ME560 Intermediate Fluid Mechanics

Assignment #1 Due Oct. 11, 2016

Nathan Schorn:____________________
Andrew Alferman:____________________

1.) Start with the Navier-Stokes equations for an incompressible flow with constant viscosity. Write this equation out in tensor notation and nondimensionalize it using the scaling variable: U for velocity, $\frac{\mu U}{L}$ for pressure, $\frac{L}{U}$ for time.

Governing Equations:

Navier-Stokes - $$\rho \frac{Du_i}{Dt} = -\gamma \frac{\partial h}{\partial x_i} - \frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j}\right)\right] - \frac{2}{3}\frac{\partial}{\partial x_i}\left(\mu \frac{\partial u_m}{\partial x_m}\right)$$

Eqn. (10.17) from Nunn

N-S for constant viscosity and constant density -

$$\rho \frac{Du_i}{Dt} = -\gamma \frac{\partial h}{\partial x_i} - \frac{\partial p}{\partial x_i} + \mu \nabla^2 u_i$$

Eqn. (10.19) from Nunn

Assumptions:

Incompressible - ~~Pressure~~ Density is constant and Eqn. 10.19 can be used.

Tensor notation of (10.19):

$$\nabla^2 \rightarrow \frac{\partial^2}{\partial x_i \partial x_i}$$

$$\boxed{\rho \frac{Du_i}{Dt} = -\gamma \frac{\partial h}{\partial x_i} - \frac{\partial p}{\partial x_i} + \mu \frac{\partial^2}{\partial x_j \partial x_j} u_i} \qquad \gamma = \rho g$$

Nondimensionalize:

$$u_i^* = \frac{u_i}{U} \rightarrow u_i = u_i^* U$$

$$t^* = \frac{t}{\frac{L}{U}} = \frac{tU}{L} \rightarrow t = \frac{t^* L}{U}$$

$$p^* = \frac{p}{\frac{\mu U}{L}} = \frac{pL}{\mu U} \rightarrow p = \frac{p^* \mu U}{L}$$

$$h^* = \frac{h}{L} \rightarrow h = h^* L$$

$$x_i^* = \frac{x_i}{L} \rightarrow x_i = x_i^* L$$

$$x_j^* = \frac{x_j}{L} \rightarrow x_j = x_j^* L$$

$$\rho \frac{D(u_i^* U)}{D\left(\frac{t^* L}{U}\right)} = -\rho g \frac{\partial (h^* L)}{\partial (x_i^* L)} - \frac{\partial\left(\frac{p^* \mu U}{L}\right)}{\partial (x_i^* L)} + \mu \frac{\partial^2}{\partial (x_j^* L)\partial (x_j^* L)}(u_i^* U)$$

$$\frac{Du_i^*}{Dt^*} = -\frac{gL}{U^2}\frac{\partial h^*}{\partial x_i^*} - \frac{\mu}{\rho L U}\frac{\partial p^*}{\partial x_i^*} + \frac{\mu}{\rho L U}\frac{\partial^2}{\partial x_j^* \partial x_j^*} u_i^*$$

$$\boxed{\frac{Du_i^*}{Dt^*} = -\frac{1}{F_r^2}\frac{\partial h^*}{\partial x_i^*} - \frac{1}{Re}\frac{\partial p^*}{\partial x_i^*} + \frac{1}{Re}\frac{\partial^2}{\partial x_j^* \partial x_j^*} u_i^*}$$

2. Very viscous flow → Assume that forces due to viscosity are much greater than forces from other sources, therefore these other forces may be neglected without losing significant accuracy:

$\therefore -\frac{1}{Fr^2}\frac{\partial h^*}{\partial x_i^*}$ Term may be neglected.

Note that $P^* = \frac{PL}{\mu U}$ and therefore P^* becomes very small as μ becomes very large.

$$\frac{Du_i^*}{Dt^*} = \underbrace{\cancel{\frac{1}{Fr^2}\frac{\partial h^*}{\partial x_i^*}}}_{0} - \underbrace{\cancel{\frac{1}{Re}\frac{\partial P^*}{\partial x_i^*}}}_{0} + \frac{1}{Re}\frac{\partial^2}{\partial x_j^*\partial x_j^*}u_i^*$$

$$\frac{Du_i^*}{Dt^*} = \frac{1}{Re}\frac{\partial^2}{\partial x_j^*\partial x_j^*}u_i^*$$

3.a) Ball of $d = 0.1\ m$

Mass of $m = 1\ kg$

$\theta_0 = 45°$

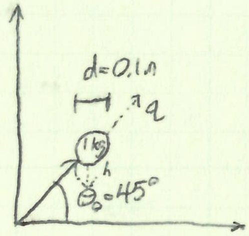
Flat terrain

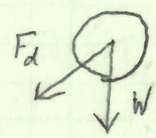
Assume $g = 9.8\ m/s^2$

Assume no wind

Assume constant drag coefficient

$\rho = 10 \frac{kg}{m^3}$





$mg = (1)(9.8) = 9.8\ N$

$\forall = \frac{4}{3}\pi r^3 = 0.000523\ m^3$

$F_{Buoyancy} = 10(0.000523) \ll 9.8$

$\therefore$ Neglect buoyancy

Governing Equations:

$\Sigma F = m \frac{dV}{dt}$

$\Sigma F = F_B + F_D$

Body — Surface (Drag)

$F_B = W = -mg \frac{\partial h}{\partial x_i}$

$F_D = -\frac{1}{2}\rho V^2 C_D A \frac{\partial q_i}{\partial x_i}$

$A = \frac{\pi}{4} D^2$

$F_D = -\frac{\pi}{8}\rho V^2 C_D D^2 \frac{\partial q_i}{\partial x_i}$

h: Vector pointing in "down" direction

q: Vector pointing in direction of travel through fluid.

u: Velocity vector

$$-mg\frac{\partial h}{\partial x_i} - \frac{\pi}{8}\rho V^2 C_D D^2 \frac{\partial q_i}{\partial x_i} = m\frac{du}{dt}$$

Nondimensionalize:

$h^* = \frac{h}{D} \longrightarrow h = h^* D$

$x_i^* = \frac{x_i}{D} \longrightarrow x_i^* D = x_i$

$q_i^* = \frac{q_i}{D} \longrightarrow q_i = q_i^* D$

$V^* = \frac{V}{V_0} \longrightarrow V = V^* V_0$

$u_i^* = \frac{u_i}{V_0} \longrightarrow u_i = u_i^* V_0$

$t^* = \frac{t}{\frac{D}{V_0}} = \frac{t V_0}{D} \rightarrow t = \frac{t^* D}{V_0}$

$$-mg\frac{\partial(h^* D)}{\partial(x_i^* D)} - \frac{\pi}{8}\rho (V^* V_0)^2 C_D D^2 \frac{\partial(q_i^* D)}{\partial(x_i^* D)} = m\frac{d(u_i^* V_0)}{d\left(\frac{t^* D}{V_0}\right)}$$

$$\boxed{-\frac{gD}{V_0^2}\frac{\partial h^*}{\partial x_i^*} - \frac{\pi \rho C_D D^3}{8m} V^{*2} \frac{\partial q_i^*}{\partial x_i^*} = \frac{du_i^*}{dt^*}}$$

3.b.) $$-\frac{gD}{V_0^2}\frac{\partial h^*}{\partial x_i^*} - \frac{\pi\rho C_D D^3}{8m} V^{*^2} \frac{\partial a_i^*}{\partial x_i^*} = \frac{du_i^*}{dt^*}$$

$P_1 := \frac{\pi\rho C_D D^3}{8m}$ Represents the resistance of the fluid (drag) on the ball. Ratio of the fluid's inertia over the ball's inertia.

$P_2 := \frac{gD}{V_0^2} = \frac{1}{Fr^2}$ Inverse of Froude's number squared.

c.) $x_1 \rightarrow x$ axis
$x_2 \rightarrow y$ axis
$x_3 \rightarrow z$ axis

$$\frac{\partial h_1}{\partial x_1} = \cos 90^\circ = 0 \quad\quad \frac{\partial h_2}{\partial x_2} = \sin 90^\circ = 1$$

$$\frac{\partial a_1}{\partial x_1} = \cos\theta \quad\quad \frac{\partial a_2}{\partial x_2} = \sin\theta$$

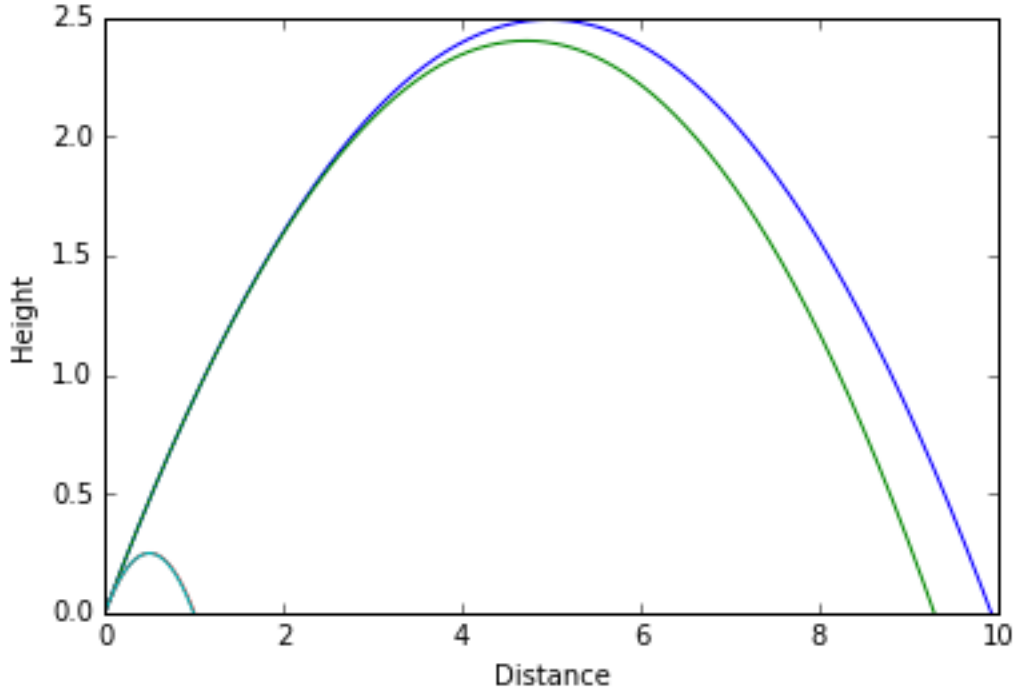
$$\boxed{\begin{aligned} \frac{\partial u_1}{\partial t^*} &= -\frac{\pi\rho C_D D^3}{8m} V^{*^2} \cos\theta \\ \frac{\partial u_2}{\partial t^*} &= -\frac{gD}{V_0^2} - \frac{\pi\rho C_D D^3}{8m} V^{*^2} \sin\theta \\ \frac{\partial u_3}{\partial t} &= 0 \end{aligned}}$$

The remainder of this problem was done numerically using Python. See attached.

This file contains the console output to the attached Python code.

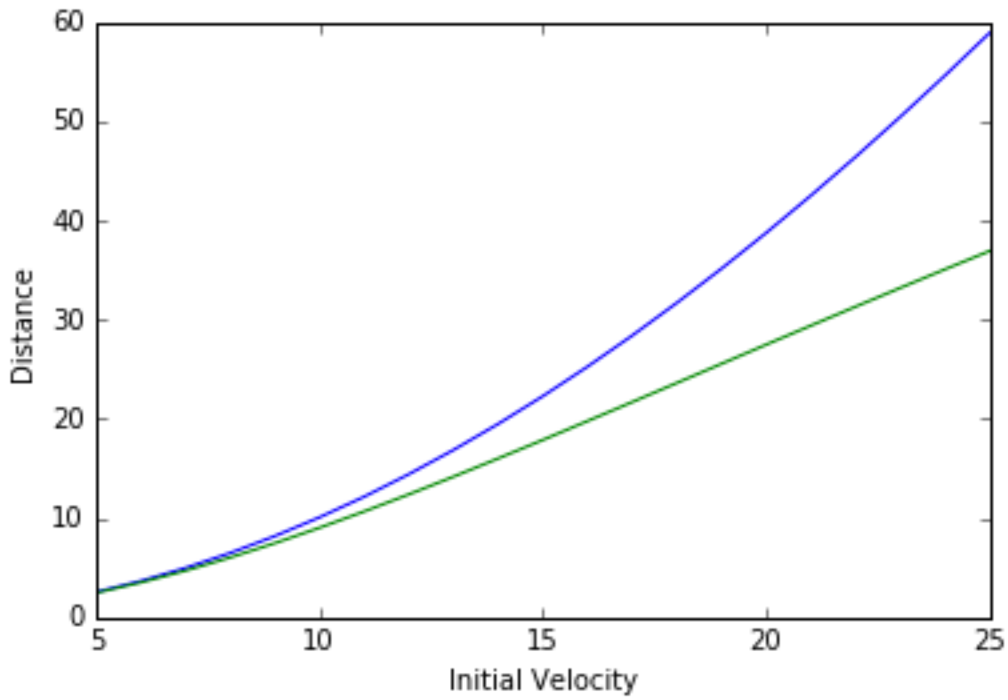
Problem 3.c)

```
runfile('/Users/andrewalferman/me_560_hw_1_10_7_16_2',
wdir='/Users/andrewalferman')
Ball 1 impacts the ground under the following conditions:
 Time: 14.113999999997617.
 Distance: 9.923354029947712.
 Speed of impact: 0.9922882297568092.
 Maximum height: 2.4899112337057785.
 Coefficient of drag: 0.25464790894703255.
----------
Ball 2 impacts the ground under the following conditions:
 Time: 13.858999999997758.
 Distance: 9.284732274870318.
 Speed of impact: 0.930263427304341.
 Maximum height: 2.4012306578667837.
 Coefficient of drag: 2.5464790894703246.
----------
Ball 3 impacts the ground under the following conditions:
 Time: 1.414999999999955.
 Distance: 0.9999822846703242.
 Speed of impact: 0.9992764142508377.
 Maximum height: 0.250248483596505.
 Coefficient of drag: 0.25464790894703255.
----------
Ball 4 impacts the ground under the following conditions:
 Time: 1.412999999999552.
 Distance: 0.9934566776038485.
 Speed of impact: 0.9927826573184741.
 Maximum height: 0.24930806363136415.
 Coefficient of drag: 2.5464790894703246.
----------
```



Problem 3.d)

```
runfile('/Users/andrewalferman/me_560_hw_1_10_10_16_1',
wdir='/Users/andrewalferman')
Coefficient of drag for P1 = 0.001: 0.2546479089470325
Coefficient of drag for P1 = 0.01: 2.5464790894703246
```



```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Oct  5 21:28:19 2016

Authors: Andrew Alferman and Nathan Schorn

This is the Python code used to obtain the answer to problem 3.c)
"""

#Importing Commands
import numpy as np
import matplotlib.pyplot as plt

#Time Variables
timescale = 0.001
runtime = 250.0
#The variables below are modified as part of the problem.
p1_1=0.001
p2_1=0.1
p1_2=0.01
p2_2=1.0

#Creating a function to plot a trajectory
def trajectory(p1,p2,timescale,runtime,case):
    #Initialize the variables and plotting vectors
    x=0.0
    y=0.0
    time = 0.0
    theta=np.radians(45.0)
    v_x=np.cos(theta)
    v_y=np.sin(theta)
    y_max = 0.0
    a=[]
    b=[]
    #Given Information
    d=0.1
    m=1.0
    rho=10.0
    #Find the coefficient of drag
    c_d=(8.0*p1*m)/(np.pi*rho*d**3)
    #Computation of the trajectory is accomplished in a single while loo
    while time<=runtime:
        #Append the plotting vectors with the current x and y coordinate
        a.append(x)
        b.append(y)
        #Move the ball based on the velocity of the previous timestep
        x+=v_x*timescale
        y+=v_y*timescale
        #Calculate the velocity and angle computed in the previous times
        v=np.sqrt(v_x**2+v_y**2)
        theta=np.arctan(v_y/v_x)
        #Accelerate the ball using the formula calculated in step 3.c)
```

```
        v_x+=-1*p1*v**2*np.cos(theta)*timescale
        v_y+=-1*(p2+(p1*v**2*np.sin(theta)))*timescale
        #Advance the time one timescale
        time+=timescale
        #Determine the maximum height and output stats of the trajectory
        if y>=y_max:
            y_max=y
        elif y <= 0:
            print("Ball {} impacts the ground under the following condit:
                  .format(case))
            print(" Time: {}.".format(time))
            print(" Distance: {}.".format(x))
            print(" Speed of impact: {}.".format(v))
            print(" Maximum height: {}.".format(y_max))
            print(" Coefficient of drag: {}.".format(c_d))
            print("----------")
            return [a,b]
            break

#Create all of the trajectory data
case1=trajectory(p1_1,p2_1,timescale,runtime,"1")
case2=trajectory(p1_2,p2_1,timescale,runtime,"2")
case3=trajectory(p1_1,p2_2,timescale,runtime,"3")
case4=trajectory(p1_2,p2_2,timescale,runtime,"4")
#Plot out the result
plt.plot(case1[0],case1[1])
plt.plot(case2[0],case2[1])
plt.plot(case3[0],case3[1])
plt.plot(case4[0],case4[1])
plt.xlabel("Distance")
plt.ylabel("Height")
plt.show()
```

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Oct  5 21:28:19 2016

authors: Andrew Alferman and Nathan Schorn

This is the Python code used to obtain the answer to problem 3.d)
"""

#Importing Commands
import numpy as np
import matplotlib.pyplot as plt

#Time Variables
timescale = 0.01
runtime = 5000.0

#Given Information
d=0.1
m=1.0
g=9.8
rho=10.0

#The variables below are modified as part of the problem.
p1_1=0.001
p1_2=0.01
p1_v=[p1_1,p1_2]

#Creating a function to plot a trajectory
def trajectory(p1,timescale,runtime,v_0,g,d):
    #Initialize the variables and plotting vectors
    x=0.0
    y=0.0
    time=0.0
    theta=np.radians(45.0)
    v_x=np.cos(theta)
    v_y=np.sin(theta)
    y_max=0.0
    v_0i=1/v_0
    a=[]
    b=[]
    #Computation of the trajectory is accomplished in a single while loo
    while time<=runtime:
        #Append the plotting vectors with the current x and y coordinate
        a.append(x)
        b.append(y)
        #Move the ball based on the velocity of the previous timestep
        x+=v_x*timescale
        y+=v_y*timescale
        #Calculate the velocity and angle computed in the previous times
        theta=np.arctan(v_y/v_x)
        v=np.sqrt(v_x**2+v_y**2)
```

```
        #Accelerate the ball using the formula calculated in step 3.c)
        v_x+=-1*p1*v**2*timescale
        v_y+=-1*(g*(v_0i**2)+p1*v**2)*timescale
        #Advance the time one timescale
        time+=timescale
        #Determine the maximum height and output stats of the trajectory
        if y>=y_max:
            y_max=y
        elif y<=0:
            return x
            break

#Find the coefficient of drag
for l in p1_v:
    c_d=(8.0*l*m)/(np.pi*rho*d**3)
    print ("Coefficient of drag for P1 = {}: {}".format(l,c_d))

#Create all of the trajectory data
x_s=[]
v_v=[]
for i in p1_v:
    for j in range(5,26):
        x_s.append(trajectory(i,timescale,runtime,j,g,d))
        v_v.append(j)
    plt.plot(v_v,x_s)
    x_s=[]
    v_v=[]


#Plot out the result
plt.xlabel("Initial Velocity")
plt.ylabel("Distance")
plt.show()
```