MAJOR PROJECT # 3

Shaun Harris
Department of Mechanical and Aerospace Engineering
Utah State University
Email: shaun.r.harris@gmail.com

ABSTRACT

A fixed Earth Gravity Cancellation (EGC) Rocket was examined. The fuel type was considered both as a monopropellant and as a hybrid ABS rocket. The fuel types were compared and examined and the resulting analysis and conclusions are presented here.

CONTENTS

NOMENCLATURE

I_{sp} Specific impulse of rocket (seconds)
M_W Molecular weight
g_0 Gravity at sea level 9.806 $\frac{m}{s^2}$
P_0 Stagnation Pressure (Pa)
T_0 Stagnation Temperature or flame temperature (K)
A_{exit} Nozzle exit area
A^* or A_t Nozzle throat area
$\frac{A_{exit}}{A_t}$ exit area over throat area ratio
t_2 time in seconds for ABS hybrid rocket fuel time
t_1 time in seconds for Monopropellant rocket
C^* Characteristic velocity $(\frac{m}{s})$
$\dot{m}$ Mass flow
$\frac{O}{F}$ Oxidizer to Fuel ratio

1 INTRODUCTION

1.1 Part 1

The EGC rocket can use a hydrogen peroxide and water combination for the oxide and fuel. This type of rocket is considered a monopropellant. It can used at varying pressure values, and varying oxide to fuel ratios. These ratios were examined and considered from 80% to 99%. The nozzle outlet has a 4:1 nozzle exapansion. NASA developed a code called CEA, this solver was used to calculate the various desired values for this case. It calculated the Nozzle Exit Temperature, which was then compared to a separate calculated isentropic stagnation temperature. The stagnation temperatures were calculated, and the nozzle exit and combuster temperatures were then compared. The nozzle exit Mach number was calculated and shown from CEA. The fuel to oxide ratio was then analyzed from the CEA output to show when all the water in the peroxide solution was completely vaporized with this nozzle and atmospheric pressure. Based on a thrust level of 3114 N the throat area was then calculated from the CEA output. The corresponding I_{sp} and $C*$ values were calculated. The mass flow corresponding to this value was also calculated.

1.2 Part 2

The EGC rocket can also use an ABS mixture as the fuel, and the hydrogen peroxide and water as the oxidizers with varying ratios. The optimal $C*$ values were considered as the optimal conditions. The corresponding I_{sp} and $\frac{O}{F}$ values were considered. The values of the hybrid rocket and the monopropellant were than compared.

1.3 Part 3

The hover time of the EGC rocket were then considered. The flight time was chosen to be the rocket consideration. If the mass fraction is similar for each rocket, then Eq. 1 shows the relationship between flight time and I_{sp} values.

$$
\begin{aligned}
\Delta V_2 &= gt_2 = g_0 I_{sp2} ln(\frac{M_i}{M_f}) \\
\Delta V_1 &= gt_1 = g_0 I_{sp1} ln(\frac{M_i}{M_f}) \\
\frac{\Delta V_2}{\Delta V_1} &= \frac{t_2}{t_1} = \frac{I_{sp2}}{I_{sp1}}
\end{aligned}
\tag{1}
$$

2 RESULTS

2.1 Part 1

The pressure was iterated upon until the resulting nozzle exit pressure was close to atmospheric pressure at sea level. The chamber pressure was found to be approximately 28.5 bar. With this pressure, CEA was simulated, and many of the calculations were used from the CEA output. Section A.1 shows the configuration file used for the calculations.

The nozzle exit temperature and stagnation temperature were compared in the left part of Fig. 1. The plots x-axis shows the percent of mass concentration for the hydrogen peroxide. We can see the nozzle exit temperature is significantly smaller than the stagnation temperature. The right part of Fig. 1 shows the comparison of the combuster and nozzle exit stagnation temperatures. These stagnation temperatures were calculated using isentropic relations.

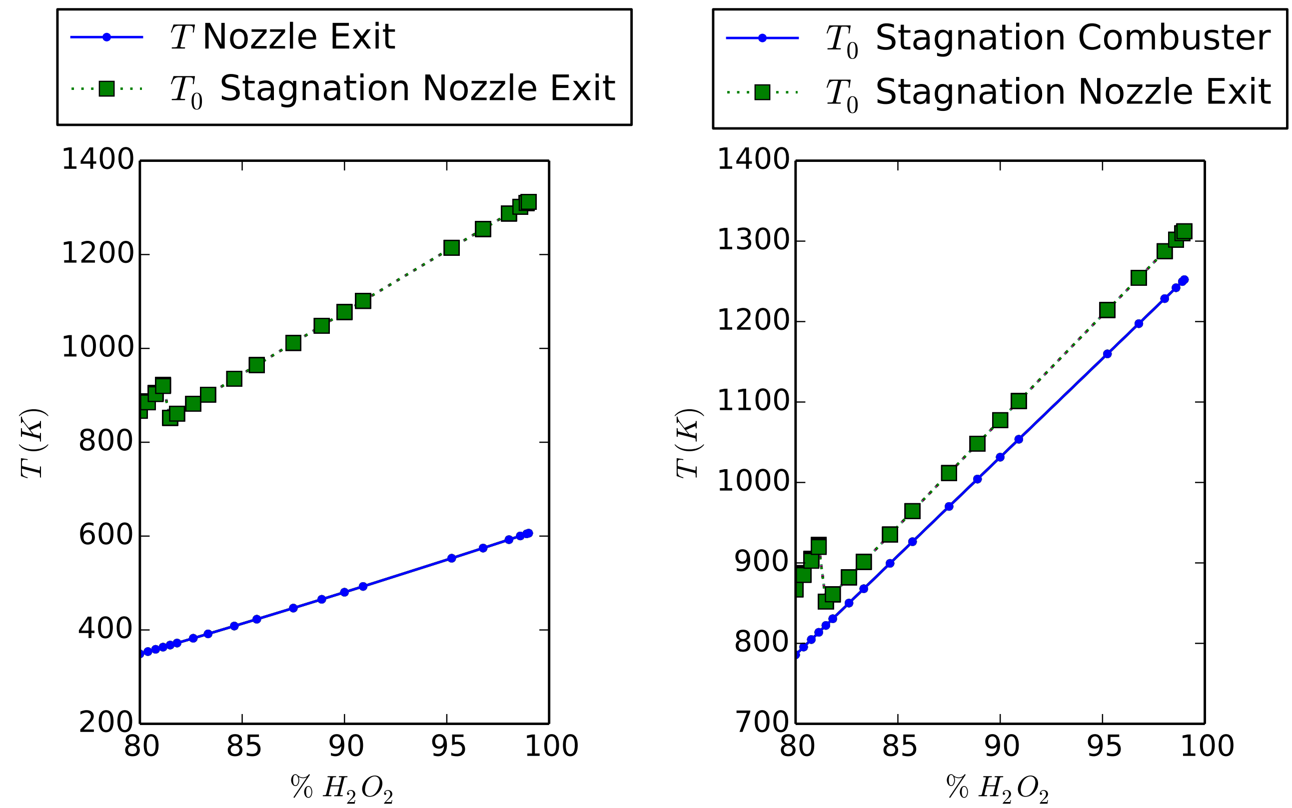

FIGURE 1. Nozzle Exit and Combuster Stagnation Temperatures

Fig. 2 shows the nozzle exit Mach and characteristic velocity and thrust values. Fig. 3 shows the mass flow and I_{sp} values. It is noted that to get the correct specific impulse, we needed to convert from the European I_{sp} values to the American standard in seconds by dividing by gravity at sea level. The values at 9:1 ratio were used in the following comparisons.

Peroxide concentration requires a 5:1 Oxide to fuel ratio to vaporize all water 90% H_2O_2 and optimal pressure gives:

$$
\begin{aligned}
\text{at} &= 0.000807083035601 \\
\text{isp} &= 137.823781358 \\
\text{isp\_opt} &= 137.699990655 \\
\text{mdot} &= 2.28586037785
\end{aligned}
$$

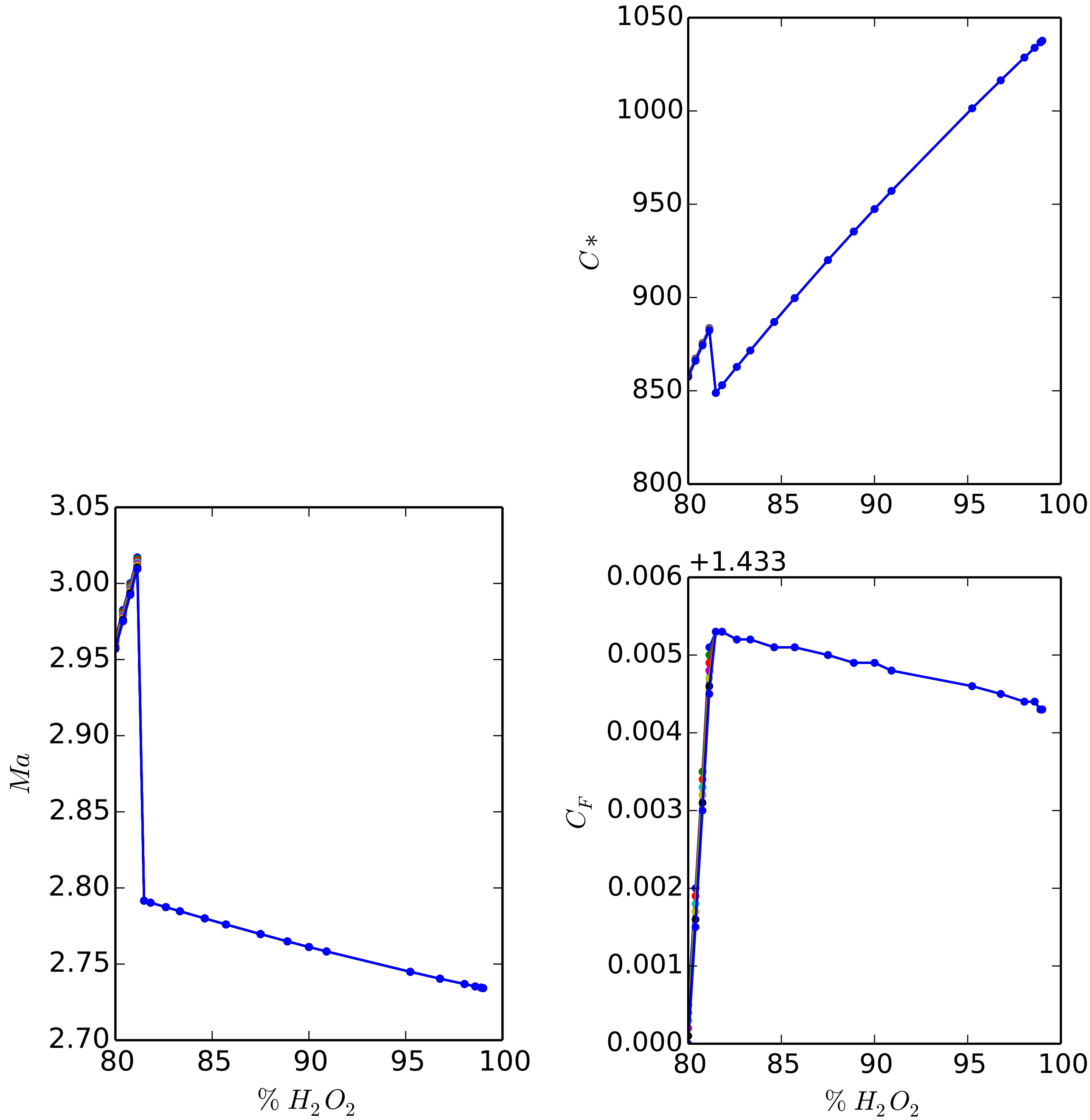

FIGURE 2. Nozzle Exit Mach and C^* and C_F values

2.2 Part 2

In order to calculate the various ABS simulations. A configuration file similar to the one shown in Section A.2 was used. The concentration of hydrogen peroxide and water was varied. Five separate simulations were used where an 80% hydrogen peroxide was increased by 5% in each simulation. The fifth simulation was run at 99%

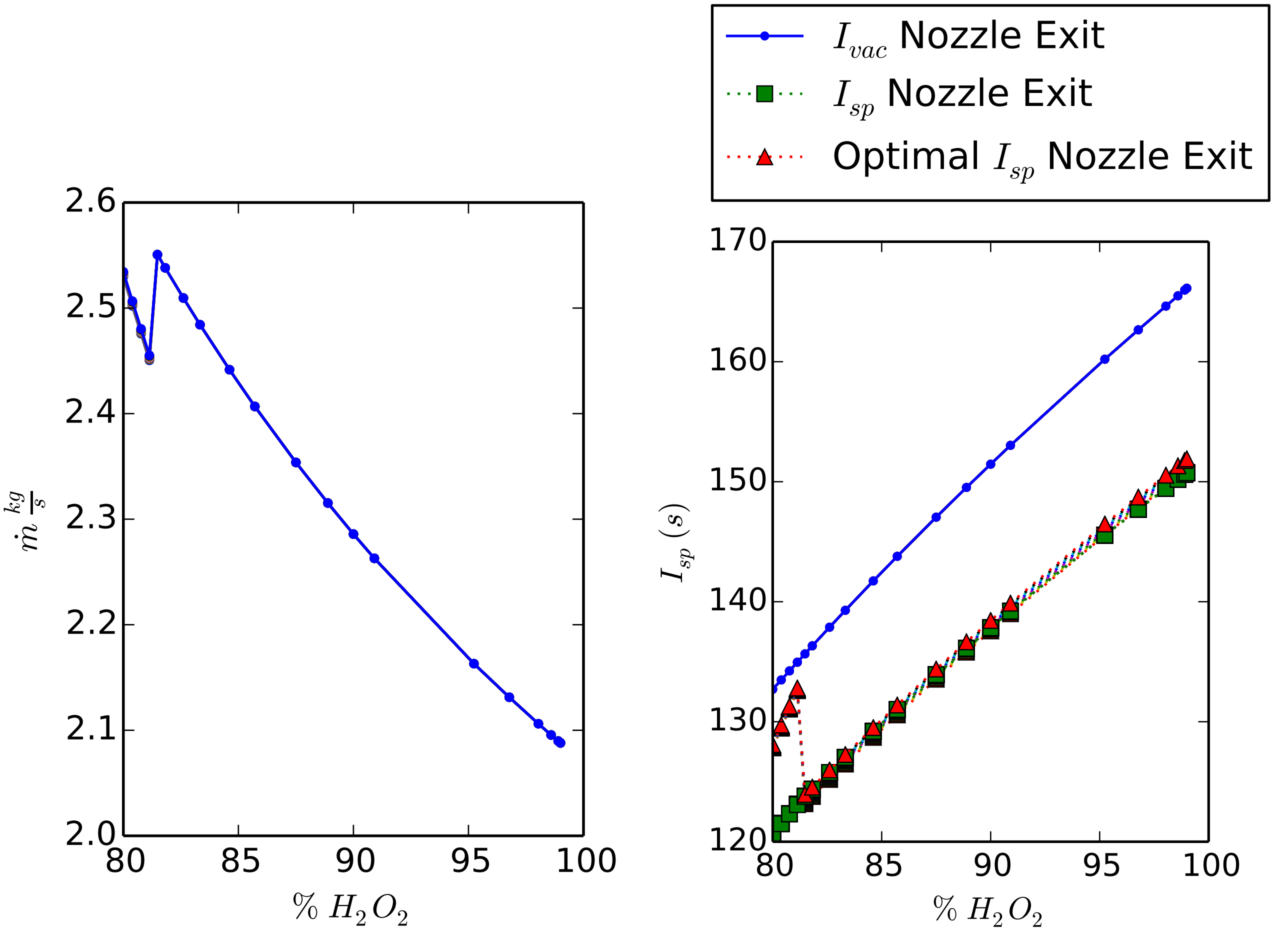

FIGURE 3. I_{sp} values at the nozzle exit

concentration. Each of the simulations varied in the $\frac{O}{F}$ ratio from 50% to 99%. This yielded each simulation had an optimal condition. This optimal condition was set as the largest C^* value. This optimal conditions appropriate output values were then shown in the following figures.

Fig. 4 shows the optimal C^* value as a function of hydrogen peroxide concentration for each of the 5 simulations on the left.

Fig. 4 also shows the optimal values from each of these simulations as a function of hydrogen peroxide concentration on the right.

Fig. 5 shows the optimal conditions $\frac{O}{F}$ ratio.

2.3 Part 3

We desired to see the hover time of each of the rocket types considered and compare their respective abilities. In order to do this, the ratio of I_{sp} values were considered. Eq. 1 was considered and the ratio was shown to relate to the ratio of hover time directly. That is, assuming the vehicle mass fraction was the same. We can see that at all simulation considered, the ABS hybrid rocket out performs the monopropellant rocket. Fig. **??** shows the time

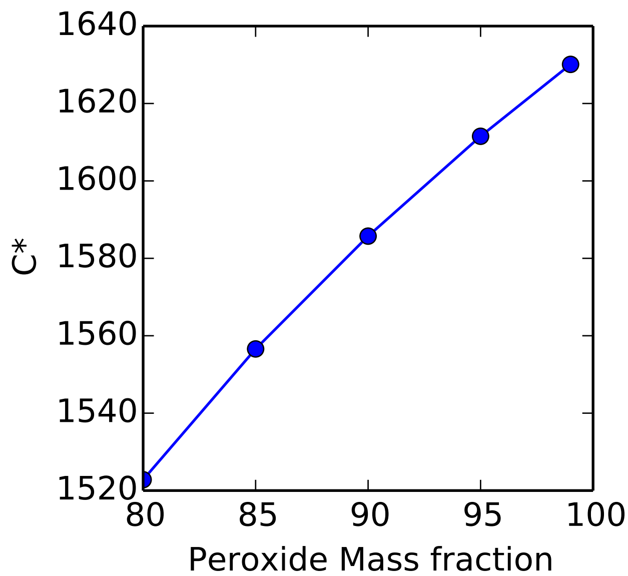

FIGURE 4. C^* and I_{sp} optimal values for each of the 5 simulations

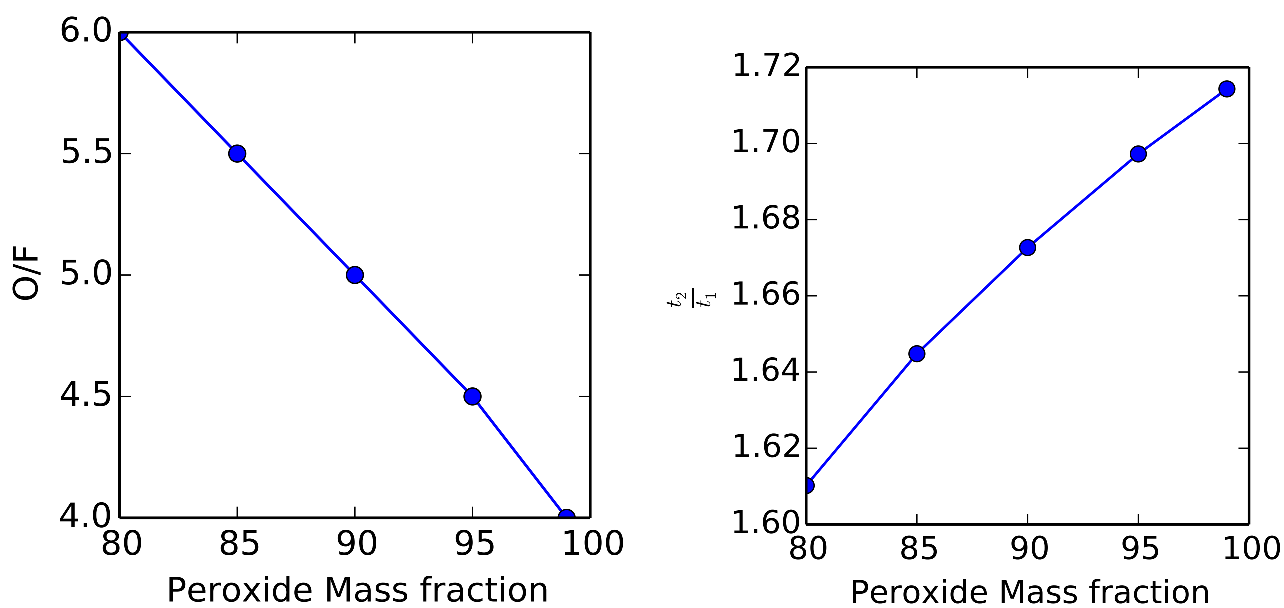

FIGURE 5. (left) The optimal O/F values are compared for each of the 5 simulations (right) shows the time ratio of the hybrid rocket over the 90% monopropellant

ratio.

It is important to note that the currently flight time of the EGC rocket is approximately 35 seconds. Thus, the ratio could be used to calculate the flight time of the ABS hybrid rocket.

3 CONCLUSION

We were able to show the advantage of using the ABS hybrid rocket in the EGC rocket motor. With this in mind, we can see the cost effectiveness of using lower hydrogen peroxide solution to acheive better performance and longer hovering time. If this motor is able to acheive the assumed mass fraction, then this could lead to a market disrupter in rocket engineering.

A Appendix A: Plotting Code

A.1 Part 1 configuration file

```
problem     o/f=4,4.1,4.2,4.3,4.4,4.5,4.75,5,5.5,6,7,8,9,10,20,30,50,70,90,99,
    rocket  equilibrium  frozen  nfz=1  tcest,k=3800
  p,bar=28.0,28.25,28.5,28.75,29.00,29.25,29.50,29.75,
  sup,ae/at=4,
react
  fuel=H2O(L) wt=100  t,k=298
  oxid=H2O2(L) wt=100  t,k=298
output  transport
    plot o/f %f p t rho h g m mw cp gam pip mach aeat cf ivac isp
end
```

A.2 Part 2 configuration file

```
problem     o/f=1,1.5,2,2.5,3,4,4.5,5,5.5,6,7,8,9,10,20,30,50,70,90,99,
    rocket  equilibrium  frozen  nfz=1  tcest,k=3800
  p,bar=24.21,24.22,24.23,24.24,24.25,24.26,24.27,24.28
  sup,ae/at=4,
react
  oxid=H2O2(L) wt=99  t,k=273.15
  oxid=H2O(L) wt=1   t,k=273.15
  fuel=ABS  wt=100  t,k=273.15
    h,kj/mol=62.63  C 3.85 H 4.85 N 0.43
output
    plot o/f %f p t rho h g m mw cp gam pip mach aeat cf ivac isp
end
```

A.3 Plotting python script

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import mlab,cm
from scipy.integrate import quad
import sys
import glob
import os

def get_data(filename):
#     header=np.genfromtxt(filename,dtype=str)[0,:]
    data=np.genfromtxt(filename,skip_header=1)
    return data
def get_data_dtypes(filename):
    data=np.genfromtxt(filename,names=True,dtype=None)
    return data

def calcs(thrust):
    # calculate C*,P0,t0,at,mdot,isp_opt
    for i in range(0,np.size(x1['gam'])):
        g=x1['gam'][i]
```

```
        x1['p'][i]=x1['p'][i]*100000. # convert bar to Pa
        x1['isp'][i]=x1['isp'][i]/9.806 # convert European Isp to American Isp
        x1['ivac'][i]=x1['ivac'][i]/9.806 # convert European Isp to American Isp
        p=x1['p'][i]
        t=x1['t'][i]
        M=x1['mach'][i]
        mw=x1['mw'][i]
        cf=x1['cf'][i]

        t0[i]=t*(1+((g-1.)/2.)*M**2)
        p0[i]=p*(1+((g-1.)/2.)*M**2)**(g/(g-1.))
        #print "P = ",p,p0[i]
        c_star[i]=(np.sqrt(g*8.3144598*1000.)/(g*np.sqrt(2./(g+1))**((g+1.)/(g-1.))) )*np.sqrt(t0[i]/mw)
        if cf!=0: at[i]=thrust/(p0[i]*cf)
        p0_at[i]=p0[i]*at[i]
        if cf!=0: mdot[i]=thrust/(c_star[i]*cf)

        if (x1['pip'][i]==1.7428 and x1['cf'][i]>=0.6613 and x1['cf'][i]<=0.6614 and x1['of'][i]==8.5 and x1['aeat'][i]==1.0 and x1['mach'][i]==1.0 and x1['pip'][i]!=1.0): print 'c_star = ',c_star[i],i

        if (cf!=0): isp_opt[i]=(
                p0_at[i]/(9.806*mdot[i])*(
                    g*np.sqrt(
                        (2./(g-1))*(2./(g+1))**((g+1)/(g-1))
                        )*np.sqrt(
                            1-(p/p0[i])**((g-1)/g))
                    +
                    x1['aeat'][i]*(p-101325.)/p0[i]
                    ))

def big():# plot big plot
    fig=plt.figure(figsize=(24,72))
    n=1
    ax1=fig.add_subplot(np.size(x1.dtype.names)+7,W,1)
    for i in x1.dtype.names:
        for j in range(0,W):
            ax1=fig.add_subplot(np.size(x1.dtype.names)+7,W,n)
            n=n+1
            for k in range(0,W*PS-(W-1),W):
                ax1.plot(100-x1['f'][j+k::W*PS],x1[i][j+k::W*PS],'.-')
            #if (i == 'isp'): ax1.set_xlabel(r'$\%\ H_2O_2$')
            if (j == 0): ax1.set_ylabel(i)

    # Isp optimal
    for j in range(0,W):
        ax1=fig.add_subplot(np.size(x1.dtype.names)+7,W,n)
        n=n+1
        for k in range(0,W*PS-(W-1),W):
            ax1.plot(100-x1['f'][j+k::W*PS],isp_opt[j+k::W*PS],'.-')
        #ax1.set_xlabel(r'$\%\ H_2O_2$')
        if (j == 0): ax1.set_ylabel('Optimal $I_{sp}$')

    # c*
    for j in range(0,W):
        ax1=fig.add_subplot(np.size(x1.dtype.names)+7,W,n)
        n=n+1
        for k in range(0,W*PS-(W-1),W):
            ax1.plot(100-x1['f'][j+k::W*PS],c_star[j+k::W*PS],'.-')
        #ax1.set_xlabel(r'$\%\ H_2O_2$')
        if (j == 0): ax1.set_ylabel('c*')

    # t0
    for j in range(0,W):
        ax1=fig.add_subplot(np.size(x1.dtype.names)+7,W,n)
        n=n+1
```

```
        for k in range(0,W*PS-(W-1),W):
            ax1.plot(100-x1['f'][j+k::W*PS],t0[j+k::W*PS],'.-')
        #ax1.set_xlabel(r'$\%\ H_2O_2$')
        if (j == 0): ax1.set_ylabel('t0')

    # p0
    for j in range(0,W):
        ax1=fig.add_subplot(np.size(x1.dtype.names)+7,W,n)
        n=n+1
        for k in range(0,W*PS-(W-1),W):
            ax1.plot(100-x1['f'][j+k::W*PS],p0[j+k::W*PS],'.-')
        #ax1.set_xlabel(r'$\%\ H_2O_2$')
        if (j == 0): ax1.set_ylabel('p0')

    # a_throat
    for j in range(0,W):
        ax1=fig.add_subplot(np.size(x1.dtype.names)+7,W,n)
        n=n+1
        for k in range(0,W*PS-(W-1),W):
            ax1.plot(100-x1['f'][j+k::W*PS],at[j+k::W*PS],'.-')
        #ax1.set_xlabel(r'$\%\ H_2O_2$')
        if (j == 0): ax1.set_ylabel('at')

    # P0*a_throat
    for j in range(0,W):
        ax1=fig.add_subplot(np.size(x1.dtype.names)+7,W,n)
        n=n+1
        for k in range(0,W*PS-(W-1),W):
            ax1.plot(100-x1['f'][j+k::W*PS],p0_at[j+k::W*PS],'.-')
        #ax1.set_xlabel(r'$\%\ H_2O_2$')
        if (j == 0): ax1.set_ylabel('p0 at')

    # m_dot
    for j in range(0,W):
        ax1=fig.add_subplot(np.size(x1.dtype.names)+7,W,n)
        n=n+1
        for k in range(0,W*PS-(W-1),W):
            ax1.plot(100-x1['f'][j+k::W*PS],mdot[j+k::W*PS],'.-')
        ax1.set_xlabel(r'$\%\ H_2O_2$')
        if (j == 0): ax1.set_ylabel(r'$\dot{m}$')

    fig.savefig('all.pdf',bbox_inches='tight')

def Part1_plots(): # Part 1 specific plots desired i
    fig=plt.figure(figsize=(3,4))
    ax1=fig.add_subplot(111)
    for k in range(0,W*PS-(W-1),W):
        ax1.plot(100-x1['f'][W-1+k::W*PS],x1['t'][W-1+k::W*PS],'.-',label='$T$ Nozzle Exit' if k==0 else "")
        ax1.plot(100-x1['f'][W-1+k::W*PS],t0[W-1+k::W*PS],'s:',label='$T_0$ Stagnation Nozzle Exit' if k==0 else "")
    ax1.set_xlabel(r'$\%\ H_2O_2$')
    ax1.set_ylabel(r'$T\ (K)$')
    ax1.legend(numpoints=1,loc='upper center',bbox_to_anchor=(0.5,1.3))
    fig.savefig('Part1_i.pdf',bbox_inches='tight')

    # Part 1 specific plots desired ii
    fig=plt.figure(figsize=(3,4))
    ax1=fig.add_subplot(111)
    for k in range(0,W*PS-(W-1),W):
        ax1.plot(100-x1['f'][W-2+k::W*PS],t0[W-2+k::W*PS],'.-',label='$T_0$ Stagnation Combuster' if k==0 else "")
        ax1.plot(100-x1['f'][W-1+k::W*PS],t0[W-1+k::W*PS],'s:',label='$T_0$ Stagnation Nozzle Exit' if k==0 else "")
    ax1.set_xlabel(r'$\%\ H_2O_2$')
    ax1.set_ylabel(r'$T\ (K)$')
```

```
    ax1.legend(numpoints=1,loc='upper center',bbox_to_anchor=(0.5,1.3))
    fig.savefig('Part1_ii.pdf',bbox_inches='tight')

    # Part 1 specific plots desired iii
    fig=plt.figure(figsize=(3,4))
    ax1=fig.add_subplot(111)
    for k in range(0,W*PS-(W-1),W):
        ax1.plot(100-x1['f'][W-1+k::W*PS],x1['mach'][W-1+k::W*PS],'.-',label='$Ma$ Nozzle Exit' if k==0 else "")
    ax1.set_xlabel(r'$\%\ H_2O_2$')
    ax1.set_ylabel(r'$Ma$')
    #ax1.legend(numpoints=1,loc='upper center',bbox_to_anchor=(0.5,1.2))
    fig.savefig('Part1_iii.pdf',bbox_inches='tight')

    # Part 1 specific plots desired iv
    fig=plt.figure(figsize=(3,8))
    ax1=fig.add_subplot(211)
    for k in range(0,W*PS-(W-1),W):
        ax1.plot(100-x1['f'][W-1+k::W*PS],c_star[W-1+k::W*PS],'.-',label='$C*$ Nozzle Exit' if k==0 else "")
    ax1.set_ylabel(r'$C*$')
    #ax1.legend(numpoints=1,loc='upper center')

    ax1=fig.add_subplot(212)
    for k in range(0,W*PS-(W-1),W):
        ax1.plot(100-x1['f'][W-1+k::W*PS],x1['cf'][W-1+k::W*PS],'.-',label='$C_F$ Nozzle Exit' if k==0 else "")
    ax1.set_xlabel(r'$\%\ H_2O_2$')
    ax1.set_ylabel(r'$C_F$')
    #ax1.legend(numpoints=1,loc='upper center')
    fig.savefig('Part1_iv.pdf',bbox_inches='tight')

    # Part 1 specific plots desired v
    fig=plt.figure(figsize=(3,4))
    ax1=fig.add_subplot(111)
    for k in range(0,W*PS-(W-1),W):
        ax1.plot(100-x1['f'][W-1+k::W*PS],x1['ivac'][W-1+k::W*PS],'.-',label='$I_{vac}$ Nozzle Exit' if k==0 else "")
        ax1.plot(100-x1['f'][W-1+k::W*PS],x1['isp'][W-1+k::W*PS],'s:',label='$I_{sp}$ Nozzle Exit' if k==0 else "")
        ax1.plot(100-x1['f'][W-1+k::W*PS],isp_opt[W-1+k::W*PS],'^:',label='Optimal $I_{sp}$ Nozzle Exit' if k==0 else "")
    ax1.set_xlabel(r'$\%\ H_2O_2$')
    ax1.set_ylabel(r'$I_{sp}\ (s)$')
    ax1.legend(numpoints=1,loc='upper center',bbox_to_anchor=(0.5,1.425))
    fig.savefig('Part1_v.pdf',bbox_inches='tight')

    # Part 1 specific peroxide concentration required?
    print "Peroxide concentration requires a 5:1 Oxide to fuel ratio to vaporize all water"

    # Part 1 What at is required for the optimal operating chamber pressure at 90% H2O2 concentration?, corresponding Isp
    #print 0,W*PS-(W-1),W
    #print np.size(at[2+0::W*PS]),W*PS
    #print "A* and optimal Isp at 90% H2O2 for 3114N thrust"
    for k in range(0,W*PS-(W-1),W):
        for i in range(W-1+k,np.size(at),W*PS):
            #print i,k,x1['f'][i],at[i]
            if (x1['f'][i]==10 and x1['p'][i]==100440.):
                print "90% H2O2 and optimal pressure gives:"
                print "  at =        ",at[i]
                print "  isp=        ",x1['isp'][i]
                print "  isp_opt=    ",isp_opt[i]
                print "  mdot=       ",mdot[i]

    # Part 1 specific plots desired vi
    fig=plt.figure(figsize=(3,4))
    ax1=fig.add_subplot(111)
```

```python
    for k in range(0,W*PS-(W-1),W):
        ax1.plot(100-x1['f'][W-1+k::W*PS],mdot[W-1+k::W*PS],'.-')
    ax1.set_xlabel(r'$\%\ H_2O_2$')
    ax1.set_ylabel(r'$\dot{m}\ \frac{kg}{s}$')
    fig.savefig('Part1_vi.pdf',bbox_inches='tight')

def output_file():# output file
    f=open('output.txt','w')
    # headings
    f.write('# ')
    for t in x1.dtype.names:
        f.write(t+' ')
    f.write('isp_opt ')
    f.write('t0 ')
    f.write('p0 ')
    f.write('c_star ')
    f.write('at ')
    f.write('p0at ')
    f.write('m_dot \n')
    #for i in x1.dtype.names:
        #for j in np.size(x1[i]):
    for i in range(0,np.size(x1['gam'])):
        for t in x1.dtype.names:
            f.write(str(x1[t][i]))
            f.write(' ')
        f.write(str(isp_opt[i]))
        f.write(' ')
        f.write(str(t0[i]))
        f.write(' ')
        f.write(str(p0[i]))
        f.write(' ')
        f.write(str(c_star[i]))
        f.write(' ')
        f.write(str(at[i]))
        f.write(' ')
        f.write(str(p0_at[i]))
        f.write(' ')
        f.write(str(mdot[i]))
        f.write(' \n')
    f.close()

def plot_data(xlabel,x,ylabel,y,filename):
    fig=plt.figure(figsize=(3,3))
    ax1=fig.add_subplot(111)
    ax1.plot(x,y,'o')
    ax1.set_xlabel(xlabel)
    ax1.set_ylabel(ylabel)
    #ax1.axis('equal')
    fig.savefig(filename,bbox_inches='tight')

def example(): # example from section 7.1
    global PS
    global W
    PS=5
    W=2
    # get argument list using sys module
    global x1
    x1=get_data_dtypes(filename1)

    # subroutines for part 1
    # initialize values
    global c_star
    global p0
    global p0_at
    global t0
```

```
    global at
    global mdot
    global isp_opt

    c_star  = np.zeros(np.size(x1['gam']))
    p0      = np.zeros(np.size(x1['gam']))
    p0_at   = np.zeros(np.size(x1['gam']))
    t0      = np.zeros(np.size(x1['gam']))
    at      = np.zeros(np.size(x1['gam']))
    mdot    = np.zeros(np.size(x1['gam']))
    isp_opt = np.zeros(np.size(x1['gam']))

    calcs(110.)
    big()
    #Part1_plots()
    #output_file()


def Part1(): #  Part 1
    global PS
    global W
    PS=8
    W=3
    # get argument list using sys module
    global x1
    x1=get_data_dtypes(filename1)

    # subroutines for part 1
    # initialize values
    global c_star
    global p0
    global p0_at
    global t0
    global at
    global mdot
    global isp_opt

    c_star  = np.zeros(np.size(x1['gam']))
    p0      = np.zeros(np.size(x1['gam']))
    p0_at   = np.zeros(np.size(x1['gam']))
    t0      = np.zeros(np.size(x1['gam']))
    at      = np.zeros(np.size(x1['gam']))
    mdot    = np.zeros(np.size(x1['gam']))
    isp_opt = np.zeros(np.size(x1['gam']))

    calcs(3114.)
    big()
    Part1_plots()
    output_file()

def Part2_iter(): #  Part 2
    global PS
    global W
    PS=8
    W=3
    # get argument list using sys module
    global x1
    x1=get_data_dtypes(filename1)

    # subroutines for part 1
    # initialize values
    global c_star
    global p0
    global p0_at
    global t0
```

```
    global at
    global mdot
    global isp_opt

    c_star  = np.zeros(np.size(x1['gam']))
    p0      = np.zeros(np.size(x1['gam']))
    p0_at   = np.zeros(np.size(x1['gam']))
    t0      = np.zeros(np.size(x1['gam']))
    at      = np.zeros(np.size(x1['gam']))
    mdot    = np.zeros(np.size(x1['gam']))
    isp_opt = np.zeros(np.size(x1['gam']))

    calcs(3114.)
    # Part 2 specific plots for iteratin
    fig=plt.figure(figsize=(3,4))
    ax1=fig.add_subplot(111)
    for k in range(0,W*PS-(W-1),W):
        ax1.plot(100-x1['f'][W-1+k::W*PS],x1['p'][W-1+k::W*PS],'.-',label=k)
    ax1.set_xlabel(r'$\%\ ABS$')
    ax1.set_ylabel(r'$P\ (Pa)$')
    ax1.legend(numpoints=1,loc='best')
    fig.savefig('Part2_p.pdf',bbox_inches='tight')
    output_file()

def Part2(): #  Part 2, combined files
    # get all txt files
    W=3
    PS=8
    s= np.sort(glob.glob("./Part2/*/*.txt"))
    max_values=np.zeros(np.size(s))
    loc_of=np.zeros(np.size(s))
    loc_isp_opt=np.zeros(np.size(s))
    #loc_isp=np.zeros(np.size(s))
    fig = plt.figure(figsize=(3,3))
    fig2 = plt.figure(figsize=(3,3))
    ax1=fig.add_subplot(111)
    ax2=fig2.add_subplot(111)
    for f in range(0,np.size(s)):
        filename=s[f]
        x_data=get_data_dtypes(filename)
        max_values[f] = np.max(x_data['c_star'][2::3])
        for i in range(0,np.size(x_data['c_star'])):
            if (x_data['c_star'][i] == max_values[f] and x_data['isp'][i] !=0):
                loc_of[f]=x_data['of'][i]
                loc_isp_opt[f]=x_data['isp_opt'][i]
        for k in range(0,W*PS-(W-1),W):
            ax1.plot(100-x_data['f'][W-1+k::W*PS],x_data['isp_opt'][W-1+k::W*PS],'.-',label=filename if k==0
                else '')
            ax2.plot(100-x_data['f'][W-1+k::W*PS],x_data['c_star'][W-1+k::W*PS],'.-',label=filename if k==0 else
                 '')
    ax1.set_xlabel('O/F')
    ax2.set_xlabel('O/F')
    ax1.set_ylabel('$I_{sp}$')
    ax2.set_ylabel('$C*$')
    ax1.legend(numpoints=1,loc='best')
    ax2.legend(numpoints=1,loc='best')
    fig.savefig('Part2_Isps.pdf',bbox_inches='tight')
    fig2.savefig('Part2_C_stars.pdf',bbox_inches='tight')
    # c*
    fig = plt.figure(figsize=(3,3))
    ax1=fig.add_subplot(111)
    ax1.plot(np.array([80,85,90,95,99]),max_values,'o-')
    ax1.set_xlabel('Peroxide Mass fraction')
    ax1.set_ylabel('C*')
    fig.savefig('Part2_C_star.pdf',bbox_inches='tight')
```

```
    # O/F
    fig = plt.figure(figsize=(3,3))
    ax1=fig.add_subplot(111)
    ax1.plot(np.array([80,85,90,95,99]),loc_of,'o-')
    ax1.set_xlabel('Peroxide Mass fraction')
    ax1.set_ylabel('O/F')
    fig.savefig('Part2_OF.pdf',bbox_inches='tight')

    # Isp
    fig = plt.figure(figsize=(3,3))
    ax1=fig.add_subplot(111)
    ax1.plot(np.array([80,85,90,95,99]),loc_isp_opt[:],'o-')
    #ax1.plot(np.array([80,85,90,95]),loc_isp[:-1:],'o-')
    ax1.set_xlabel('Peroxide Mass fraction')
    ax1.set_ylabel('Optimal $I_{sp}$')
    fig.savefig('Part2_OptimalIsp.pdf',bbox_inches='tight')

def Part3(): #  Part 2, combined files
    # get all txt files
    s= np.sort(glob.glob("./Part2/*/*.txt"))
    max_values=np.zeros(np.size(s))
    loc_of=np.zeros(np.size(s))
    loc_isp_opt=np.zeros(np.size(s))
    loc_isp=np.zeros(np.size(s))
    for f in range(0,np.size(s)):
        filename=s[f]
        x_data=get_data_dtypes(filename)
        max_values[f] = np.max(x_data['c_star'][2::3])
        for i in range(0,np.size(x_data['c_star'])):
            if (x_data['c_star'][i] == max_values[f] and x_data['isp'][i]!=0):
                loc_of[f]=x_data['of'][i]
                loc_isp_opt[f]=x_data['isp_opt'][i]
                loc_isp[f]=x_data['isp'][i]

    # ratio of Isp/Isp (hybrid/monopropollent)
    fig = plt.figure(figsize=(3,3))
    ax1=fig.add_subplot(111)
    ax1.plot(np.array([80,85,90,95,99]),(loc_isp_opt[:])/137.69999,'o-')
    ax1.set_xlabel('Peroxide Mass fraction')
    ax1.set_ylabel(r'$\frac{t_2}{t_1}$')
    fig.savefig('Part3.pdf',bbox_inches='tight')




def main():
    #user defined values
    #sys.argv
    global filename1
    filename1 = str(sys.argv[1])

    # example stuff
    #example()

    # Part 1 stuff
    #Part1()

    # part 2 stuff
    #Part2_iter()
    Part2()

    # part 3 stuff
    #Part3()

```

```python
if __name__ == '__main__':
    main()
```