A616 - Gas Dynamics

QUESTION 1

Consider the one-dimensional flow with heat addition (or extraction) taking place, the governing equations are:

$$\rho_1 u_1 = \rho_2 u_2$$

$$P_1 + \rho_1 u_1^2 = P_2 + \rho_2 u_2^2$$

$$h1 + \frac{u_1^2}{2} + q = h2 + \frac{u_2^2}{2}$$

$$h = CpT$$

$$P = \rho RT$$

From the definition of total temperature

$$q = C_p(T_{02} - T_{01})$$

Ratios of properties between regions 1 and 2 in terms of the Mach numbers M$_1$ and M$_2$:

$$\rho u^2 = \rho a^2 M^2 = \frac{\rho\gamma P}{\rho} M = \gamma P M^2$$

$$\rho_1 u_1^2 = \gamma P_1 M_1^2$$

$$P_2 - P_1 = \rho_1 u_1^2 - \rho_2 u_2^2$$

$$P_2 - P_1 = \gamma P_1 M_1^2 - \gamma P_2 M_2^2$$

$$\frac{P_2}{P_1} - 1 = \gamma M_1^2 - \frac{\gamma P_2}{P_1^{\square}} M_2^2$$

$$\frac{P_2}{P_1}(1 + \gamma M_2^2) = \gamma M_1^2 + 1$$

$$\frac{P_2}{P_1} = \frac{1 + \gamma M_1^2}{1 + \gamma M_2^2}$$

$$\frac{T_2}{T_1} = \frac{P_2}{P_1} . \frac{U_2}{U_1}$$

$$\frac{T_2}{T_1} = \left(\frac{1+\gamma M_1^2}{1+\gamma M_2^2}\right)^2 \left(\frac{M_2}{M_1}\right)^2$$

$$\frac{T_{02}}{T_{01}} = \frac{T_{02}}{T_2} \cdot \frac{T_1}{T_{01}} \cdot \frac{T_2}{T_1}$$

$$\frac{T_{02}}{T_{01}} = \left(\frac{\left(1+\frac{\gamma-1}{2}M_2^2\right)}{1+\frac{\gamma-1}{2}M_1^2}\right)^2 \cdot \left(\frac{1+\gamma M_1^2}{1+\gamma M_2^2}\right)^2 \cdot \left(\frac{M_2}{M_1}\right)^2$$

For convenience of calculation, we use sonic flow as a reference condition.

Let M_1 =1; the corresponding flow properties are denoted by P_1= p*, T_1= T *,ρ_1= ρ*, P_{01} =P_0* and T_{01}, =T_0*. The flow properties at any other value of M are then obtained by inserting M_1 =1 and M_2 =M

$$\frac{T_{02}}{T_{01}} = \left(\frac{M_2}{M_1}\right)^2 \left(\frac{1+\gamma M_1^2}{1+\gamma M_2^2}\right)^2 \left(\frac{1+\frac{(\gamma-1)}{2}M_2^2}{1+\frac{(\gamma-1)}{2}M_1^2}\right)$$

Then, $$\frac{T_0}{T_0^*} = \frac{2(\gamma+1)M^2[1+\frac{(\gamma-1)}{2}M^2]}{\left(1+\gamma M^2\right)^2}$$

The reference sonic conditions (the starred quantities) are constant values. These starred values, although defined as conditions that exist at Mach 1, are fundamentally different than T*, p*, and ρ*.

Let $\frac{T_0}{T_0^*} = \Phi,\ M^2 = x$

$$\Phi = \frac{2(\gamma+1)x[1+\frac{(\gamma-1)}{2}x]}{(1+\gamma x)^2}$$

$$(1+\gamma x)^2\Phi = 2(\gamma+1)x[1+\frac{(\gamma-1)}{2}x]$$

$$(1+\gamma x)^2\Phi = (\gamma+1)x[2+(\gamma-1)x]$$

$$(1+2\gamma x+\gamma^2x^2)\Phi = (\gamma+1)x[2+(\gamma-1)x]$$

$$(1+2\gamma x+\gamma^2x^2)\Phi = 2\gamma x+2x+x^2(\gamma^2-1)$$

$$x^2(\gamma^2\Phi-(\gamma^2-1)) + x(2\gamma\Phi-2\gamma-2)+\Phi = 0$$

$$a = \left(\gamma^2\Phi-(\gamma^2-1)\right)$$

$$b = (2\gamma\Phi-2\gamma-2)$$

$$c = \Phi$$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

For subsonic, $x < 1$

For supersonic, $x > 1$

$$\sqrt{b^2 - 4ac} > 0$$

$$b^2 - 4ac > 0$$

$$(2\gamma\Phi - 2\gamma - 2)^2 - 4\left(\gamma^2\Phi - (\gamma^2 - 1)\right)\Phi > 0$$

$$-4(\gamma^2 + (2\gamma) + 1)\Phi + 4\left(\gamma^2 + (2\gamma)\right) + 4 > 0$$

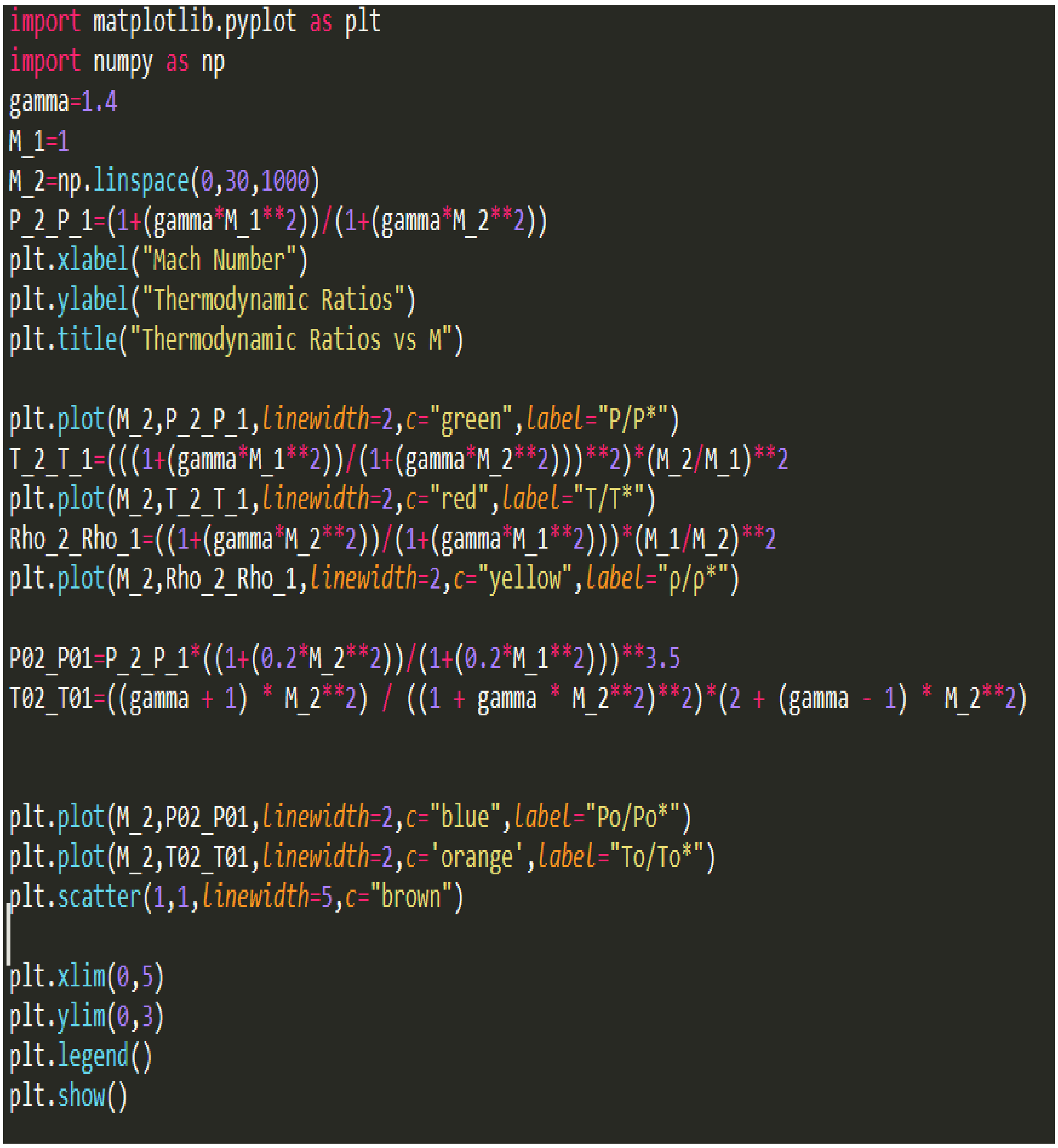
$$\gamma^2 + (2\gamma) + 1 < (\gamma^2 + (2\gamma) + 1)\,\Phi$$

$$\Phi < \frac{\gamma^2 + (2\gamma) + 1}{\gamma^2 + (2\gamma) + 1}$$

$$\Phi < 1$$

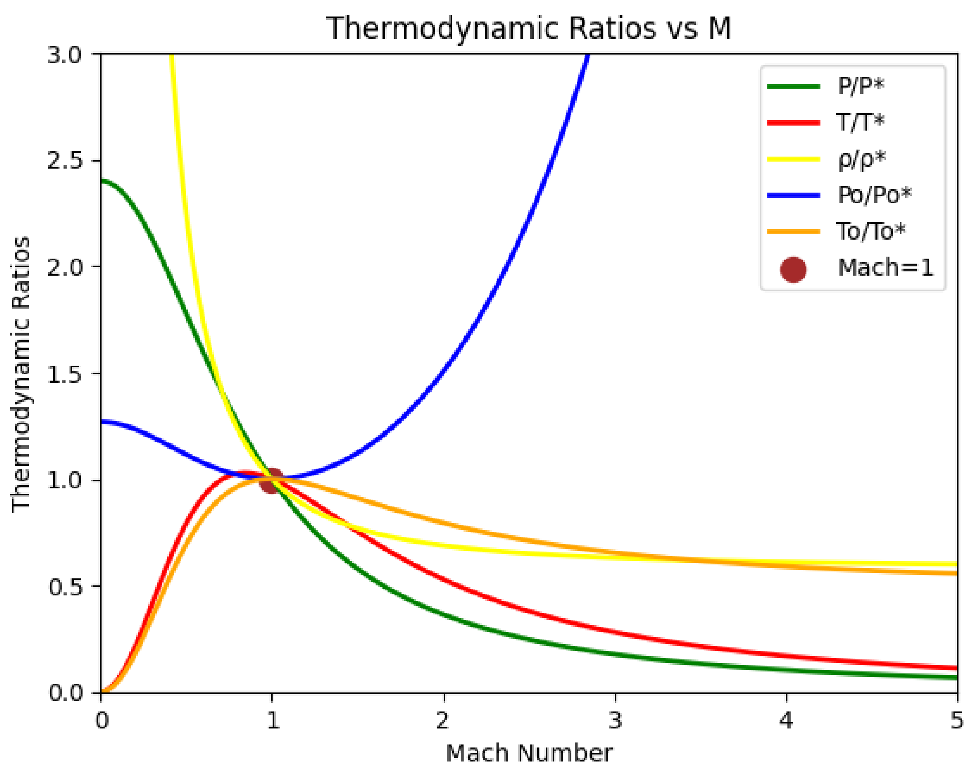
QUESTION 2:

CODE

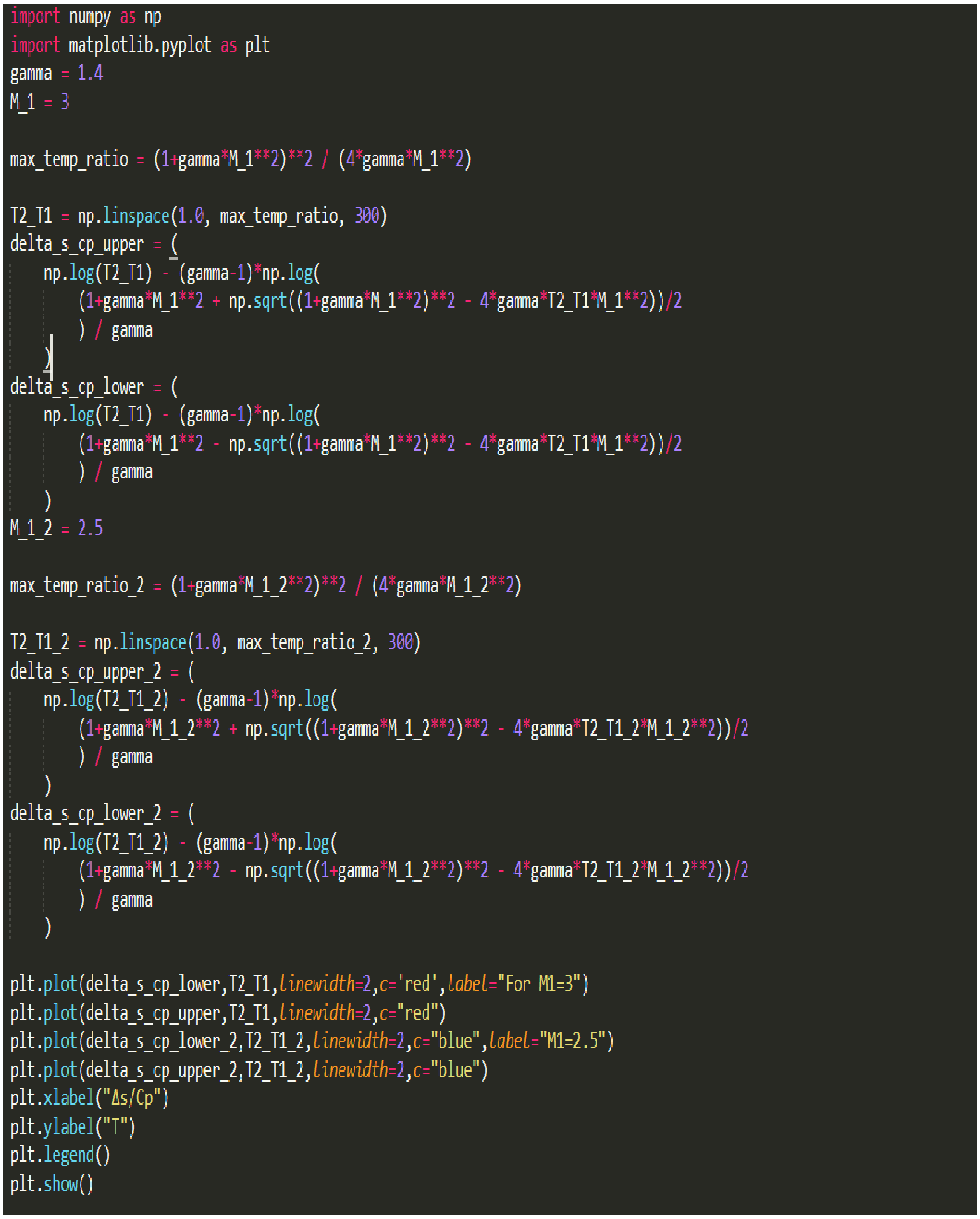


```python
import matplotlib.pyplot as plt
import numpy as np
gamma=1.4
M_1=1
M_2=np.linspace(0,30,1000)
P_2_P_1=(1+(gamma*M_1**2))/(1+(gamma*M_2**2))
plt.xlabel("Mach Number")
plt.ylabel("Thermodynamic Ratios")
plt.title("Thermodynamic Ratios vs M")

plt.plot(M_2,P_2_P_1,linewidth=2,c="green",label="P/P*")
T_2_T_1=(((1+(gamma*M_1**2))/(1+(gamma*M_2**2)))**2)*(M_2/M_1)**2
plt.plot(M_2,T_2_T_1,linewidth=2,c="red",label="T/T*")
Rho_2_Rho_1=((1+(gamma*M_2**2))/(1+(gamma*M_1**2)))*(M_1/M_2)**2
plt.plot(M_2,Rho_2_Rho_1,linewidth=2,c="yellow",label="p/p*")

P02_P01=P_2_P_1*((1+(0.2*M_2**2))/(1+(0.2*M_1**2)))**3.5
T02_T01=((gamma + 1) * M_2**2) / ((1 + gamma * M_2**2)**2)*(2 + (gamma - 1) * M_2**2)


plt.plot(M_2,P02_P01,linewidth=2,c="blue",label="Po/Po*")
plt.plot(M_2,T02_T01,linewidth=2,c='orange',label="To/To*")
plt.scatter(1,1,linewidth=5,c="brown")

plt.xlim(0,5)
plt.ylim(0,3)
plt.legend()
plt.show()
```

Plot of Thermodynamic properties vs Mach number:

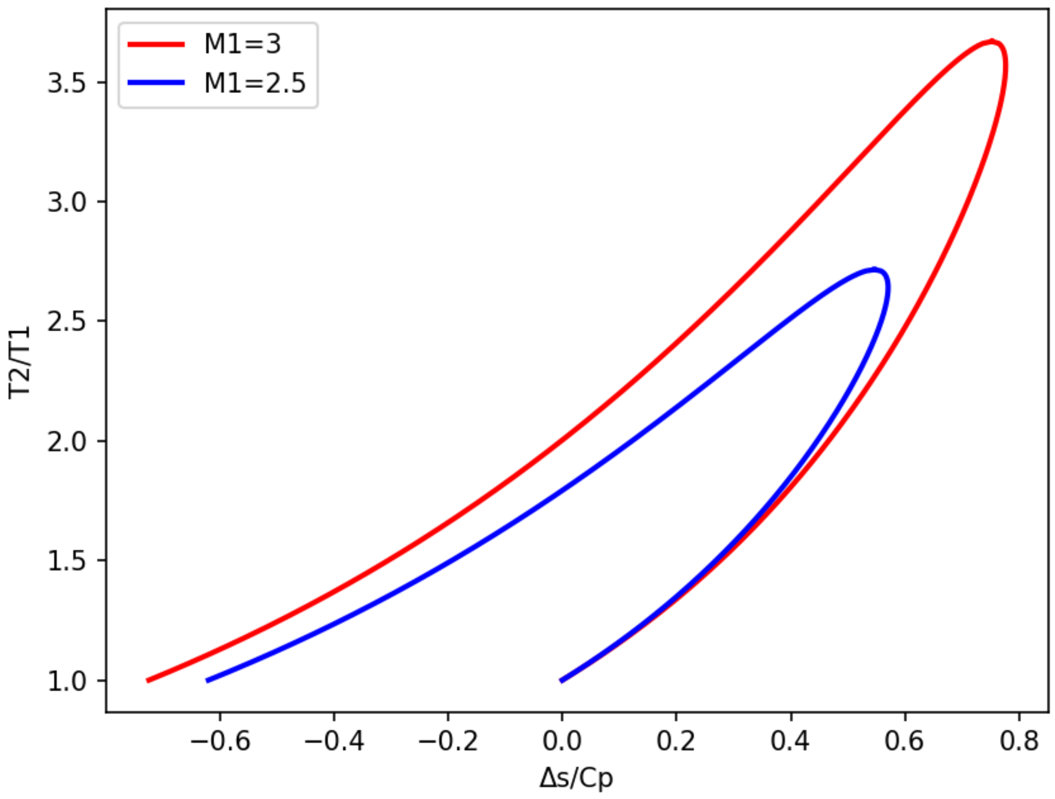


Code for T-s diagram:



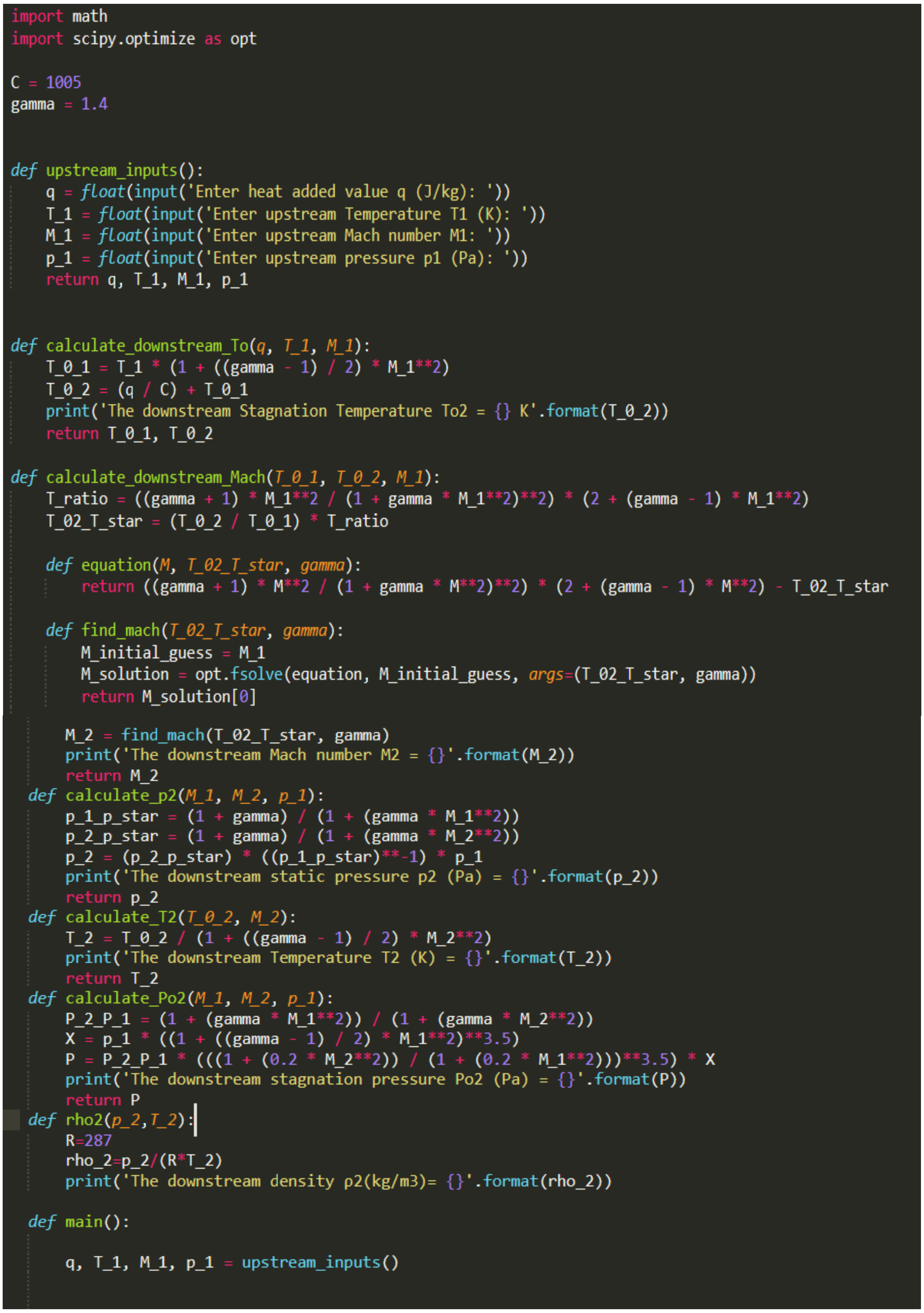
```python
import numpy as np
import matplotlib.pyplot as plt
gamma = 1.4
M_1 = 3

max_temp_ratio = (1+gamma*M_1**2)**2 / (4*gamma*M_1**2)

T2_T1 = np.linspace(1.0, max_temp_ratio, 300)
delta_s_cp_upper = (
    np.log(T2_T1) - (gamma-1)*np.log(
        (1+gamma*M_1**2 + np.sqrt((1+gamma*M_1**2)**2 - 4*gamma*T2_T1*M_1**2))/2
        ) / gamma
    )
delta_s_cp_lower = (
    np.log(T2_T1) - (gamma-1)*np.log(
        (1+gamma*M_1**2 - np.sqrt((1+gamma*M_1**2)**2 - 4*gamma*T2_T1*M_1**2))/2
        ) / gamma
    )
M_1_2 = 2.5

max_temp_ratio_2 = (1+gamma*M_1_2**2)**2 / (4*gamma*M_1_2**2)

T2_T1_2 = np.linspace(1.0, max_temp_ratio_2, 300)
delta_s_cp_upper_2 = (
    np.log(T2_T1_2) - (gamma-1)*np.log(
        (1+gamma*M_1_2**2 + np.sqrt((1+gamma*M_1_2**2)**2 - 4*gamma*T2_T1_2*M_1_2**2))/2
        ) / gamma
    )
delta_s_cp_lower_2 = (
    np.log(T2_T1_2) - (gamma-1)*np.log(
        (1+gamma*M_1_2**2 - np.sqrt((1+gamma*M_1_2**2)**2 - 4*gamma*T2_T1_2*M_1_2**2))/2
        ) / gamma
    )

plt.plot(delta_s_cp_lower,T2_T1,linewidth=2,c='red',label="For M1=3")
plt.plot(delta_s_cp_upper,T2_T1,linewidth=2,c="red")
plt.plot(delta_s_cp_lower_2,T2_T1_2,linewidth=2,c="blue",label="M1=2.5")
plt.plot(delta_s_cp_upper_2,T2_T1_2,linewidth=2,c="blue")
plt.xlabel("Δs/Cp")
plt.ylabel("T")
plt.legend()
plt.show()
```

Plot of T-S diagram

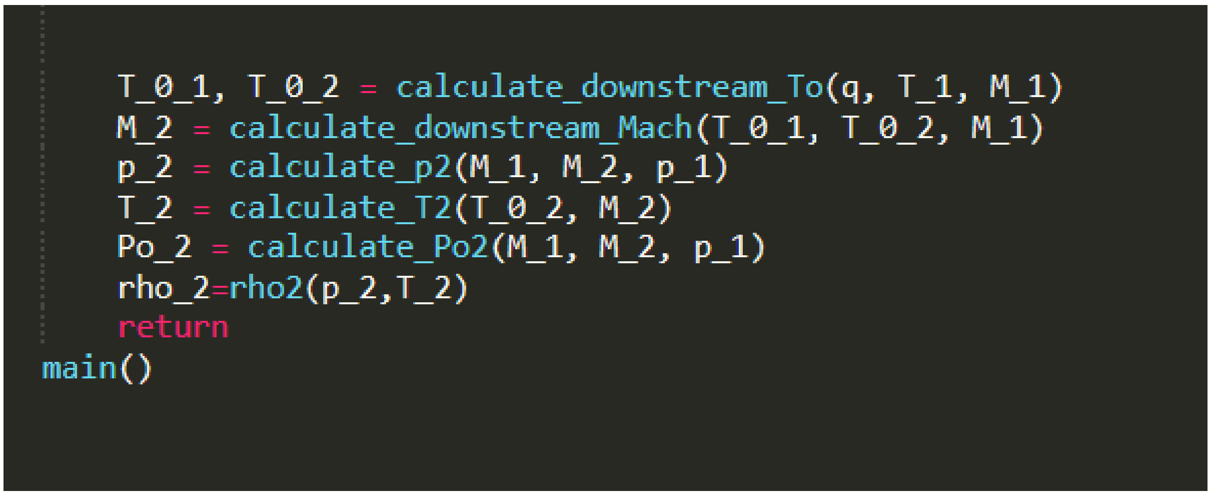


QUESTION 3

CODE:



```python
import math
import scipy.optimize as opt

C = 1005
gamma = 1.4


def upstream_inputs():
    q = float(input('Enter heat added value q (J/kg): '))
    T_1 = float(input('Enter upstream Temperature T1 (K): '))
    M_1 = float(input('Enter upstream Mach number M1: '))
    p_1 = float(input('Enter upstream pressure p1 (Pa): '))
    return q, T_1, M_1, p_1


def calculate_downstream_To(q, T_1, M_1):
    T_0_1 = T_1 * (1 + ((gamma - 1) / 2) * M_1**2)
    T_0_2 = (q / C) + T_0_1
    print('The downstream Stagnation Temperature To2 = {} K'.format(T_0_2))
    return T_0_1, T_0_2

def calculate_downstream_Mach(T_0_1, T_0_2, M_1):
    T_ratio = ((gamma + 1) * M_1**2 / (1 + gamma * M_1**2)**2) * (2 + (gamma - 1) * M_1**2)
    T_02_T_star = (T_0_2 / T_0_1) * T_ratio

    def equation(M, T_02_T_star, gamma):
        return ((gamma + 1) * M**2 / (1 + gamma * M**2)**2) * (2 + (gamma - 1) * M**2) - T_02_T_star

    def find_mach(T_02_T_star, gamma):
        M_initial_guess = M_1
        M_solution = opt.fsolve(equation, M_initial_guess, args=(T_02_T_star, gamma))
        return M_solution[0]

    M_2 = find_mach(T_02_T_star, gamma)
    print('The downstream Mach number M2 = {}'.format(M_2))
    return M_2
def calculate_p2(M_1, M_2, p_1):
    p_1_p_star = (1 + gamma) / (1 + (gamma * M_1**2))
    p_2_p_star = (1 + gamma) / (1 + (gamma * M_2**2))
    p_2 = (p_2_p_star) * ((p_1_p_star)**-1) * p_1
    print('The downstream static pressure p2 (Pa) = {}'.format(p_2))
    return p_2
def calculate_T2(T_0_2, M_2):
    T_2 = T_0_2 / (1 + ((gamma - 1) / 2) * M_2**2)
    print('The downstream Temperature T2 (K) = {}'.format(T_2))
    return T_2
def calculate_Po2(M_1, M_2, p_1):
    P_2_P_1 = (1 + (gamma * M_1**2)) / (1 + (gamma * M_2**2))
    X = p_1 * ((1 + ((gamma - 1) / 2) * M_1**2)**3.5)
    P = P_2_P_1 * (((1 + (0.2 * M_2**2)) / (1 + (0.2 * M_1**2)))**3.5) * X
    print('The downstream stagnation pressure Po2 (Pa) = {}'.format(P))
    return P
def rho2(p_2,T_2):
    R=287
    rho_2=p_2/(R*T_2)
    print('The downstream density p2(kg/m3)= {}'.format(rho_2))

def main():

    q, T_1, M_1, p_1 = upstream_inputs()

```



```python
    T_0_1, T_0_2 = calculate_downstream_To(q, T_1, M_1)
    M_2 = calculate_downstream_Mach(T_0_1, T_0_2, M_1)
    p_2 = calculate_p2(M_1, M_2, p_1)
    T_2 = calculate_T2(T_0_2, M_2)
    Po_2 = calculate_Po2(M_1, M_2, p_1)
    rho_2=rho2(p_2,T_2)
    return
main()
```

Example output:

For example 3.13 from Modern Compressible Flow by John D Anderson

```
Enter heat added value q (J/kg): 1000000
Enter upstream Temperature T1 (K): 273
Enter upstream Mach number M1: 0.2
Enter upstream pressure p1 (Pa): 101325
The downstream Stagnation Temperature To2 = 1270.2088756218907 K
The downstream Mach number M2 = 0.5840066194598165
The downstream static pressure p2 (Pa) = 72419.61446980316
The downstream Temperature T2 (K) = 1189.0972842293988
The downstream stagnation pressure Po2 (Pa) = 91234.34404050978
The downstream density p2(kg/m3)= 0.21220564104328493
```

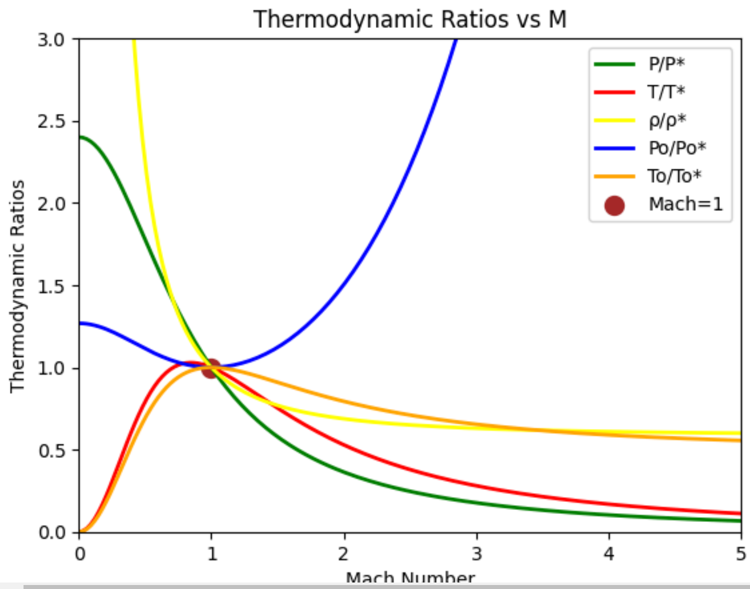
For example 3.14 from Modern Compressible Flow by John D Anderson

```
Enter heat added value q (J/kg): 300000
Enter upstream Temperature T1 (K): 300
Enter upstream Mach number M1: 3
Enter upstream pressure p1 (Pa): 101325
The downstream Stagnation Temperature To2 = 1138.5074626865671 K
The downstream Mach number M2 = 1.591186123206545
The downstream static pressure p2 (Pa) = 303219.89838803234
The downstream Temperature T2 (K) = 755.793028215505
The downstream stagnation pressure Po2 (Pa) = 1272105.7000124604
The downstream density p2(kg/m3)= 1.3978897475256793
```

Code for the variation of various thermodynamic ratios with Mach number in rayleigh flow.

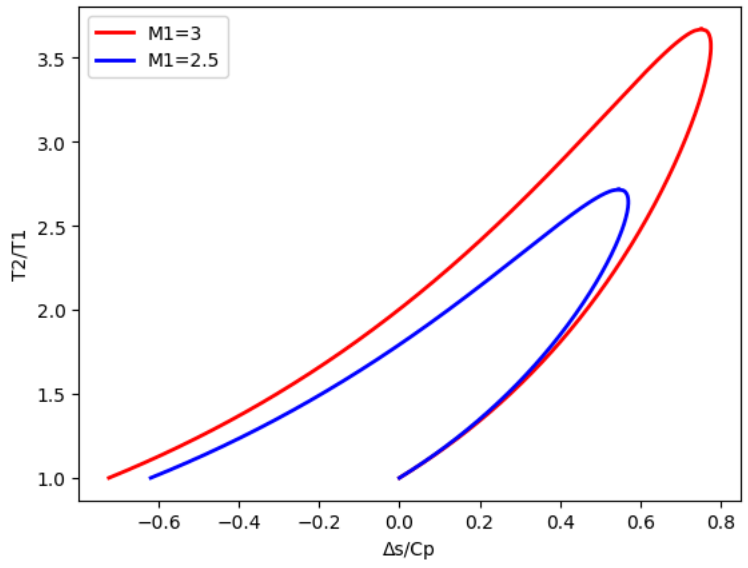
```
import matplotlib.pyplot as plt
import numpy as np
gamma=1.4
M_1=1
M_2=np.linspace(0,30,1000)
P_2_P_1=(1+(gamma*M_1**2))/(1+(gamma*M_2**2))
plt.xlabel("Mach Number")
plt.ylabel("Thermodynamic Ratios")
plt.title("Thermodynamic Ratios vs M")

plt.plot(M_2,P_2_P_1,linewidth=2,c="green",label="P/P*")
T_2_T_1=(((1+(gamma*M_1**2))/(1+(gamma*M_2**2)))**2)*(M_2/M_1)**2
plt.plot(M_2,T_2_T_1,linewidth=2,c="red",label="T/T*")
Rho_2_Rho_1=((1+(gamma*M_2**2))/(1+(gamma*M_1**2)))*(M_1/M_2)**2
plt.plot(M_2,Rho_2_Rho_1,linewidth=2,c="yellow",label="ρ/ρ*")

P02_P01=P_2_P_1*((1+(0.2*M_2**2))/(1+(0.2*M_1**2)))**3.5
T02_T01=((gamma + 1) * M_2**2) / ((1 + gamma * M_2**2)**2)*(2 + (gamma - 1) * M_2**2)


plt.plot(M_2,P02_P01,linewidth=2,c="blue",label="Po/Po*")
plt.plot(M_2,T02_T01,linewidth=2,c='orange',label="To/To*")
plt.scatter(1,1,linewidth=5,c="brown",label='Mach=1')

plt.xlim(0,5)
plt.ylim(0,3)
plt.legend()
plt.show()
```

```
<ipython-input-2-1cef68c448e8>:14: RuntimeWarning: divide by zero encountered in divide
  Rho_2_Rho_1=((1+(gamma*M_2**2))/(1+(gamma*M_1**2)))*(M_1/M_2)**2
```



Code for T-s variation of rayleigh flow for 2 different Mach numbers

```
import numpy as np
import matplotlib.pyplot as plt
gamma = 1.4
M_1 = 3

max_temp_ratio = (1+gamma*M_1**2)**2 / (4*gamma*M_1**2)

T2_T1 = np.linspace(1.0, max_temp_ratio, 300)
delta_s_cp_upper = (
    np.log(T2_T1) - (gamma-1)*np.log(
        (1+gamma*M_1**2 + np.sqrt((1+gamma*M_1**2)**2 - 4*gamma*T2_T1*M_1**2))/2
        ) / gamma
    )
delta_s_cp_lower = (
    np.log(T2_T1) - (gamma-1)*np.log(
        (1+gamma*M_1**2 - np.sqrt((1+gamma*M_1**2)**2 - 4*gamma*T2_T1*M_1**2))/2
        ) / gamma
    )
M_1_2 = 2.5

max_temp_ratio_2 = (1+gamma*M_1_2**2)**2 / (4*gamma*M_1_2**2)

T2_T1_2 = np.linspace(1.0, max_temp_ratio_2, 300)
delta_s_cp_upper_2 = (
    np.log(T2_T1_2) - (gamma-1)*np.log(
        (1+gamma*M_1_2**2 + np.sqrt((1+gamma*M_1_2**2)**2 - 4*gamma*T2_T1_2*M_1_2**2))/2
        ) / gamma
    )
delta_s_cp_lower_2 = (
    np.log(T2_T1_2) - (gamma-1)*np.log(
        (1+gamma*M_1_2**2 - np.sqrt((1+gamma*M_1_2**2)**2 - 4*gamma*T2_T1_2*M_1_2**2))/2
        ) / gamma
    )

plt.plot(delta_s_cp_lower,T2_T1,linewidth=2,c='red',label="M1=3")
plt.plot(delta_s_cp_upper,T2_T1,linewidth=2,c="red")
plt.plot(delta_s_cp_lower_2,T2_T1_2,linewidth=2,c="blue",label="M1=2.5")
plt.plot(delta_s_cp_upper_2,T2_T1_2,linewidth=2,c="blue")
plt.xlabel("Δs/Cp")
plt.ylabel("T2/T1")
plt.legend()
plt.show()
```



Code and solution for Andersons example 3.13

```
import math
import scipy.optimize as opt

C = 1005
gamma = 1.4


def upstream_inputs():
    q = float(input('Enter heat added value q (J/kg): '))
    T_1 = float(input('Enter upstream Temperature T1 (K): '))
    M_1 = float(input('Enter upstream Mach number M1: '))
    p_1 = float(input('Enter upstream pressure p1 (Pa): '))
    return q, T_1, M_1, p_1


def calculate_downstream_To(q, T_1, M_1):
    T_0_1 = T_1 * (1 + ((gamma - 1) / 2) * M_1**2)
    T_0_2 = (q / C) + T_0_1
    print('The downstream Stagnation Temperature To2 = {} K'.format(T_0_2))
    return T_0_1, T_0_2

def calculate_downstream_Mach(T_0_1, T_0_2, M_1):
    T_ratio = ((gamma + 1) * M_1**2 / (1 + gamma * M_1**2)**2) * (2 + (gamma - 1) * M_1**2)
    T_02_T_star = (T_0_2 / T_0_1) * T_ratio

    def equation(M, T_02_T_star, gamma):
        return ((gamma + 1) * M**2 / (1 + gamma * M**2)**2) * (2 + (gamma - 1) * M**2) - T_02_T_star
    def find_mach(T_02_T_star, gamma):
        M_initial_guess = M_1
        M_solution = opt.fsolve(equation, M_initial_guess, args=(T_02_T_star, gamma))
        return M_solution[0]

    M_2 = find_mach(T_02_T_star, gamma)
    print('The downstream Mach number M2 = {}'.format(M_2))
    return M_2
def calculate_p2(M_1, M_2, p_1):
    p_1_p_star = (1 + gamma) / (1 + (gamma * M_1**2))
    p_2_p_star = (1 + gamma) / (1 + (gamma * M_2**2))
    p_2 = (p_2_p_star) * ((p_1_p_star)**-1) * p_1
    print('The downstream static pressure p2 (Pa) = {}'.format(p_2))
    return p_2
def calculate_T2(T_0_2, M_2):
    T_2 = T_0_2 / (1 + ((gamma - 1) / 2) * M_2**2)
    print('The downstream Temperature T2 (K) = {}'.format(T_2))
    return T_2
def calculate_Po2(M_1, M_2, p_1):
    P_2_P_1 = (1 + (gamma * M_1**2)) / (1 + (gamma * M_2**2))
    X = p_1 * ((1 + ((gamma - 1) / 2) * M_1**2)**3.5)
    P = P_2_P_1 * (((1 + (0.2 * M_2**2)) / (1 + (0.2 * M_1**2)))**3.5) * X
    print('The downstream stagnation pressure Po2 (Pa) = {}'.format(P))
    return P
def rho2(p_2,T_2):
    R=287
    rho_2=p_2/(R*T_2)
    print('The downstream density ρ2(kg/m3)= {}'.format(rho_2))

def main():

    q, T_1, M_1, p_1 = upstream_inputs()


    T_0_1, T_0_2 = calculate_downstream_To(q, T_1, M_1)
    M_2 = calculate_downstream_Mach(T_0_1, T_0_2, M_1)
    p_2 = calculate_p2(M_1, M_2, p_1)
    T_2 = calculate_T2(T_0_2, M_2)
    Po_2 = calculate_Po2(M_1, M_2, p_1)
    rho_2=rho2(p_2,T_2)
    return
main()
```

```
Enter heat added value q (J/kg): 300000
Enter upstream Temperature T1 (K): 300
Enter upstream Mach number M1: 3
Enter upstream pressure p1 (Pa): 101325
The downstream Stagnation Temperature To2 = 1138.5074626865671 K
The downstream Mach number M2 = 1.591186123206545
The downstream static pressure p2 (Pa) = 303219.89838803234
The downstream Temperature T2 (K) = 755.793028215505
The downstream stagnation pressure Po2 (Pa) = 1272105.7000124604
The downstream density ρ2(kg/m3)= 1.3978897475256793
```

QUESTION 1

Consider example 8.3 of Yahya's textbook that was discussed in class; it had a specified supersonic inlet, sonic outlet and pre-shock Mach number, and you were asked to find the lengths of the duct upstream and downstream of the shock. Now consider the related, and more. Practical, problem, where you are given the length of the duct and the inlet (supersonic) Mach number, and told that the outlet is sonic. Write a code to find the location of the normal shock in the duct, if it exists. Show that it reproduces the solution of the example problem. Also exercise your code by making up some other problems. Do you find situations where your code fails to produce any answer? Why?

Solution

1. We use the following equation to evaluate the length of duct between two locations having Mach No. M_1 and M_2 in Fanno flow. In this problem, M_1 and M_2 refers to the inlet and pre-shock Mach numbers respectively. Hence, the length L calculated from the relation given below is actually L_1 (i.e. the distance of Normal Shock from duct inlet): -

$$\frac{4f}{D}\mathrm{L} = \frac{1}{\gamma}\left(\frac{1}{{M_1}^2} - \frac{1}{{M_2}^2}\right) + \frac{\gamma + 1}{2\gamma} ln \frac{{M_1}^2}{{M_2}^2} \frac{\left(1 + \frac{\gamma - 1}{2}{M_2}^2\right)}{1 + \frac{\gamma - 1}{2}{M_1}^2}$$

2. Similarly, the above relation is used to calculate L_2 (i.e. the distance of normal shock from duct exit) by substituting M_3 (i.e. the post-shock Mach number) for M_1 and duct exit Mach Number (M_e) for M_2. Further, following Normal Shock relation has been used to relate M_2 and M_3: -

$$M_3 = \sqrt{\frac{2 + (\gamma - 1){M_2}^2}{2\gamma {M_2}^2 - (\gamma - 1)}}$$

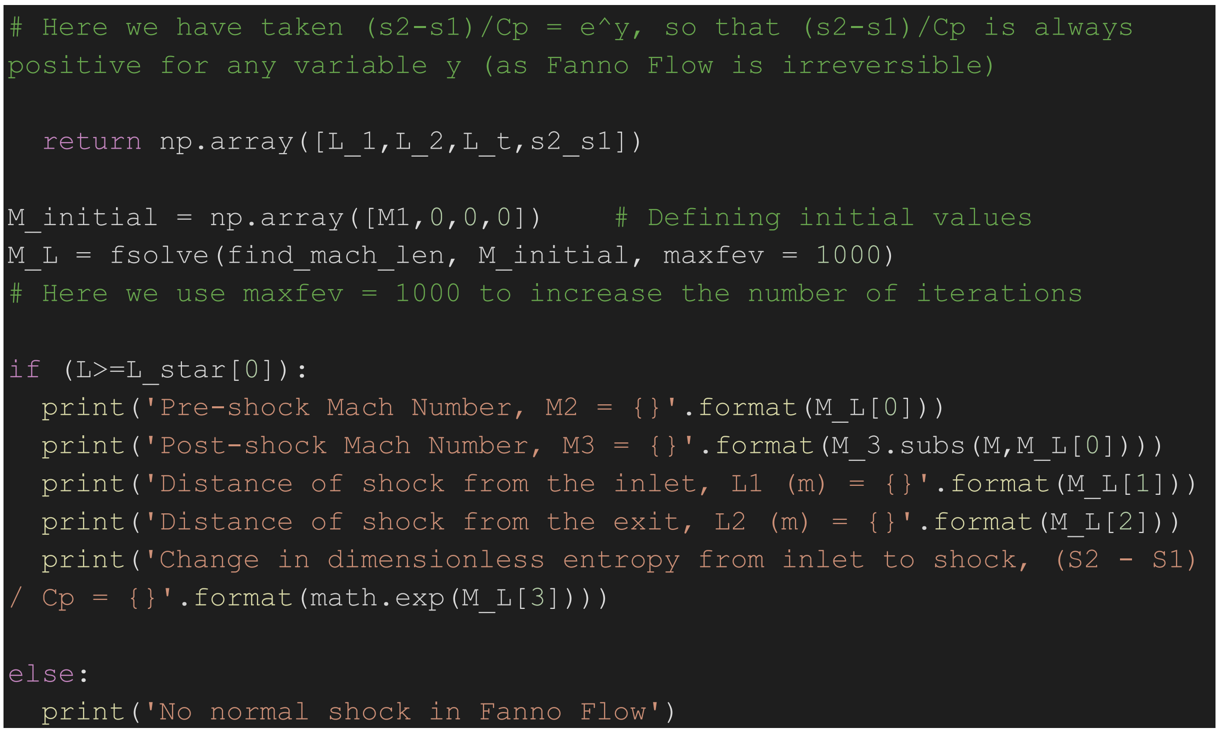
M_2 is the mach no. just upstream of the normal shock
M_3 is the mach no. just downstream of the normal shock

3. The code for given problem and output for Example 8.3 of Yahya's textbook is as given below: -

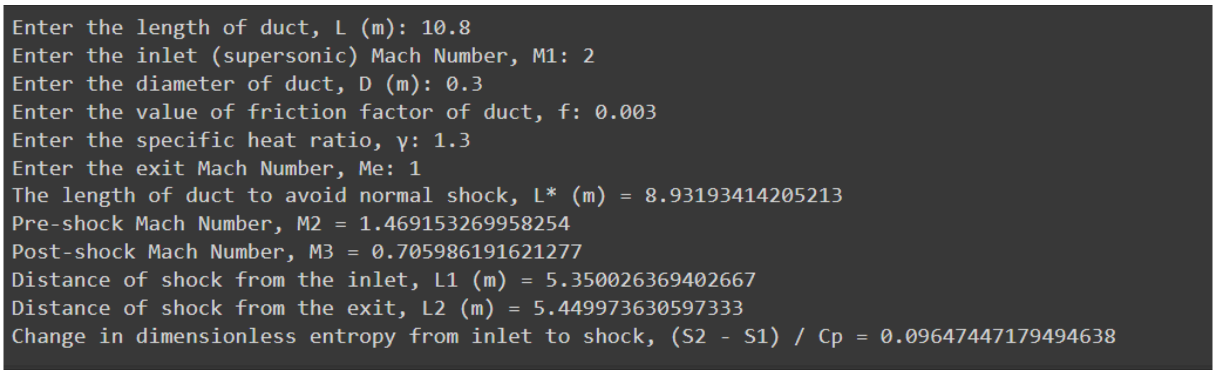
CODE

```
import numpy as np
import math
from sympy import symbols, solve
import sympy
from scipy.optimize import fsolve

L = float(input('Enter the length of duct, L (m): '))
M1 = float(input('Enter the inlet (supersonic) Mach Number, M1: '))
```

```
D = float(input('Enter the diameter of duct, D (m): '))
f = float(input('Enter the value of friction factor of duct, f: '))
γ = float(input('Enter the specific heat ratio, γ: '))
Me = float(input('Enter the exit Mach Number, Me: '))

L_star = symbols ('L*')
M = symbols ('M')
M_3 = symbols ('M_3')

expr = L_star - (D/(4*f))*((1-M**2)/(γ*M**2) +
(((γ+1)/(2*γ))*sympy.log(((γ+1)*M**2)/(2+((γ-1)*M**2)))))
# Using the relation for fanno flow relating 4fL*/D and Mach No.

expr1 = expr.subs(M,M1)
L_star = solve(expr1)

print('The length of duct to avoid normal shock, L* (m) =
{}'.format(L_star[0]))
# As any length greater than L* will cause a normal shock in the duct
(for supersonic inlet)

M_3 = sympy.sqrt((1+((γ-1)/2)*M**2)/((γ*M**2)-((γ-1)/2)))   # Normal
shock relation between upstream and downstream Mach No.

def find_mach_len(M_L):

  M2 = M_L[0]   # M2 is the Mach No. just upstream of the Normal Shock
  L1 = M_L[1]   # L1 is the length from inlet upto Normal Shock
  L2 = M_L[2]   # L2 is the length from Normal Shock upto exit
  y = M_L[3]

  M3 = math.sqrt((1+((γ-1)/2)*M2**2)/((γ*M2**2)-((γ-1)/2)))
  # Normal shock relation between M2 and M3

  L_1 = L1 - (D/(4*f))*((M2**2-M1**2)/(γ*(M1**2)*(M2**2)) +
(((γ+1)/(2*γ))*math.log(((M1**2)/(M2**2))*((1+((γ-1)/2)*M2**2)/(1+((γ-
1)/2)*M1**2)))))
# Using the fanno flow relation for M1 upstream and M2 downstream

  L_2 = L2 - (D/(4*f))*((Me**2-M3**2)/(γ*(M3**2)*(Me**2)) +
(((γ+1)/(2*γ))*math.log(((M3**2)/(Me**2))*((1+((γ-1)/2)*Me**2)/(1+((γ-
1)/2)*M3**2)))))
# Using the fanno flow relation for M3 upstream and Me downstream

  L_t = L - L1 - L2   # As Total Length of Duct, L = L1 + L2

  s2_s1 = math.exp(y) - ((γ-1)/γ)*math.log((M2/M1)*(((1+((γ-
1)/2)*M1**2)/(1+((γ-1)/2)*M2**2))**((γ+1)/(2*(γ-1)))))
```



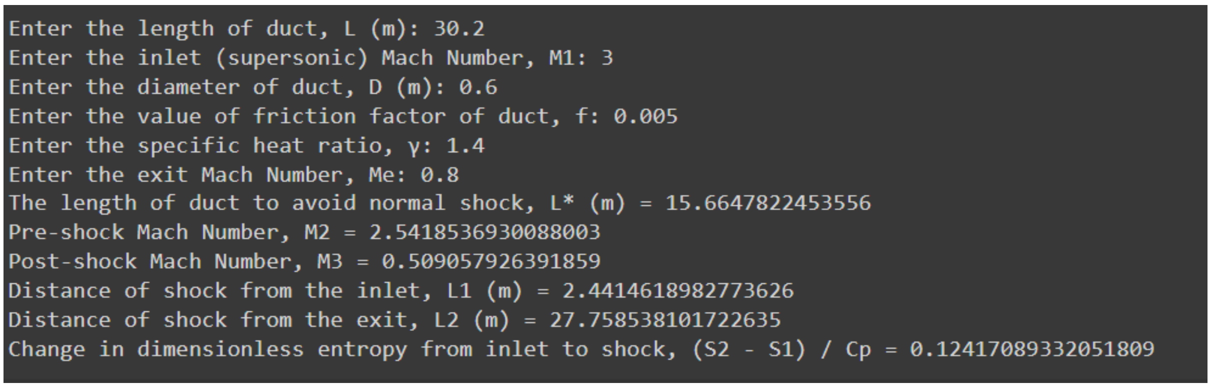
```
# Here we have taken (s2-s1)/Cp = e^y, so that (s2-s1)/Cp is always
positive for any variable y (as Fanno Flow is irreversible)

  return np.array([L_1,L_2,L_t,s2_s1])

M_initial = np.array([M1,0,0,0])     # Defining initial values
M_L = fsolve(find_mach_len, M_initial, maxfev = 1000)
# Here we use maxfev = 1000 to increase the number of iterations

if (L>=L_star[0]):
  print('Pre-shock Mach Number, M2 = {}'.format(M_L[0]))
  print('Post-shock Mach Number, M3 = {}'.format(M_3.subs(M,M_L[0])))
  print('Distance of shock from the inlet, L1 (m) = {}'.format(M_L[1]))
  print('Distance of shock from the exit, L2 (m) = {}'.format(M_L[2]))
  print('Change in dimensionless entropy from inlet to shock, (S2 - S1)
/ Cp = {}'.format(math.exp(M_L[3])))

else:
  print('No normal shock in Fanno Flow')
```



```
Enter the length of duct, L (m): 10.8
Enter the inlet (supersonic) Mach Number, M1: 2
Enter the diameter of duct, D (m): 0.3
Enter the value of friction factor of duct, f: 0.003
Enter the specific heat ratio, γ: 1.3
Enter the exit Mach Number, Me: 1
The length of duct to avoid normal shock, L* (m) = 8.93193414205213
Pre-shock Mach Number, M2 = 1.469153269958254
Post-shock Mach Number, M3 = 0.705986191621277
Distance of shock from the inlet, L1 (m) = 5.350026369402667
Distance of shock from the exit, L2 (m) = 5.449973630597333
Change in dimensionless entropy from inlet to shock, (S2 - S1) / Cp = 0.09647447179494638
```

4. Thus, there exists a normal shock for the given duct specifications (L, D, f). Its location is found to be 5.35 m from the duct entry and the pre-shock Mach Number is found to be 1.46 (against the given value of 1.5 in textbook, which is within an error of 5%).

5. To verify the code, another problem is solved from Yahya's Textbook (Unsolved Problem 8.9). The final output is in agreement with the answer given in the textbook.

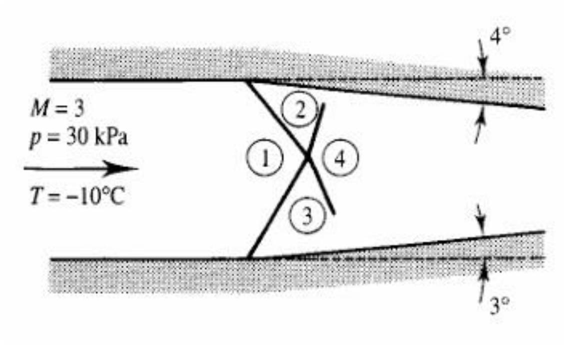


```
Enter the length of duct, L (m): 30.2
Enter the inlet (supersonic) Mach Number, M1: 3
Enter the diameter of duct, D (m): 0.6
Enter the value of friction factor of duct, f: 0.005
Enter the specific heat ratio, γ: 1.4
Enter the exit Mach Number, Me: 0.8
The length of duct to avoid normal shock, L* (m) = 15.6647822453556
Pre-shock Mach Number, M2 = 2.5418536930088003
Post-shock Mach Number, M3 = 0.509057926391859
Distance of shock from the inlet, L1 (m) = 2.4414618982773626
Distance of shock from the exit, L2 (m) = 27.758538101722635
Change in dimensionless entropy from inlet to shock, (S2 - S1) / Cp = 0.12417089332051809
```

QUESTION 2

Write a code (and run it) to solve Example 6.5 of Oosthuizen's textbook (which has a mistake towards the end of the solution). Find and solve similar examples from other textbooks/references to validate your code.

Solution



1. In the given problem, we firstly take the inputs for inlet Mach number, pressure, turning angles of the channel (both upper and lower) and the specific heat ratio.

2. Now the oblique shock angles corresponding to the initial upper and lower oblique shocks are found using Θ-β-M relation: -

$$tan\Theta = 2cot\beta\left(\frac{M^2\ (Sin\beta)^2 - 1}{(\gamma + cos2\beta)M^2 + 2}\right)$$

3. These oblique shock angles are used to find M_2 and M_3. Further, another function is defined in python to calculate the slipline angle. For this, an expression is given equating the static pressures above and below the slipline (i.e. (P4/P2) * (P2/P1) = (P5/P3) * (P3/P1)), where P4 and P5 are the static pressures above and below slipline, and ratio of static pressures across any oblique shock is found using the normal shock relations.

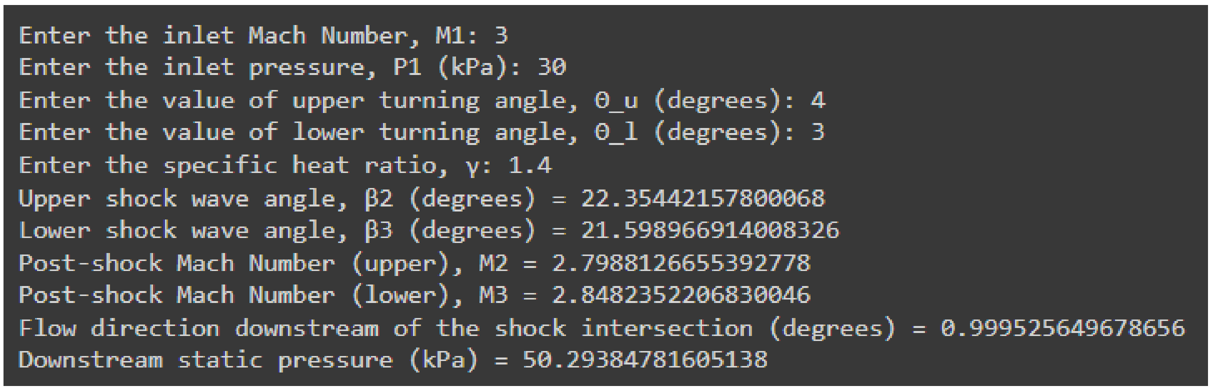
4. The code for given problem and output for Example 6.5 of Oosthuizen's textbook is as given below: -

CODE

```
import numpy as np
import math
from scipy.optimize import fsolve

M1 = float(input('Enter the inlet Mach Number, M1: '))
P1 = float(input('Enter the inlet pressure, P1 (kPa): '))
Θ_u = float(input('Enter the value of upper turning angle, Θ_u
(degrees): '))
Θ_l = float(input('Enter the value of lower turning angle, Θ_l
(degrees): '))
γ = float(input('Enter the specific heat ratio, γ: '))
```

```
Θ_u = Θ_u*math.pi/180   # Converting Θ into radians
Θ_l = Θ_l*math.pi/180
Θ = [Θ_u, Θ_l]

def find_β(β_u_l):      # Defining a function to obtain oblique shock
angles at upper and lower surface

  β2 = β_u_l[0]
  β3 = β_u_l[1]

  x = math.tan(Θ_u) - (2/(math.tan(β2))*((M1**2)*((math.sin(β2))**2) -
1))/(2 + (M1**2)*(γ + (math.cos(2*β2))))     # Using Θ-β-M relation
  y = math.tan(Θ_l) - (2/(math.tan(β3))*((M1**2)*((math.sin(β3))**2) -
1))/(2 + (M1**2)*(γ + (math.cos(2*β3))))

  return np.array([x,y])

βi = np.array([Θ_u, Θ_l])     # Defining initial values
β_u_l = fsolve(find_β, βi)

print('Upper shock wave angle, β2 (degrees) =
{}'.format((180*β_u_l[0])/math.pi))
print('Lower shock wave angle, β3 (degrees) =
{}'.format((180*β_u_l[1])/math.pi))

M_n1 = M1*np.sin(β_u_l)       # Using M1 = Mn1/sin(β)
M_n2 = np.sqrt((1+((γ-1)/2)*M_n1**2)/((γ*M_n1**2)-((γ-1)/2)))
# Using Normal Shock relation between Mn1 and Mn2
M2 = M_n2/np.sin(β_u_l - Θ)   # Using M2 = Mn2/sin(β-Θ)

print('Post-shock Mach Number (upper), M2 = {}'.format(M2[0]))
print('Post-shock Mach Number (lower), M3 = {}'.format(M2[1]))

def find_Δ(Δ_β):    # Defining a function to find flow direction
downstream of the shock intersection

  Δ = Δ_β[0]        # Δ is the slipline angle
  β4 = Δ_β[1]       # Oblique shock angle for second oblique shock at
upper surface
  β5 = Δ_β[2]       # Oblique shock angle for second oblique shock at
lower surface
  Θ_nu = Θ_u - Δ    # Turning angle for the second oblique shock at
upper surface
  Θ_nl = Θ_l + Δ    # Turning angle for the second oblique shock at
lower surface
```

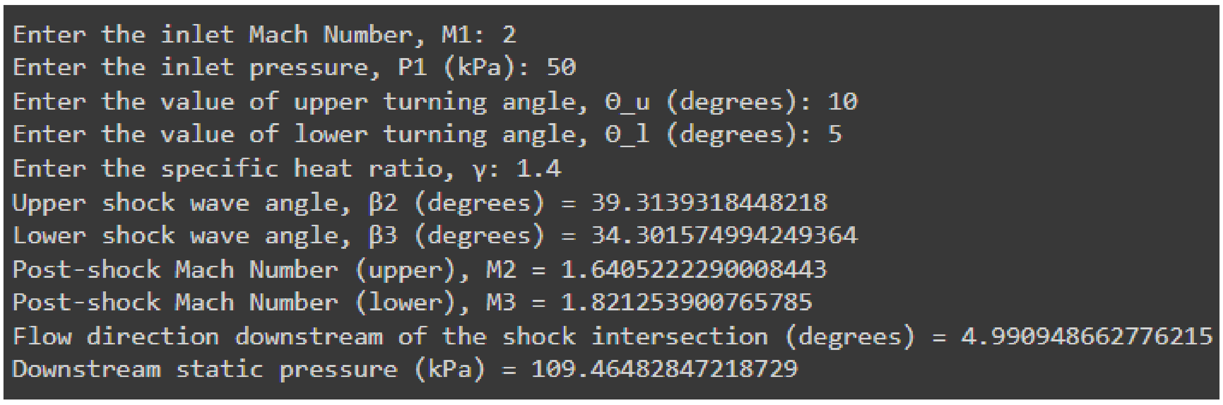
```
  x = math.tan(Θ_nu) -
(2/(math.tan(β4))*((M2[0]**2)*((math.sin(β4))**2) - 1))/(2 +
(M2[0]**2)*(γ + (math.cos(2*β4))))
# Using Θ-β-M relation for second oblique shock

  y = math.tan(Θ_nl) -
(2/(math.tan(β5))*((M2[1]**2)*((math.sin(β5))**2) - 1))/(2 +
(M2[1]**2)*(γ + (math.cos(2*β5))))

  z = (1 + (2*γ/(γ+1))*(((M2[0]*math.sin(β4))**2)-1))*(1 +
(2*γ/(γ+1))*(((M_n1[0])**2)-1)) - (1 +
(2*γ/(γ+1))*(((M2[1]*math.sin(β5))**2)-1))*(1 +
(2*γ/(γ+1))*(((M_n1[1])**2)-1))
# Static pressure remains same across slipline. Using (P4/P2)*(P2/P1) =
(P5/P3)*(P3/P1).

  return np.array([x,y,z])

Δi = np.array([0, β_u_l[0], β_u_l[1]])     # Defining initial values
Δ_β = fsolve(find_Δ, Δi)
P = P1*(1 + (2*γ/(γ+1))*(((M2[0]*math.sin(Δ_β[1]))**2)-1))*(1 +
(2*γ/(γ+1))*(((M_n1[0])**2)-1))
# Using P = (P4/P2)*(P2/P1)*P1 = P4 = P5 across a slipline

print('Flow direction downstream of the shock intersection (degrees) =
{}'.format((180*Δ_β[0])/math.pi))
print('Downstream static pressure (kPa) = {}'.format(P))
```



```
Enter the inlet Mach Number, M1: 3
Enter the inlet pressure, P1 (kPa): 30
Enter the value of upper turning angle, θ_u (degrees): 4
Enter the value of lower turning angle, θ_l (degrees): 3
Enter the specific heat ratio, γ: 1.4
Upper shock wave angle, β2 (degrees) = 22.35442157800068
Lower shock wave angle, β3 (degrees) = 21.598966914008326
Post-shock Mach Number (upper), M2 = 2.7988126655392778
Post-shock Mach Number (lower), M3 = 2.8482352206830046
Flow direction downstream of the shock intersection (degrees) = 0.999525649678656
Downstream static pressure (kPa) = 50.29384781605138
```

5. Thus, the downstream flow direction is almost 1 degree (anti-clockwise) from the freestream direction and the static pressure (downstream) is 50.3 kPa (approx).

6. Example 6.13 of Rathakrishnans' textbook (Gas Dynamics 6th Ed) is also solved using the same code for verification. An arbitrary value of inlet pressure, P_1 is chosen as 50kPa as it is not given in the problem. Therefore, it is found that the slipline angle doesn't vary with the inlet pressure. Textbook answer is 4.5 degree (based on approximations) and output obtained from code is 4.99 degrees (as shown below) which is more accurate.

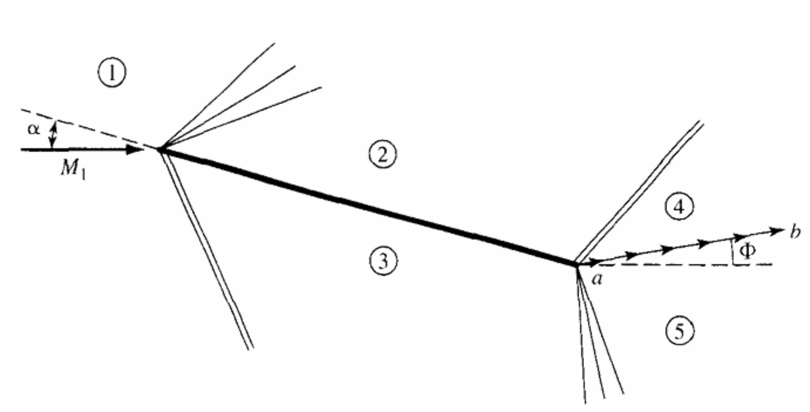


```
Enter the inlet Mach Number, M1: 2
Enter the inlet pressure, P1 (kPa): 50
Enter the value of upper turning angle, θ_u (degrees): 10
Enter the value of lower turning angle, θ_l (degrees): 5
Enter the specific heat ratio, γ: 1.4
Upper shock wave angle, β2 (degrees) = 39.3139318448218
Lower shock wave angle, β3 (degrees) = 34.301574994249364
Post-shock Mach Number (upper), M2 = 1.6405222290008443
Post-shock Mach Number (lower), M3 = 1.821253900765785
Flow direction downstream of the shock intersection (degrees) = 4.990948662776215
Downstream static pressure (kPa) = 109.46482847218729
```

QUESTION 3

Write a code (and run it) to solve Example 4.15 of Anderson (2003) involving calculation of supersonic flow over a flat plate using shock-expansion theory. Calculate the C_l and C_d. Also, calculate the precise slip-line angle (see Fig. 4.37 of the book). Find similar examples from others books to validate the code.

Solution



1. In the given problem, we firstly take the inputs for inlet Mach number, angle of attack of the flat plate and the specific heat ratio.

2. Next, a function is defined to find Prandtl-Meyer functions & Mach No. M_2 behind Expansion Wave (above the upper surface i.e. in region 2). The Prandtl-Meyer relation is given by: -

$$v = \sqrt{\frac{\gamma+1}{\gamma-1}} tan^{-1}\left(\sqrt{\frac{M^2-1}{\gamma+1}}(\gamma-1)\right) - tan^{-1}\sqrt{M^2-1}$$

3. Using isentropic relations across expansion wave, P2/P1 is evaluated. Further, another function is defined to evaluate oblique shock angle and M_3 on lower surface, followed by calculating P3/P1 using normal shock relations. C_l and C_d are then computed using the following relation: -

$$c_l = \frac{L'}{q_1 c} = \frac{2}{\gamma M_1^2}\left(\frac{p_3}{p_1} - \frac{p_2}{p_1}\right) cos\alpha$$

$$c_d = \frac{D'}{q_1 c} = \frac{2}{\gamma M_1^2}\left(\frac{p_3}{p_1} - \frac{p_2}{p_1}\right) sin\alpha$$

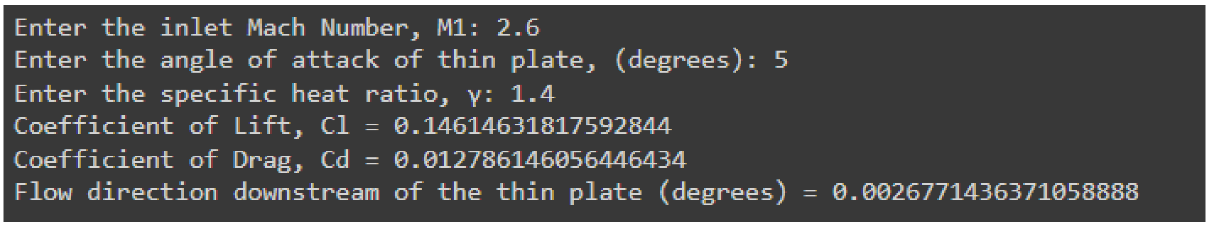
4. A separate function is now defined in python to calculate the slip line angle. As done in the previous problem, an expression is given equating the static pressures above and below the slip line (i.e. (P4/P2) * (P2/P1) = (P5/P3) * (P3/P1)), where P4 and P5 are the static pressures above and below slip line.

5. The code for given problem and output for Example 4.15 of Anderson (2003) is as given below: -

CODE

```
import numpy as np
import math
from scipy.optimize import fsolve

M1 = float(input('Enter the inlet Mach Number, M1: '))
α = float(input('Enter the angle of attack of thin plate, (degrees):
'))
γ = float(input('Enter the specific heat ratio, γ: '))

α = α*math.pi/180 # Converting AOA into radians

def find_M2(M2_v):
# Defining a function to find Prandtl-Meyer functions & Mach No. behind
Expansion Wave (above upper surface)

  M2 = M2_v[0]
  v1 = M2_v[1]
  v2 = M2_v[2]

  x = v1 - (math.sqrt((γ+1)/(γ-1))*math.atan(math.sqrt(((γ-
1)/(γ+1))*(M1**2 -1))) - math.atan(math.sqrt(M1**2 -1)))
# Prantl-Meyer Relation
  y = v2 - (math.sqrt((γ+1)/(γ-1))*math.atan(math.sqrt(((γ-
1)/(γ+1))*(M2**2 -1))) - math.atan(math.sqrt(M2**2 -1)))
  z = v2 - v1 - α
# As α is the turning angle, so α = v2 - v1

  return np.array([x,y,z])

M_vi = np.array([M1,0,α])    # Defining initial values
M2_v = fsolve(find_M2, M_vi)

P2_P1 = ((1+((γ-1)/2)*M1**2)/(1+((γ-1)/2)*M2_v[0]**2))**(γ/(γ-1))
# Using P2/P1 relation for expansion waves (isentropic relation)

def find_M_β(M_β):
# Defining a function to find oblique shock angle and Mach No. behind
oblique shock (below lower surface)

  β2 = M_β[0]
  M_n1 = M_β[1]
# M_n1 = Normal component of M1 to oblique shock & similarly M_n2
  M_n2 = M_β[2]
  x = math.tan(α) - (2/(math.tan(β2))*((M1**2)*((math.sin(β2))**2) -
1))/(2 + (M1**2)*(γ + (math.cos(2*β2))))   # Θ-β-M relation
  y = M_n1 - M1*math.sin(β2)
  z = M_n2 - math.sqrt((1+((γ-1)/2)*M_n1**2)/((γ*M_n1**2)-((γ-1)/2)))
  # Using Normal Shock relation between Mn1 and Mn2

  return np.array([x,y,z])

M_β_i = np.array([α,M1,M1])  # Defining initial values
M_β = fsolve(find_M_β, M_β_i)
M3 = M_β[2]/math.sin(M_β[0] - α)
# Finally evaluating Mach No. behind oblique shock (below lower
surface) using M2 = Mn2/sin(β-Θ)

P3_P1 = 1 + (2*γ/(γ+1))*((M_β[1]**2)-1)
# Using P2/P1 relation for oblique shocks

C_l = (2/(γ*(M1**2)))*(P3_P1 - P2_P1)*math.cos(α)
C_d = (2/(γ*(M1**2)))*(P3_P1 - P2_P1)*math.sin(α)

print('Coefficient of Lift, Cl = {}'.format(C_l))
print('Coefficient of Drag, Cd = {}'.format(C_d))

def find_Δ(Δ_β_v):
# Defining a function to find flow direction downstream of the thin
plate

  Δ = Δ_β_v[0]
  βu = Δ_β_v[1]      # βu is the oblique shock angle on upper trailing
edge
  vl_1 = Δ_β_v[2]    # vl_1 & vl_2 are the prandtl-meyer functions for
expansion wave on lower trailing edge
  vl_2 = Δ_β_v[3]
  M5 = Δ_β_v[4]      # M5 is the Mach no. behind expansion wave on
lower trailing edge

  Θ_nu = α + Δ       # Turning angle for both oblique shock and
expansion wave at the trailing edge

  x = math.tan(Θ_nu) -
(2/(math.tan(βu))*((M2_v[0]**2)*((math.sin(βu))**2) - 1))/(2 +
(M2_v[0]**2)*(γ + (math.cos(2*βu))))    # Θ-β-M relation

  y = vl_1 - (math.sqrt((γ+1)/(γ-1))*math.atan(math.sqrt(((γ-
1)/(γ+1))*(M3**2 -1))) - math.atan(math.sqrt(M3**2 -1)))
# Prantl-Meyer Relation
```

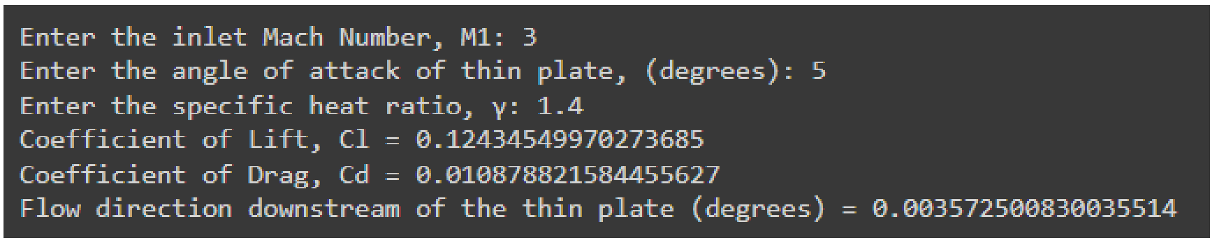
```
  z = vl_2 - (math.sqrt((γ+1)/(γ-1))*math.atan(math.sqrt(((γ-
1)/(γ+1))*(M5**2 -1))) - math.atan(math.sqrt(M5**2 -1)))

  w = vl_2 - vl_1 - Θ_nu

  p = (((1+((γ-1)/2)*M3**2)/(1+((γ-1)/2)*M5**2))**(γ/(γ-1)))*P3_P1 - (1
+ (2*γ/(γ+1))*(((M2_v[0]*math.sin(βu))**2)-1))*P2_P1
# Static pressure remains same across slipline. Using (P5/P3)*(P3/P1) =
(P4/P2)*(P2/P1).

  return np.array([x,y,z,w,p])

Δ_β_v_i = np.array([0, M_β[0], M2_v[1], M2_v[2], M1])
# Defining initial values
Δ_β_v = fsolve(find_Δ, Δ_β_v_i)

print('Flow direction downstream of the thin plate (degrees) =
{}'.format((180*Δ_β_v[0])/math.pi))
```



```
Enter the inlet Mach Number, M1: 2.6
Enter the angle of attack of thin plate, (degrees): 5
Enter the specific heat ratio, γ: 1.4
Coefficient of Lift, Cl = 0.14614631817592844
Coefficient of Drag, Cd = 0.012786146056446434
Flow direction downstream of the thin plate (degrees) = 0.0026771436371058888
```

6. Thus, the values of C_l and C_d found using the code are within 5 percent error of the values calculated in textbook ($C_l = 0.152$ and $C_d = 0.0133$).

7. Example 9.10 from Fundamentals of Aerodynamics (3rd Ed) by Anderson, is also solved using the same code for verification. Textbook answer is $C_l = 0.125$ and $C_d = 0.011$ and output obtained from code is as shown below: -



```
Enter the inlet Mach Number, M1: 3
Enter the angle of attack of thin plate, (degrees): 5
Enter the specific heat ratio, γ: 1.4
Coefficient of Lift, Cl = 0.12434549970273685
Coefficient of Drag, Cd = 0.010878821584455627
Flow direction downstream of the thin plate (degrees) = 0.003572500830035514
```