Appendix A Problem 2 Python Code

```python
# Compressible Flow
# AEE 553
# Homework 6 - Problem 2
# Evan Burke

import numpy as np
from matplotlib import pyplot as plt
import shocks as ns
import oblique as os
import isentropic as isen
from scipy.optimize import fsolve

pt = 4000*1000 # Pa
Tt = 500 # K
Me = 6 # Design exit Mach
D_th = 4.114 # throat diameter, inviscid, inches
D_th_r = 3.71 # throat diameter, real, viscous, inches

# Convert diameters to meters
D_th = D_th * 0.0254
D_th_r = D_th_r * 0.0254

print(f'Throat Diameter (Inviscid) = {D_th}')
print(f'Throat Diameter (Viscous) = {D_th_r}')

A_th = np.pi * D_th**2/4
print(f'A* = {A_th}')

def mass_flow(pt=None, A_star=None, Tt=None, gamma=1.4, R=287):
    mdot = pt*A_star / Tt**0.5 * (gamma/R * (2/(gamma+1))**((gamma+1)/(gamma-1)))**0.5
    print(f'Choked Mass Flow Rate = {mdot} kg/s')

def A_from_A_star(A_star=None,M=None,gamma=1.4):
    A = ((A_star**2/M**2) * (2/(gamma+1) * (1 + (gamma-1)/2 * M**2 ))**((gamma+1)/(gamma-1)))**0.5
    print(f'Area for M = {M}: {A} m^2')
    return A

# Part A
mdot = mass_flow(pt=pt, A_star=A_th, Tt=Tt, gamma=1.4, R=287)

# Part B
A_exit = A_from_A_star(A_star=A_th,M=Me,gamma=1.4)

# Part C
p_exit = isen.get_static_pressure(M=Me,p_t=pt)
```

```python
T_exit = isen.get_static_temperature(M=Me,T_t=Tt)

# Part D
def A_A_star(M=None,A_A_star=None,gamma=1.4):
    eq = ((1/M**2) * (2/(gamma+1) * (1 + (gamma-1)/2 * M**2 ))**((gamma+1)
    /(gamma-1)))**0.5 - A_A_star
    #print(f'A/A* = {A_A_star}')
    return eq

M_e_sub = float(fsolve(A_A_star,x0=0.01,args=(A_exit/A_th)))
print(f'Subsonic Exit Mach = {M_e_sub}')
p_e_sub = isen.get_static_pressure(M=M_e_sub,p_t=pt)

# Part E

# Normal shock will stand at nozzle exit when
# static pressure is equal to the static pressure
# across a normal shock at the nozzle design condition
p_exit_ns = ns.get_static_pressure_normal_shock(M1=Me,p1=p_exit)
print(f'Back Pressure for which exit NS= {p_exit_ns}')

# Part F
print(f'Back Pressure below which no shocks in nozzle = {p_exit_ns}')

# Part G
# Range of back pressures for which there are oblique shocks
# in nozzle exhaust
# pe < pb
print(f'Range of back pressures for oblique shocks: {p_exit} < p_b < {
    p_exit_ns}')

# Part H
# Range of back pressures for expansion waves
# pb < pe

print(f'Range of back pressures for expansion waves: p_b < {p_exit}')

# Part I

A_avg = (A_th + A_exit)/2
print(f'Average nozzle area = {A_avg} m^2')
M_avg = float(fsolve(A_A_star,x0=1.5,args=(A_avg/A_th)))
print(f'M_avg = {M_avg}')
p_avg = isen.get_static_pressure(M=M_avg,p_t=pt)
p_avg_NS = ns.get_static_pressure_normal_shock(M1=M_avg,p1=p_avg)

# Part J

# Part K
```

Appendix B Problem 3 Python Code

```
# Compressible Flow
# AEE 553
# Homework 6 - Problem 2
# Evan Burke

import numpy as np
from matplotlib import pyplot as plt
import shocks as ns
import oblique as os
import isentropic as isen
from scipy.optimize import fsolve

pt = 20.408 * 10**6 # MPa to Pa
T_i = 3600 # static temp at inlet of CD nozzle, K
A_i = 0.21 # area at inlet of CD nozzle, m^2
A_th = 0.054 # area at throat of CD nozzle, m^2
A_e = 4.17 # area at exit of CD nozzle, m^2
R = 287
gamma = 1.2 # this is different!!!

def mass_flow(pt=None, A_star=None, Tt=None, gamma=1.4, R=287):
    mdot = pt*A_star / Tt**0.5 * (gamma/R * (2/gamma+1)**((gamma+1)/(gamma-1)))**0.5
    print(f'Choked Mass Flow Rate = {mdot} kg/s')

def A_from_A_star(A_star=None,M=None,gamma=1.4):
    A = ((A_star**2/M**2) * (2/(gamma+1) * (1 + (gamma-1)/2 * M**2 ))**((gamma+1)/(gamma-1)))**0.5
    print(f'Area for M = {M}: {A} m^2')
    return A

altitudes = np.linspace(0,20000,num=20001, endpoint=True)
print(altitudes)

pressures = [101325*(1-(2.25577*10**(-5)*h))**5.25588 for h in altitudes]
print(pressures[0:10])

def SSME_thrust(mdot=None,u_e=None,p_e=None,p_amb=None,A_e=None):
    thrust = mdot * u_e + (p_e-p_amb)*A_e
    pass
# To get mdot:
# Need pt, A_th, Tt, gamma, R
# pt is given, Tt unknown, M_th = 1, A_i/A_th known
# Solve for M_i from A_i/A_th given that M_th = 1

def A_A_star(M=None,A_A_star=None,gamma=1.4):
    eq = ((1/M**2) * (2/(gamma+1) * (1 + (gamma-1)/2 * M**2 ))**((gamma+1)
```

```
    /(gamma-1)))**0.5 - A_A_star
    return eq

M_i = float(fsolve(A_A_star,x0=0.01,args=(A_i/A_th,gamma)))
print(f'Nozzle Inlet Mach = {M_i}')
Tt = isen.get_total_temperature(M=M_i,T=T_i,gamma=gamma)
print(f'Total temperature at nozzle inlet = {Tt} K')

mdot = mass_flow(pt=pt,A_star=A_th,Tt=Tt,gamma=gamma,R=R)

# mdot, A_e, p_ambs known, need u_e and p_e
# Use A/A* relationship to get exit Mach number, exit sonic velocity, exit
    velocity
```