```
#EAS 4300 Final Project
#Grayson Savage
#G2434773

from sympy.solvers import solve
from sympy import Symbol, sqrt
import math
import matplotlib.pyplot as plt
import numpy as np

#Solves Mach Number for A/A*
#M = Symbol('M')
#solve(((1/M)*((2/(g+1)*(1+(((g-1)/2)*M**2)))**((g+1)/(2*(g-1))))-3.23,M)

# Conditions given in the problem
Pa = 18.823
Ta = 216.65
To4_max = 2300
gamma1 = 1.4
#gamma2 = 1.35
R = 287
Cp1 = (gamma1/(gamma1-1))*R/1000
#Cp2 = (gamma2/(gamma2-1))*R/1000
Ma=4.0
Fst=0.06
#hc=44000
A6=1.0

#Efficiencies
rd=(1-0.075*(Ma-1)**(1.35))
print 'rd'
print rd

#Setup Arrays for Graphs
Mlist = []
Ilist = []
TSFClist = []
nthlist = []
nplist = []
nolist = []
Aratlist = []
Thrustlist = []
rblist = []
fblist = []

#Flow Conditions
Toa = Ta*(1 + ((gamma1-1)/2)*Ma**2)
print 'Toa '
print Toa
Poa = Pa*(1 + ((gamma1-1)/2)*Ma**2)**(gamma1/(gamma1-1))
print 'Poa '
print Poa
u = Ma*sqrt(gamma1*R*Ta)
print 'u '
print u

#Inlet/Diffuser
To2=Toa
print 'To2 '
print To2
Po2=Poa*rd
print 'Po2 '
print Po2
```

```
To4 = To4_max

Mx = np.arange(2.0,6.0,0.1)

for Arat in np.arange(1.1,2.5,0.1):
    print '--------A/A*--------'
    print Arat
    M = Symbol('M')
    M_solve =
solve(((1/M)*((2/(gamma1+1)*(1+(((gamma1-1)/2)*M**2)))**((gamma1+1)/(2*(gamma1-1))))
-Arat),M)
    M4 = M_solve[0]
    print 'M4 '
    print M4
    T4 = To4/(1+((gamma1-1)/2)*M4**2)
    print 'T4 '
    print T4
    P4rat = (To4/T4)**(gamma1/(gamma1-1))
    print 'PO4/P4 '
    print P4rat
    u4 = M4*sqrt(gamma1*R*T4)
    print 'u4 '
    print u4
    T2 = Symbol('T2')
    T_solve =
solve(((sqrt(gamma1*R*(To2-T2)*(2/(gamma1-1)))*(T4+((u4**2)/R)))/u4-(gamma1*(To2-T2)
*(2/(gamma1-1))))-T2,T2)
    T2 = T_solve[1]
    print 'T2 '
    print T2

    M2 = Symbol('M2')
    M2_solve =  solve((1 + ((gamma1-1)/2)*(M2**2))-(To2/T2),M2)
    M2 = M2_solve[1]
    print 'M2'
    print M2
    P2 = Po2/((To2/T2)**(gamma1/(gamma1-1)))
    print 'P2'
    print P2
    u2 = M2*sqrt(gamma1*R*T2)
    print 'u2'
    print u2

    P4 = Symbol('P4')
    P4_solve = solve(((u2*(P2/T2))/(u4*(P4/T4))) - Arat,P4)
    P4 = P4_solve[0]
    print 'P4'
    print P4
    Po4 = P4*P4rat
    print 'Po4 '
    print Po4

    Po6 = Po4
    print 'Po6'
    print Po6
    To6 = To4
    print 'To6'
    print To6
    P6 = Pa
    print 'P6'
    print P6
```

```
    T6 = To6/((Po6/P6)**((gamma1-1)/gamma1))
    print 'T6'
    print T6
    M6 = Symbol('M6')
    M6_solve =  solve(1 + ((gamma1-1)/2)*(M6**2)-(To6/T6),M6)
    M6 = M6_solve[1]
    u6 = M6*sqrt(gamma1*R*T6)
    print 'u6'
    print u6
    print "M6"
    print M6
    Aexitrat = 
((1/M6)*((2/(gamma1+1)*(1+(((gamma1-1)/2)*M6**2)))**((gamma1+1)/(2*(gamma1-1)))))
    print 'Aexit/A*'
    print Aexitrat
    A1 = 1/Aexitrat
    print 'A1=A2=A5'
    print A1
    ma = (Pa*1000/(R*Ta))*u*A1
    print 'ma'
    print ma
    Thrust = ma*(u6-u)
    print 'Thrust'
    print Thrust

    I = Thrust/ma
    TSFC = 1.06/I
    Pav = ma*((1.06)*((u6**2)/2)-((u**2)/2))
    Pin=ma*1.06*44000*1000
    wp=Thrust*u
    nth=Pav/Pin
    np=wp/Pav
    no=nth*np

    Thrustlist.append([Thrust])
    TSFClist.append([TSFC])
    Aratlist.append([Arat])

    if Thrust > 44482:
        break

plt.figure(0)
plt.subplot(211)
plt.plot(Aratlist, TSFClist)
#plt.xlabel('A4/A* Ratio')
plt.ylabel('TSFC')
plt.title('TSFC vs A4/A*')

plt.subplot(212)
plt.plot(Aratlist, Thrustlist)
plt.xlabel('A4/A* Ratio')
plt.ylabel('Thrust')
plt.title('Thrust vs A4/A*')

Thrustlist = []
TSFClist = []

for Ma in Mx:
    print '--------Ma--------'
    print Ma
    rd=(1-0.075*(Ma-1)**(1.35))
    print 'rd'
    print rd
```

```
    #Flow Conditions
    Toa = Ta*(1 + ((gamma1-1)/2)*Ma**2)
    print 'Toa '
    print Toa
    Poa = Pa*(1 + ((gamma1-1)/2)*Ma**2)**(gamma1/(gamma1-1))
    print 'Poa '
    print Poa
    u = Ma*sqrt(gamma1*R*Ta)
    print 'u '
    print u

    #Inlet/Diffuser
    To2=Toa
    print 'To2 '
    print To2
    Po2=Poa*rd
    print 'Po2 '
    print Po2

    To4 = To4_max

    M = Symbol('M')
    M_solve = 
solve(((1/M)*((2/(gamma1+1)*(1+(((gamma1-1)/2)*M**2)))**((gamma1+1)/(2*(gamma1-1))))
-Arat),M)
    M4 = M_solve[0]
    print 'M4 '
    print M4
    T4 = To4/(1+((gamma1-1)/2)*M4**2)
    print 'T4 '
    print T4
    P4rat = (To4/T4)**(gamma1/(gamma1-1))
    print 'P04/P4 '
    print P4rat
    u4 = M4*sqrt(gamma1*R*T4)
    print 'u4 '
    print u4
    T2 = Symbol('T2')
    T_solve = 
solve(((sqrt(gamma1*R*(To2-T2)*(2/(gamma1-1)))*(T4+((u4**2)/R)))/u4-(gamma1*(To2-T2)
*(2/(gamma1-1))))-T2,T2)
    T2 = T_solve[1]
    print 'T2 '
    print T2

    M2 = Symbol('M2')
    M2_solve =  solve((1 + ((gamma1-1)/2)*(M2**2))-(To2/T2),M2)
    M2 = M2_solve[1]
    print 'M2'
    print M2
    P2 = Po2/((To2/T2)**(gamma1/(gamma1-1)))
    print 'P2'
    print P2
    u2 = M2*sqrt(gamma1*R*T2)
    print 'u2'
    print u2

    P4 = Symbol('P4')
    P4_solve = solve(((u2*(P2/T2))/(u4*(P4/T4))) - Arat,P4)
    P4 = P4_solve[0]
    print 'P4'
    print P4
    Po4 = P4*P4rat
```

```
    print 'Po4 '
    print Po4

    Po6 = Po4
    print 'Po6'
    print Po6
    To6 = To4
    print 'To6'
    print To6
    P6 = Pa
    print 'P6'
    print P6

    T6 = To6/((Po6/P6)**((gamma1-1)/gamma1))
    print 'T6'
    print T6
    M6 = Symbol('M6')
    M6_solve =  solve(1 + ((gamma1-1)/2)*(M6**2)-(To6/T6),M6)
    M6 = M6_solve[1]
    u6 = M6*sqrt(gamma1*R*T6)
    print 'u6'
    print u6
    print "M6"
    print M6
    Aexitrat =
((1/M6)*((2/(gamma1+1)*(1+(((gamma1-1)/2)*M6**2)))**((gamma1+1)/(2*(gamma1-1)))))
    print 'Aexit/A*'
    print Aexitrat
    A1 = 1/Aexitrat
    print 'A1=A2=A5'
    print A1
    ma = (Pa*1000/(R*Ta))*u*A1
    print 'ma'
    print ma
    Thrust = ma*(u6-u)
    print 'Thrust'
    print Thrust

    fb = 0.06
    rb = Po4/Po2
    I = Thrust/ma
    print 'I'
    print I
    TSFC = 1.06/I
    print 'TSFC'
    print TSFC
    Pav = ma*((1.06)*((u6**2)/2)-((u**2)/2))
    Pin=ma*1.06*44000*1000
    wp=Thrust*u
    nth=Pav/Pin
    np=wp/Pav
    no=nth*np

    fblist.append([fb])
    rblist.append([rb])
    Mlist.append([Ma])
    Thrustlist.append([Thrust])
    Ilist.append([I])
    TSFClist.append([TSFC])
    nthlist.append([nth])
    nplist.append([np])
    nolist.append([no])
```

```
# Now to plot everything!
plt.figure(1)
plt.subplot(211)
plt.plot(Mlist, Thrustlist)
#plt.xlabel('Ma')
plt.ylabel('Thrust')
plt.title('Thrust vs Ma')

plt.subplot(212)
plt.plot(Mlist, Ilist)
plt.xlabel('Ma')
plt.ylabel('Specific Thrust, I')
plt.title('I vs Ma')

plt.figure(2)
plt.plot(Mlist, TSFClist)
plt.xlabel('Ma')
plt.ylabel('TSFC')
plt.title('TSFC vs Ma')

plt.figure(3)
plt.subplot(311)
plt.plot(Mlist, nthlist)
#plt.xlabel('Ma')
plt.ylabel('Thermal Efficiency, nth')
plt.title('Thermal Efficiency vs Ma')

plt.subplot(312)
plt.plot(Mlist, nplist)
#plt.xlabel('Ma')
plt.ylabel('Propulsive Efficiency, np')
plt.title('Propulsive Efficiency vs Ma')

plt.subplot(313)
plt.plot(Mlist, nolist)
plt.xlabel('Ma')
plt.ylabel('Overall Efficiency, no')
plt.title('Overall Efficiency vs Ma')

plt.figure(4)
plt.subplot(211)
plt.plot(Mlist, fblist)
#plt.xlabel('Ma')
plt.ylabel('Fuel/Air, Fb')
plt.title('Fuel/Air Ratio vs Ma')

plt.subplot(212)
plt.plot(Mlist, rblist)
plt.xlabel('Ma')
plt.ylabel('Burner Pressure Ratio, rb')
plt.title('Burner Pressure Ratio vs Ma')

plt.show()
```