Appendix A Problem 1 Python Code

```
# Compressible Flow
# AEE 553
# Homework 4 - Problem 1
# Evan Burke

import numpy as np
from matplotlib import pyplot as plt
import shocks as ns
import isentropic as isen

# b
gamma = 1.4

def hugoniot(gamma=1.4,rho_ratio=None):
    p2_p1 = ( (gamma+1)/(gamma-1) * rho_ratio - 1) / ((gamma+1)/(gamma-1)
    - rho_ratio)
    return p2_p1

rho_ratio = np.linspace(1,5,endpoint=True)

p2_p1_h = [hugoniot(rho_ratio=r) for r in rho_ratio] # normal shock
p2_p1_i = [r**gamma for r in rho_ratio]

fig,ax = plt.subplots()
ax.plot(rho_ratio,p2_p1_h,label='Hugoniot - NS')
ax.plot(rho_ratio,p2_p1_i,label='Isentropic')
ax.legend()
ax.set_xlabel('Density Ratio')
ax.set_ylabel('Pressure Ratio')
ax.set_title('Compression vs. Density Ratio')
plt.savefig('../images/problem_1/hugoniot_vs_isentropic_compression.png')
plt.close()

machs = np.linspace(1,6,endpoint=True)
print(machs)
pt2_pt1 = [ns.get_total_pressure_ratio_normal_shock(M1=m) for m in machs]
    #get_total_pressure worked without a static pressure? need error
    handling
print(pt2_pt1)
fig,ax = plt.subplots()
ax.plot(machs,pt2_pt1)
ax.set_xlabel('Mach')
ax.set_ylabel('Total Pressure Ratio')
ax.set_title('Normal Shock Total Pressure Ratio vs. Mach Number')
plt.savefig('../images/problem_1/compression_efficiency_NS.png')
plt.close()

```

```python
# d
cp = 1004.5
R = 287
ds_isen = [(cp * np.log(r**(gamma-1)) - R * np.log(pr)) for r,pr in zip(
    rho_ratio,p2_p1_i)]

m1 = [ns.get_upstream_mach_normal_shock(p2_p1=pr) for pr in p2_p1_h]
pt2_pt1 = [ns.get_total_pressure_ratio_normal_shock(M1=m) for m in m1]
ds_ns = [-R*np.log(ptr) for ptr in pt2_pt1]
print(ds_isen)
print(ds_ns)

fig,ax = plt.subplots()

ax.plot(rho_ratio,ds_isen,label='Isentropic')
ax.plot(rho_ratio,ds_ns,label='Normal Shock')
ax.set_title('Entropy Change vs. Density Ratio')
ax.set_xlabel('Density ratio')
ax.set_ylabel('Entropy Change [kJ/kg*K]')
ax.legend()

plt.savefig('../images/problem_1/entropy_change.png')

pr_crit = hugoniot(rho_ratio=2.5)
m1_crit = ns.get_upstream_mach_normal_shock(p2_p1=pr_crit)
print(f'Critical Mach: {m1_crit}')
```

Appendix B Problem 2 Python Code

```
# Compressible Flow
# AEE 553
# Homework 4 - Problem 2
# Evan Burke

from gettext import find
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

class SimpleRamjet:

    def __init__(self,M1,theta,q,cp,gamma,T,delta):
        self.M1 = M1
        self.theta = theta
        self.T=T
        self.delta=delta
        self.beta = SimpleRamjet.find_beta(M=self.M1,theta=self.theta)
        self.M1n = os.get_m1_normal(M1=self.M1,beta=self.beta)
        self.M2n = os.get_m2_normal(M1n=self.M1n)
        self.p2_p1 = ns.get_static_pressure_ratio_normal_shock(M1=self.M1n
    )
        self.pt2_pt1 = ns.get_total_pressure_ratio_normal_shock(M1=self.
    M1n)
        self.M2 = os.get_m2(M2n=self.M2n,beta=self.beta,theta=self.theta)
        self.M3 = ns.get_mach_normal_shock(M1=self.M2)
        self.p3_p2 = ns.get_static_pressure_ratio_normal_shock(M1=self.M2)
        self.pt3_pt2 = ns.get_total_pressure_ratio_normal_shock(M1=self.M2
    )
        self.p1_p3 = 1/self.p2_p1 * 1/self.p3_p2
        self.pt3_pt1 = self.pt2_pt1 * self.pt3_pt2
        self.T2_T1 = ns.get_static_temperature_ratio_normal_shock(M1=self.
    M1n)
        self.T3_T2 = ns.get_static_temperature_ratio_normal_shock(M1=self.
    M2)
        self.T3_T1 = self.T3_T2 * self.T2_T1
        self.T3 = self.T3_T2 * self.T2_T1 * self.T
        self.Tt3 = isen.get_total_temperature(M=self.M3,T=self.T3)
        self.q = q
        self.cp = cp
        self.Tt4 = self.Tt3 + (1000*self.q)/self.cp
        self.Tt4_Tt3 = self.Tt4/self.Tt3
        self.T4 = float(isen.get_static_temperature(M=self.M3,T_t=self.Tt4
    ))
```

```python
        self.gamma=gamma
        self.eta = 1 - ((self.p1_p3)**((self.gamma-1)/self.gamma) * (self.T4 - (self.pt3_pt1)**((self.gamma-1)/self.gamma)*self.T3) / (self.T4-self.T3))
        print(f'Ramjet efficiency = {self.eta}')

    def find_beta(M=None,gamma=1.4,theta=None,delta=1):
        theta = np.deg2rad(theta)
        lamb = ((M**2-1)**2 - 3*(1 + (gamma-1)/2*M**2) * (1 + (gamma+1)/2*M**2) * np.tan(theta)**2)**0.5
        chi = ((M**2-1)**3 - 9 * (1 + (gamma-1)/2 * M**2) * (1 + (gamma-1)/2 * M**2 + (gamma+1)/4*M**4)*np.tan(theta)**2)/lamb**3
        tan_beta = (M**2 - 1 + 2*lamb*np.cos((4*np.pi*delta+np.arccos(chi))/3)) / (3 * (1 + (gamma-1)/2*M**2)*np.tan(theta))
        beta = np.arctan(tan_beta)
        beta = np.rad2deg(beta)
        print(f'Shock angle = {beta}')
        return beta

if __name__=='__main__':

    M = 3
    T = 217 # K
    p = 20000 # Pa
    gamma = 1.4
    R = 287 # J/kg K
    cp = 1000 # J/kg K
    q = 500 # kJ/kg

    thetas = np.linspace(1,34,num=67,endpoint=True)

    delta = 1 # weak shock solution
    betas = [SimpleRamjet.find_beta(M=M,theta=th) for th in thetas if not np.isnan(SimpleRamjet.find_beta(M=M,theta=th))]
    ramjets = [SimpleRamjet(M1=M,theta=th,q=q,cp=cp,gamma=1.4,T=T,delta=delta) for th in thetas]
    efficiencies = [ramjet.eta for ramjet in ramjets]

    max_eta = max(efficiencies)
    print(f'Max efficiency = {max_eta}')
    idx_max = efficiencies.index(max_eta)
    print(f'idx max = {idx_max}')
    theta_ideal = thetas[idx_max]
    print(f'Ideal half angle = {theta_ideal}')

    fig,ax = plt.subplots()
    ax.set_title("Ramjet Cycle Efficiency vs. Inlet Half Angle")
    ax.set_xlabel('Inlet Half Angle [degrees]')
    ax.set_ylabel('Ramjet Efficiency')
    plt.plot(thetas,efficiencies,'-')
```

```
    plt.plot(theta_ideal,max_eta,'r*')
    plt.savefig('../images/problem_2/idealtheta.png')

    # c

    M2 = M
    M3 = ns.get_mach_normal_shock(M1=M2)
    p3_p2 = ns.get_static_pressure_ratio_normal_shock(M1=M2)
    pt3_pt2 = ns.get_total_pressure_ratio_normal_shock(M1=M2)
    p2_p3 = 1/p3_p2
    pt3_pt1 = pt3_pt2
    T3_T2 = ns.get_static_temperature_ratio_normal_shock(M1=M2)
    T3_T1 = T3_T2
    print(f'T3_T1 = {T3_T1}')
    T3 = T3_T2 * T
    print(f'T3 = {T3}')
    Tt3 = isen.get_total_temperature(M=M3,T=T3)
    Tt4 = Tt3 + (1000*q)/cp
    print(f'Tt4 = {Tt4}')
    Tt4_Tt3 = Tt4/Tt3
    print(f'Tt4/Tt3 = {Tt4_Tt3}')
    T4 = float(isen.get_static_temperature(M=M3,T_t=Tt4))
    eta_NS = 1 - ((p2_p3)**((gamma-1)/gamma) * (T4 - (pt3_pt2)**((gamma-1)
   /gamma)*T3) / (T4-T3))
    print(f'Scramjet efficiency -- no spike = {eta_NS}')


    # e

    machs = np.linspace(2,6,num=21,endpoint=True)
    ramjets_mach = [SimpleRamjet(M1=Mi,theta=theta_ideal,q=q,cp=cp,gamma
   =1.4,T=T,delta=delta) for Mi in machs]
    efficiencies = [ramjets.eta for ramjets in ramjets_mach]

    max_eta = max(efficiencies)
    idx_ideal = efficiencies.index(max_eta)
    ideal_mach = machs[idx_ideal]

    print(f'Ideal Mach = {ideal_mach}, ideal efficiency = {max_eta}')

    fig,ax = plt.subplots()
    ax.set_title("Ramjet Cycle Efficiency vs. Cruise Mach")
    ax.set_xlabel('Cruise Mach')
    ax.set_ylabel('Ramjet Efficiency')
    ax.set_ylim(bottom=0,top=1)
    plt.plot(machs,efficiencies,'-')
    plt.plot(ideal_mach,max_eta,'r*')
    plt.savefig('../images/problem_2/eta_vs_mach.png')
    plt.close()
    print(f'Max efficiency = {max_eta}')
```

```

    pt3_pt1s = [ramjets.pt3_pt1 for ramjets in ramjets_mach]

    fig,ax = plt.subplots()
    ax.set_title("Inlet Total Pressure Ratio vs. Cruise Mach")
    ax.set_xlabel('Cruise Mach')
    ax.set_ylabel('Inlet Total Pressure Ratio')
    plt.plot(machs,pt3_pt1s,'-')
    plt.savefig('../images/problem_2/tpr_vs_mach.png')

    p3_p1s = [1/ramjets.p1_p3 for ramjets in ramjets_mach]

    fig,ax = plt.subplots()
    ax.set_title("Inlet Static Pressure Ratio vs. Cruise Mach")
    ax.set_xlabel('Cruise Mach')
    ax.set_ylabel('Inlet Static Pressure Ratio')
    plt.plot(machs,p3_p1s,'-')
    plt.savefig('../images/problem_2/pr_vs_mach.png')

    T3s = [ramjets.T3 for ramjets in ramjets_mach]

    fig,ax = plt.subplots()
    ax.set_title("Combustor Inlet Static Temperature vs. Cruise Mach")
    ax.set_xlabel('Cruise Mach')
    ax.set_ylabel('Combustor Inlet Static Temperature')
    plt.plot(machs,T3s,'-')
    plt.savefig('../images/problem_2/t3_vs_mach.png')
```

Appendix C Problem 3 Python Code

```python
# Compressible Flow
# AEE 553
# Homework 4 - Problem 3
# Evan Burke

from re import T
import numpy as np
from matplotlib import pyplot as plt
import shocks as ns
import isentropic as isen
import oblique as os
from homework_4_problem_2 import SimpleRamjet

M = 5
T = 217
p = 20000

theta1 = 7 #treat this as positive
delta = 1 # weak shock solution

def find_beta(M=None,gamma=1.4,theta=None):
    theta = np.deg2rad(theta)
    lamb = ((M**2-1)**2 - 3*(1 + (gamma-1)/2*M**2) * (1 + (gamma+1)/2*M
    **2) * np.tan(theta)**2)**0.5
    chi = ((M**2-1)**3 - 9 * (1 + (gamma-1)/2 * M**2) * (1 + (gamma-1)/2 *
    M**2 + (gamma+1)/4*M**4)*np.tan(theta)**2)/lamb**3
    tan_beta = (M**2 - 1 + 2*lamb*np.cos((4*np.pi*delta+np.arccos(chi))/3)
    ) / (3 * (1 + (gamma-1)/2*M**2)*np.tan(theta))
    beta = np.arctan(tan_beta)
    beta = np.rad2deg(beta)
    print(f'Shock angle = {beta}')
    return beta

beta1 = find_beta(M=M,theta=theta1)
M1n = os.get_m1_normal(M1=M,beta=beta1)
M2n = os.get_m2_normal(M1n=M1n)
M2 = os.get_m2(M2n=M2n,beta=beta1,theta=theta1)

theta2 = 7
beta2 = find_beta(M=M2,theta=theta2)
M2np = os.get_m1_normal(M1=M2,beta=beta2)
M3n = os.get_m2_normal(M1n=M2np)
M3 = os.get_m2(M2n=M3n,beta=beta2,theta=theta2)

pt = isen.get_total_pressure(M=M,p=p)
p2_p1 = ns.get_static_pressure_ratio_normal_shock(M1=M1n)
pt2_pt1 = ns.get_total_pressure_ratio_normal_shock(M1=M1n)
```

```python
p3_p2 = ns.get_static_pressure_ratio_normal_shock(M1=M2np)
pt3_pt2 = ns.get_total_pressure_ratio_normal_shock(M1=M2np)
T2_T1 = ns.get_static_temperature_ratio_normal_shock(M1=M1n)
T3_T2 = ns.get_static_temperature_ratio_normal_shock(M1=M2np)

p3_p1 = p2_p1*p3_p2
pt3_pt1 = pt3_pt2*pt2_pt1
T3 = T3_T2 * T2_T1 * T
print(f'T3 = {T3}')
p3 = p3_p1 * p
pt3 = pt3_pt2*pt2_pt1*pt
ramjet_m5 = SimpleRamjet(M1=M,theta=21,q=500,cp=1000,gamma=1.4,T=T,delta=
    delta)
print('\n\n')
print(f'Inlet Static Pressure Ratio:\nRamjet = {1/ramjet_m5.p1_p3}\
    nScramjet = {p3_p1}\n')
print(f'Inlet Total Pressure Ratio:\nRamjet = {ramjet_m5.pt3_pt1}\
    nScramjet = {pt3_pt1}\n')
print(f'Combustor Inlet Static Temperature:\nRamjet = {ramjet_m5.T3}\
    nScramjet = {T3}')

M3 = isen.get_mach_number(p_t_ratio=pt3/p3)
print(f'Combustor Mach = {M3}')
```