Lance Tan
Homework 2

(1) $p_{tot} = 14.7$ psi

(a) tunnel speed?

$$\dot{m}_1 = \dot{m}_2$$

$$\cancel{\rho} v_1 A_1 = \cancel{\rho} v_2 A_2 \iff V_1 \times 2.97 = 9.14 \times 1.49$$

$$V_1 = 4.585 ms^{-1}$$

(b) area at outlet: ?

$$P_1 = P_3 = P_{ambient}, \quad \rho_1 = \rho_3$$

$$P_1 + \frac{\rho_1 V_1^2}{2} = P_3 + \frac{\rho_3 V_3^2}{2}$$

$$V_1 = V_3 \iff A = 2.97 \text{ m}^2$$

(c) pressure reading at inlet?

= ambient pressure since p_{static_0} is also ambient pressure

= 2116.73 psf

(2)

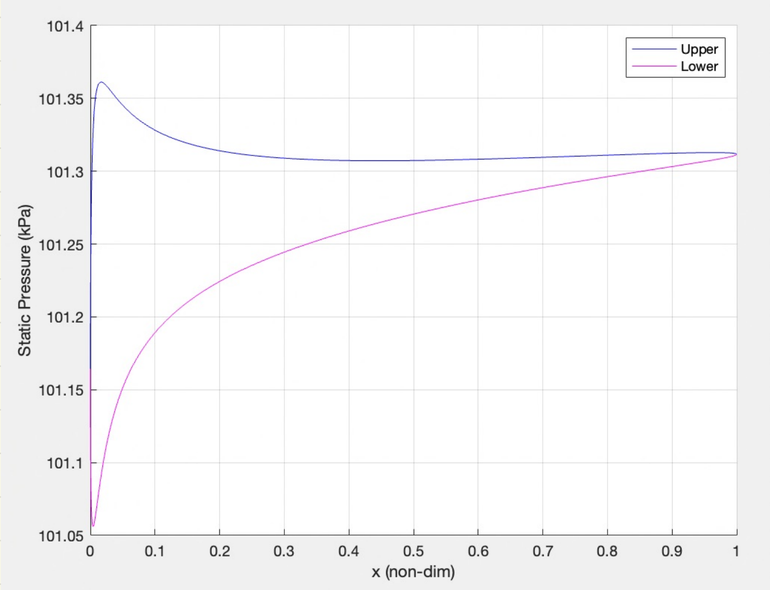
```
% data pre-processing
airfoil_data = readcell('airfoil_data.txt');
airfoil_data = airfoil_data(2:end-1,:);
x_Upper = cell2mat(airfoil_data(:,5));
x_Lower = cell2mat(airfoil_data(:,7));
v_Upper = cell2mat(airfoil_data(:,6));
v_Lower = cell2mat(airfoil_data(:,8));
v_FreeStream = 10; % m/s

% initialize variables
P_amb = 101.3*10^3; % Pa
rho_amb = 1.225; % kg/m^3

% Total air pressure
P_tot = P_amb + 1/2*rho_amb*v_FreeStream^2;

% Calculate static air pressure
P_st_u = (P_tot - 1/2*rho_amb*v_Upper.^2)/1000; % kPa
P_st_l = (P_tot - 1/2*rho_amb*v_Lower.^2)/1000; % kPa

% plot pressure distribution
hold on
plot(x_Lower,P_st_l,'b')
plot(x_Upper,P_st_u,'m')
hold off
xlabel('x (non-dim)')
ylabel('Static Pressure (kPa)')
legend('Upper','Lower')
grid on
```



There is an inverse relation observed in the graph between the upper and lower surfaces. This may be attributed to the fact that if velocity increases then dynamic pressure increases and static pressure decreases. The opposite occurs when velocity decreases.

(3) (a) assumptions:

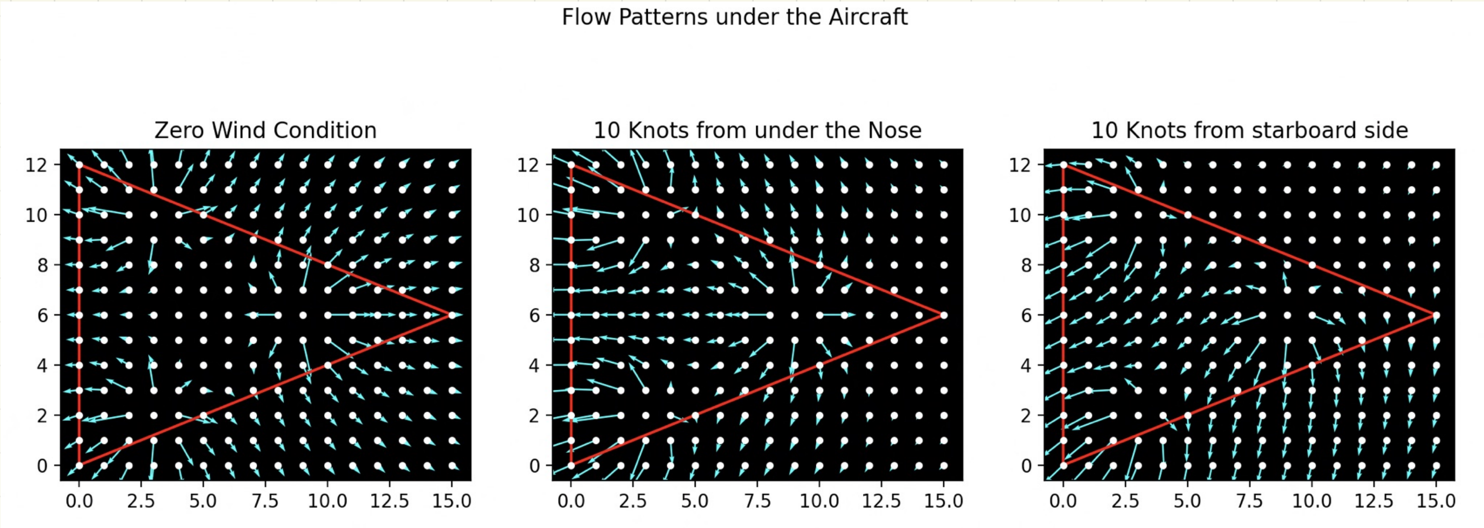
inviscid, irrotational, incompressible, steady

$$\psi = V_\infty r \sin\theta + \frac{\lambda}{2\pi}\theta \qquad V_r = \frac{\lambda}{2\pi r} \Leftrightarrow \lambda = 295.9$$

$$\psi(x,y) = \frac{295.9}{2\pi}\tan^{-1}\left(\frac{y-6}{x-9}\right) + \frac{295.9}{2\pi}\tan^{-1}\left(\frac{y-10}{x-3}\right) + \frac{295.9}{2\pi}\tan^{-1}\left(\frac{y-2}{x-3}\right)$$

$$\psi_h(x,y) = -10(y) + \frac{295.9}{2\pi}\tan^{-1}\left(\frac{y-6}{x-9}\right) + \frac{295.9}{2\pi}\tan^{-1}\left(\frac{y-10}{x-3}\right) + \frac{295.9}{2\pi}\tan^{-1}\left(\frac{y-2}{x-3}\right)$$

$$\psi_s(x,y) = -10(x) + \frac{295.9}{2\pi}\tan^{-1}\left(\frac{y-6}{x-9}\right) + \frac{295.9}{2\pi}\tan^{-1}\left(\frac{y-10}{x-3}\right) + \frac{295.9}{2\pi}\tan^{-1}\left(\frac{y-2}{x-3}\right)$$



(MATLAB CODE ATTACHED BELOW)

(b) (MATLAB CODE ATTACHED BELOW)

```python
import matplotlib.pyplot as plt
import matplotlib.tri as tri
import numpy as np
from math import *

##############
#   PART A   #
##############

# V_r magnitude
x_1 = 4
y_1 = 11
M1 = np.zeros((13, 16))

for c in range(16):
    for r in range(13):
        if r+1 == y_1 and c+1 == x_1:
            M1[r,c] = inf
            continue
        M1[r,c] = 47.1/sqrt((c+1-x_1)**2 + (r+1-y_1)**2)

x_2 = 10
y_2 = 7
M2 = np.zeros((13, 16))

for c in range(16):
    for r in range(13):
        if r+1 == y_2 and c+1 == x_2:
            M2[r,c] = inf
            continue
        M2[r,c] = 47.1/sqrt((c+1-x_2)**2 + (r+1-y_2)**2)

x_3 = 4
y_3 = 3
M3 = np.zeros((13, 16))

for c in range(16):
    for r in range(13):
        if r+1 == y_3 and c+1 == x_3:
            M3[r,c] = inf
            continue
        M3[r,c] = 47.1/sqrt((c+1-x_3)**2 + (r+1-y_3)**2)

# V_r unit vector in the x direction
M4 = np.zeros((13, 16))

for c in range(16):
    for r in range(13):
        if r+1 == y_1 and c+1 == x_1:
            M4[r,c] = inf
            continue
        M4[r,c] = (c+1-x_1)/sqrt((c+1-x_1)**2 + (y_1-r-1)**2)

M5 = np.zeros((13, 16))

for c in range(16):
    for r in range(13):
        if r+1 == y_2 and c+1 == x_2:
            M5[r,c] = inf
            continue
        M5[r,c] = (c+1-x_2)/sqrt((c+1-x_2)**2 + (y_2-r-1)**2)

M6 = np.zeros((13, 16))

for c in range(16):
    for r in range(13):
        if r+1 == y_3 and c+1 == x_3:
```

```
            M6[r,c] = inf
            continue
        M6[r,c] = (c+1-x_3)/sqrt((c+1-x_3)**2 + (y_3-r-1)**2)

# V_r unit vector in the x direction
M7 = np.zeros((13, 16))

for c in range(16):
    for r in range(13):
        if r+1 == y_1 and c+1 == x_1:
            M7[r,c] = inf
            continue
        M7[r,c] = (y_1-r-1)/sqrt((c+1-x_1)**2 + (y_1-r-1)**2)

M8 = np.zeros((13, 16))

for c in range(16):
    for r in range(13):
        if r+1 == y_2 and c+1 == x_2:
            M8[r,c] = inf
            continue
        M8[r,c] = (y_2-r-1)/sqrt((c+1-x_2)**2 + (y_2-r-1)**2)

M9 = np.zeros((13, 16))

for c in range(16):
    for r in range(13):
        if r+1 == y_3 and c+1 == x_3:
            M9[r,c] = inf
            continue
        M9[r,c] = (y_3-r-1)/sqrt((c+1-x_3)**2 + (y_3-r-1)**2)

# Unit vectors
u_1 = np.multiply(M1, M4)
v_1 = np.multiply(M1, M7)

u_2 = np.multiply(M2, M5)
v_2 = np.multiply(M2, M8)

u_3 = np.multiply(M3, M6)
v_3 = np.multiply(M3, M9)

u_4 = -16.9*np.ones((13, 16))
v_4 = 16.9*np.ones((13, 16))

U_1 = u_1 + u_2 + u_3
V_1 = v_1 + v_2 + v_3

U_h = u_1 + u_2 + u_3 + u_4
V_s = v_1 + v_2 + v_3 + v_4

# Quiver plot
fig, (ax1, ax2, ax3) = plt.subplots(1,3)
fig.suptitle('Flow Patterns under the Aircraft')
X, Y = np.meshgrid(list(range(u_1.shape[1])), list(range(u_1.shape[0])))

### PLOT 1
ax1.set_facecolor('black')
ax1.plot(X, Y, 'w.', linewidth=10)
ax1.set_aspect(1)
ax1.set_title('Zero Wind Condition')
ax1.quiver(X, Y, U_1, -V_1, color='cyan')
ship_x = [0, 0, 15]
ship_y = [0, 12, 6]
ship_tr = tri.Triangulation(ship_x, ship_y)
ax1.triplot(ship_tr, color='r')
```

```
### PLOT 2
ax2.set_facecolor('black')
ax2.plot(X, Y, 'w.', linewidth=10)
ax2.set_aspect(1)
ax2.set_title('10 Knots from under the Nose')
ax2.quiver(X, Y, U_h, -V_1, color='cyan')
ax2.triplot(ship_tr, color='r')

### PLOT 3
ax3.set_facecolor('black')
ax3.plot(X, Y, 'w.', linewidth=10)
ax3.set_aspect(1)
ax3.set_title('10 Knots from starboard side')
ax3.quiver(X, Y, U_h, -V_s, color='cyan')
ax3.triplot(ship_tr, color='r')

plt.show()

##############
#   PART B   #
##############

a_1 = sqrt(np.ma.masked_invalid(U_1).mean()**2 + np.ma.masked_invalid(-V_1).mean()**2)
a_2 = sqrt(np.ma.masked_invalid(U_h).mean()**2 + np.ma.masked_invalid(-V_1).mean()**2)
a_3 = sqrt(np.ma.masked_invalid(U_1).mean()**2 + np.ma.masked_invalid(-V_s).mean()**2)

P_amb = 2116.0
rho = 0.0765
V_avg_o = 16.87
V_avg_u = a_2
P_dyn_o = .5 * rho * V_avg_o**2
P_dyn_u = .5 * rho * V_avg_u**2
P_st_u = P_amb + P_dyn_o - P_dyn_u
area = 90
force_suction = (P_amb - P_st_u) * area
print('Suction Force = ' + str(force_suction))
```