)
    plot(x,delta,'--')
    hold on
    plot(x,deltam)
    hold on
    xlabel('x')
    ylabel('delta')
    title('Momentum Thickness/Boundary Layer Thickness vs Length of the Plate')
    figure (5)
    plot(x,Cf)
    xlabel('x')
    ylabel('Cf')
    title('Local Skin Friction vs Length of the Plate')
    hold on
end
figure(1)
legend('BL: .12 m/s','Mom.: .12 m/s')
figure(2)
legend('BL: .2 m/s','Mom.: .2 m/s')
figure(3)
legend('BL: 100 m/s','Mom.: 100 m/s')
figure(4)
legend('BL: 200 m/s','Mom.: 200 m/s')
figure(5)
legend('Cf: .12 m/s','Cf: .2 m/s','Cf: 100 m/s','Cf: 200 m/s')
figure(6)
plot(U,P)
xlabel('Fluid Velocity (m/s)')
ylabel('Power (N*m/s)')
title('Fluid Velocity vs Power Required')
```