A616 Gas Dynamics HW – 1, Team 10

1) Mach in terms of the ratio of total temperature to diabatic sonic temperature $\frac{T_0}{T_0^*}$

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

$$\frac{T_0}{T_0^*} = \theta$$

$$\left(1+\gamma M^2\right)^2\theta = 2(\gamma+1)M^2[1+\frac{(\gamma-1)}{2}M^2]$$

$$\left(1+\gamma M^2\right)^2\theta = (\gamma+1)M^2[2+(\gamma-1)M^2]$$

$$(1+2\gamma M^2+\gamma^2M^4)\theta = (\gamma+1)M^2[2+(\gamma-1)M^2]$$

$$(1+2\gamma M^2+\gamma^2M^4)\theta = 2\gamma M^2+2M^2+M^4(\gamma^2-1)$$

$$M^4(\gamma^2\theta-(\gamma^2-1)) + M^2(2\gamma\theta-2\gamma-2)+\theta = 0$$

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

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

$$c = \theta$$

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

$M^2$is obtained, when $b^2-4ac>0$

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

$$\gamma^2+2\gamma+1>(\gamma^2+2\gamma+1)\theta$$

$$\theta<1$$

Two equal roots (M^2), when $b^2-4ac = 0$

$$\gamma^2+2\gamma+1=(\gamma^2+2\gamma+1)\theta$$

$$\theta=1$$

$$\theta\le 1$$

For subsonic solution, $M^2<1$.

For supersonic solution, $M^2 > 1$

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

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

$$b^2 - 4ac \; > b^2$$

$$-ac > 0$$

$$-\big(\gamma^2\theta - (\gamma^2 - 1)\big)\theta \; > 0$$

$$-\gamma^2\theta + \gamma^2 - 1 > 0$$

$$-\gamma^2\theta > -\gamma^2 + 1$$

$$-\gamma^2\theta > -(\gamma^2 - 1)$$

$$\theta > \frac{\gamma^2 - 1}{\gamma^2}$$

$\gamma = 1.4$, then the equation is valid when $\theta > 0.4898$.

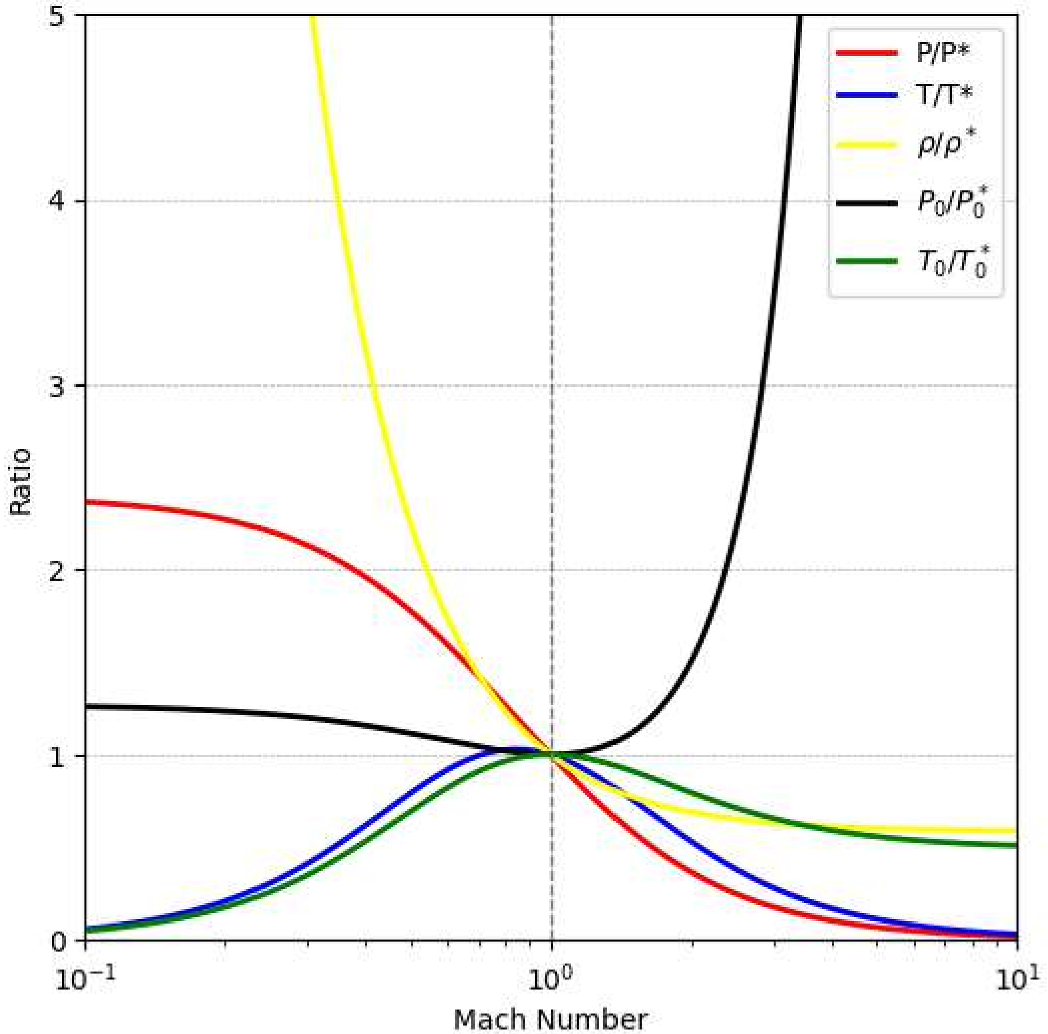
For supersonic solution, $0.4989 < \theta \leq 1$

```
# Question 1
import numpy as np

# Input total temperature / diabatic temperature ratio
T = float(input("Enter theta: "))
gamma = 1.4

# Coefficients for the quadratic equation
a = (gamma**2) * T - (gamma**2) + 1
b = (2 * gamma * T) - (2 * gamma) - 2
c = T

# Calculate the discriminant
d = np.sqrt((b**2) - (4 * a * c))

# Calculate M
M_1 = (-b + d) / (2 * a)  #Supersonic
M_2 = (-b - d) / (2 * a)  #Subsonic

# results
print("M1:", M_1)
print("M2:", M_2)
print("Supersonic, M:" ,round(np.sqrt(M_1),3))
print("Subsonic, M:" ,round(np.sqrt(M_2),3))
```

```
Enter theta: 0.5
M1: 169.8528137423846
M2: 0.14718625761430587
Supersonic, M: 13.033
Subsonic, M: 0.384
```

```
#Question 2
import numpy as np
from matplotlib import pyplot as plt

plt.rcParams['lines.linewidth'] = 2 #line width for all plot

gamma = 1.4                         #Air

# Function
def dia_ratio(M):
    p_r   = (1 + gamma) / (1 + gamma * M**2)
    T_r   = M**2 * ((1 + gamma) / (1 + gamma * M**2))**2
    rho_r = (1/M**2) * ((1 + gamma*M**2) / (1 + gamma))
    p0_r  = p_r * ((2 + (gamma - 1) * M**2) / (gamma + 1))**(gamma / (gamma - 1))
    T0_r  = (((gamma + 1) * M**2) / (1 + gamma * M**2)**2) * (2 + (gamma -1) * M**2

    return p_r, T_r, rho_r, p0_r, T0_r

#Input Mach numbe
x = np.logspace(-1, 1, 500)   # Generates 500 points from 10^-1 to 10^1

#Function Calling to return ratios based on input mach number
```

```
#Function calling to return ratios based on input mach number
p_r, T_r, rho_r, p0_r, T0_r = dia_ratio(x)

plt.figure(figsize=(6, 6))    #figsize
plt.plot(x, p_r, color="red", label="P/P*")
plt.plot(x, T_r, color="blue", label="T/T*")
plt.plot(x, rho_r, color="yellow", label=r"$\rho/\rho^*$")
plt.plot(x, p0_r, color="Black",label=r"$P_0/P_0^*$")
plt.plot(x, T0_r, color="green",label=r"$T_0/T_0^*$")

plt.xscale('log')               # log scale
plt.xlabel('Mach Number')
plt.ylabel('Ratio')

plt.axvline(x=1, color='gray', linestyle='--', linewidth = 1) #reference line at ma
plt.xlim([10**(-1), 10**(1)])   # x-axis limit
plt.ylim([0,5])                 # y-axis limit
plt.grid(True, linestyle= "--", linewidth="0.5")
plt.legend()
plt.show()
```




```
#Question 3 #3.13

import numpy as np
#Inputs
```

```
gamma = 1.4                        #Air
R     = 287
Cp    = (gamma*R) / (gamma-1)
q     = int(input("Enter Heat Added:"))
T_1   = int(input("Enter T1 (k):"))
P_1   = int(input("Enter P1 (atm):"))

# Function for diabatic Ratio
def dia_ratio(M):
    p_r   = (1 + gamma) / (1 + gamma * M**2)
    T_r   = M**2 * ((1 + gamma) / (1 + gamma * M**2))**2
    rho_r = (1/M**2) * ((1 + gamma*M**2) / (1 + gamma))
    p0_r  = p_r * ((2 + (gamma - 1) * M**2) / (gamma + 1))**(gamma / (gamma - 1))
    T0_r  = (((gamma + 1) * M**2) / (1 + gamma * M**2)**2) * (2 + (gamma -1) * M**2

    return p_r, T_r, rho_r, p0_r, T0_r

# Function for Stagnation properties
def Stagnation(M):
  T0_1 = T_1 * (1 + ((gamma-1)/2)*M**2)
  P0_1 = P_1 * (T0_1/T_1)**(gamma / (gamma-1))
  T0_2 = (q/Cp) + T0_1
  T0_2_1 = T0_2/T0_1
  return T0_2_1, T0_1, P0_1, T0_2

# To Calculate diabatic T0_r at M_2
def T0_r_2(M):
  T0_2_1, T0_1, P0_1, T0_2 = Stagnation(M)
  p_r, T_r, rho_r, p0_r, T0_r = dia_ratio(M)
  T0_r2 = T0_2_1 * (T0_r)
  return T0_r2

# To get M2
def Get_M_2(T0_r2, M_1):
  T = T0_r2
  gamma = 1.4
  a = (gamma**2) * T - (gamma**2) + 1
  b = (2 * gamma * T) - (2 * gamma) - 2
  c = T
  d = np.sqrt((b**2) - (4 * a * c))
  if M_1 <= 1:
    M_2 = np.sqrt((-b - d) / (2 * a))  #Subsonic
    return M_2
  elif M_1 > 1:
    M_2 = np.sqrt((-b + d) / (2 * a))  #Supersonic
    return M_2


M_1 = float(input("Enter Mach Number:"))
p_r1, T_r1, rho_r1, p0_r1, T0_r1 = map(lambda x: round(x,4), dia_ratio(M_1))
print("P_r:", p_r1, "T_r:", T_r1, "rho_r:", rho_r1, "p0_r:", p0_r1, "T0_r:", T0_r1)

#Stagnation properties at M_1
T0_2_1, T0_1, P0_1, T0_2 = Stagnation(M_1)
```

```
#Calling T0_r_2(M_1) to get (T0/T0*) at M_2
T0_r2 = T0_r_2(M_1)
print("T0_r2 at M_2:",round(T0_r2,4) )

#Calling Get_M_2(T0_r2, M_1) which gives M_2 based on (T0/T0*) and M_1
print("M_2:", round(Get_M_2(T0_r2, M_1),4))

#Result: Adiabatic Ratios at M_2
M_2 = Get_M_2(T0_r2, M_1)
p_r2, T_r2, rho_r2, p0_r2, T0_r2 = map(lambda x: round(x,4), dia_ratio(M_2))
print("P_r:", p_r2, "T_r:", T_r2, "rho_r:", rho_r2, "p0_r:", p0_r2, "T0_r:", T0_r2)

#P, T, rho at M_2
P_2     = p_r2 * (1/p_r1) * P_1
T_2     = T_r2 * (1/T_r1) * T_1
rho_2   = P_2*(1.01*10**5) / (R * T_2)
P0_2_1  = p0_r2 / p0_r1  #P02/P01
P0_2    = P0_2_1 * P0_1  #P02
print("P_2:",P_2, "T_2:", T_2, "rho_2:", rho_2, "P0_2:", P0_2)
```

```
Enter Heat Added:1000000
Enter T1 (k):273
Enter P1 (atm):1
Enter Mach Number:0.2
P_r: 2.2727 T_r: 0.2066 rho_r: 11.0 p0_r: 1.2346 T0_r: 0.1736
T0_r2 at M_2: 0.8014
M_2: 0.5843
P_r: 1.6239 T_r: 0.9002 rho_r: 1.8039 p0_r: 1.081 T0_r: 0.8014
P_2: 0.7145245742948915 T_2: 1189.518877057115 rho_2: 0.21139042331970917 P0_2: 0.900
```

Start coding or generate with AI.

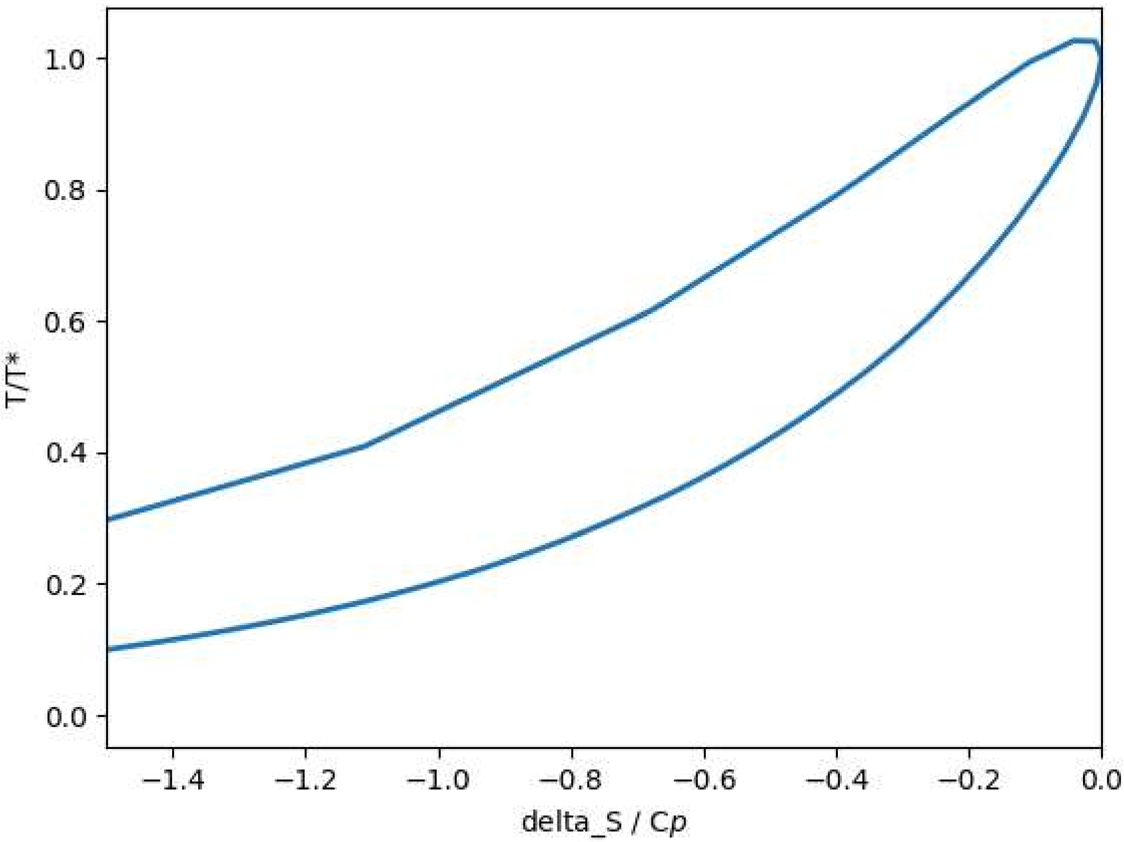
```
#Question 3 #3.14

import numpy as np
#Inputs
gamma = 1.4                        #Air
R     = 287
Cp    = (gamma*R) / (gamma-1)
q     = int(input("Enter Heat Added:"))
T_1   = int(input("Enter T1 (k):"))
P_1   = int(input("Enter P1 (atm):"))

# Function for diabatic Ratio
def dia_ratio(M):
    p_r   = (1 + gamma) / (1 + gamma * M**2)
    T_r   = M**2 * ((1 + gamma) / (1 + gamma * M**2))**2
    rho_r = (1/M**2) * ((1 + gamma*M**2) / (1 + gamma))
    p0_r  = p_r * ((2 + (gamma - 1) * M**2) / (gamma + 1))**(gamma / (gamma - 1))
    T0_r  = (((gamma + 1) * M**2) / (1 + gamma * M**2)**2) * (2 + (gamma -1) * M**2

    return p_r, T_r, rho_r, p0_r, T0_r
```

```
# Function for Stagnation properties
def Stagnation(M):
  T0_1 = T_1 * (1 + ((gamma-1)/2)*M**2)
  P0_1 = P_1 * (T0_1/T_1)**(gamma / (gamma-1))
  T0_2 = (q/Cp) + T0_1
  T0_2_1 = T0_2/T0_1
  return T0_2_1, T0_1, P0_1, T0_2

# To Calculate diabatic T0_r at M_2
def T0_r_2(M):
  T0_2_1, T0_1, P0_1, T0_2 = Stagnation(M)
  p_r, T_r, rho_r, p0_r, T0_r = dia_ratio(M)
  T0_r2 = T0_2_1 * (T0_r)
  return T0_r2

# To get M2
def Get_M_2(T0_r2, M_1):
  T = T0_r2
  gamma = 1.4
  a = (gamma**2) * T - (gamma**2) + 1
  b = (2 * gamma * T) - (2 * gamma) - 2
  c = T
  d = np.sqrt((b**2) - (4 * a * c))
  if M_1 <= 1:
    M_2 = np.sqrt((-b - d) / (2 * a))  #Subsonic
    return M_2
  elif M_1 > 1:
    M_2 = np.sqrt((-b + d) / (2 * a))  #Supersonic
    return M_2


M_1 = float(input("Enter Mach Number:"))
p_r1, T_r1, rho_r1, p0_r1, T0_r1 = map(lambda x: round(x,4), dia_ratio(M_1))
print("P_r:", p_r1, "T_r:", T_r1, "rho_r:", rho_r1, "p0_r:", p0_r1, "T0_r:", T0_r1)

#Stagnation properties at M_1
T0_2_1, T0_1, P0_1, T0_2 = Stagnation(M_1)

#Calling T0_r_2(M_1) to get (T0/T0*) at M_2
T0_r2 = T0_r_2(M_1)
print("T0_r2 at M_2:",round(T0_r2,4) )

#Calling Get_M_2(T0_r2, M_1) which gives M_2 based on (T0/T0*) and M_1
print("M_2:", round(Get_M_2(T0_r2, M_1),4))

#Result: Adiabatic Ratios at M_2
M_2 = Get_M_2(T0_r2, M_1)
p_r2, T_r2, rho_r2, p0_r2, T0_r2 = map(lambda x: round(x,4), dia_ratio(M_2))
print("P_r:", p_r2, "T_r:", T_r2, "rho_r:", rho_r2, "p0_r:", p0_r2, "T0_r:", T0_r2)

#P, T, rho at M_2
P_2     = p_r2 * (1/p_r1) * P_1
T_2     = T_r2 * (1/T_r1) * T_1
rho_2   = P_2*(1.01*10**5) / (R * T_2)
P0_2_1  = p0_r2 / p0_r1  #P02/P01
```

```
P0_2    = P0_2_1 * P0_1  #P02
print("P_2:",P_2, "T_2:", T_2, "rho_2:", rho_2, "P0_2:", P0_2)
```

```
Enter Heat Added:300000
Enter T1 (k):300
Enter P1 (atm):1
Enter Mach Number:3
P_r: 0.1765 T_r: 0.2803 rho_r: 0.6296 p0_r: 3.4245 T0_r: 0.654
T0_r2 at M_2: 0.8865
M_2: 1.5907
P_r: 0.5283 T_r: 0.7063 rho_r: 0.748 p0_r: 1.1702 T0_r: 0.8865
P_2: 2.9932011331444763 T_2: 755.9400642169106 rho_2: 1.3934391709007525 P0_2: 12.552
```

```
#Question 2
import numpy as np
from matplotlib import pyplot as plt

plt.rcParams['lines.linewidth'] = 2 #line width for all plot

gamma = 1.4

def dia_ratio(M):
        p_r   = (1 + gamma) / (1 + gamma * M**2)
        T_r   = M**2 * ((1 + gamma) / (1 + gamma * M**2))**2
        d_s= np.log(T_r) - ((287/1004)*np.log(p_r))
        return d_s, T_r

x= np.arange(0.2,50,0.1)
d_s, T_r = dia_ratio(x)
plt.plot(d_s, T_r)
plt.xlabel("delta_S / C"r'$p$')
plt.ylabel("T/T*")
plt.xlim([-1.5,0])
plt.show()
```




```
# Question 2, P/P* in term of T/T*
import numpy as np
from matplotlib import pyplot as plt

plt.rcParams['lines.linewidth'] = 2  #line width for all plot

gamma = 1.4

# Function
def dia_ratio(M):
        T_r   = M**2 * ((1 + gamma) / (1 + gamma * M**2))**2 # Ratio of Static to D

        # a+b or a-b = Ratio of Pressure to Diabatic pressure in terms of T_r
        a     = (1+gamma**2)/2
        b     = np.sqrt((1+gamma)**2 - 4*gamma*(T_r) )/2
        d_s1   = np.log(T_r) - ((gamma-1)/gamma) *np.log(a+b)
        d_s2   = np.log(T_r) - ((gamma-1)/gamma) *np.log(a-b)
        return d_s1, d_s2, T_r

# Input Mach
x= np.arange(0.1,50,0.001)

# Calling function
d_s1, d_s2, T_r = dia_ratio(x)

#results
plt.plot(d_s1, T_r, color="blue")
plt.plot(d_s2, T_r, color="blue")
plt.xlabel(r"$\Delta S / C_p$")
plt.ylabel(r"$T/T^*$")
```

```
plt.xlim([-1.5,0])
plt.show()
```