

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

1 INTRODUCTION	3
1.1 Part 1	3
1.2 Part 2	3
1.3 Part 3	3
2 RESULTS	3
2.1 Part 1	3
2.2 Part 2	5
2.3 Part 3	6
3 CONCLUSION	8
A Appendix A: Plotting Code	8
A.1 Part 1 configuration file	8
A.2 Part 2 configuration file	8
A.3 Plotting python script	8

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 be 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 expansion. 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 separately calculated isentropic stagnation temperature. The stagnation temperatures were calculated, and the nozzle exit and combustor 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 then 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.

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

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 com-

buster and nozzle exit stagnation temperatures. These stagnation temperatures were calculated using isentropic relations.

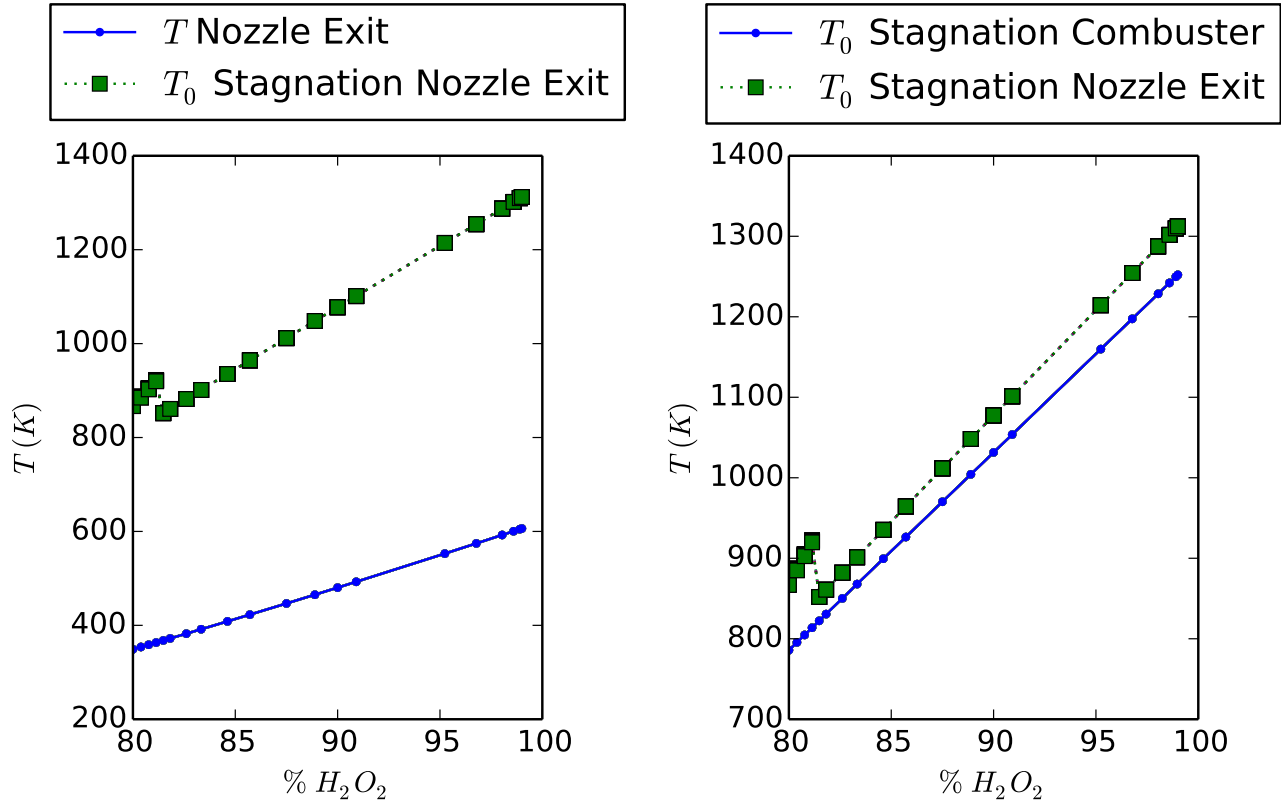


FIGURE 1. Nozzle Exit and Combuster Stagnation Temperatures

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:

at = 0.000807083035601
 isp = 137.823781358
 isp_opt = 137.699990655
 mdot = 2.28586037785

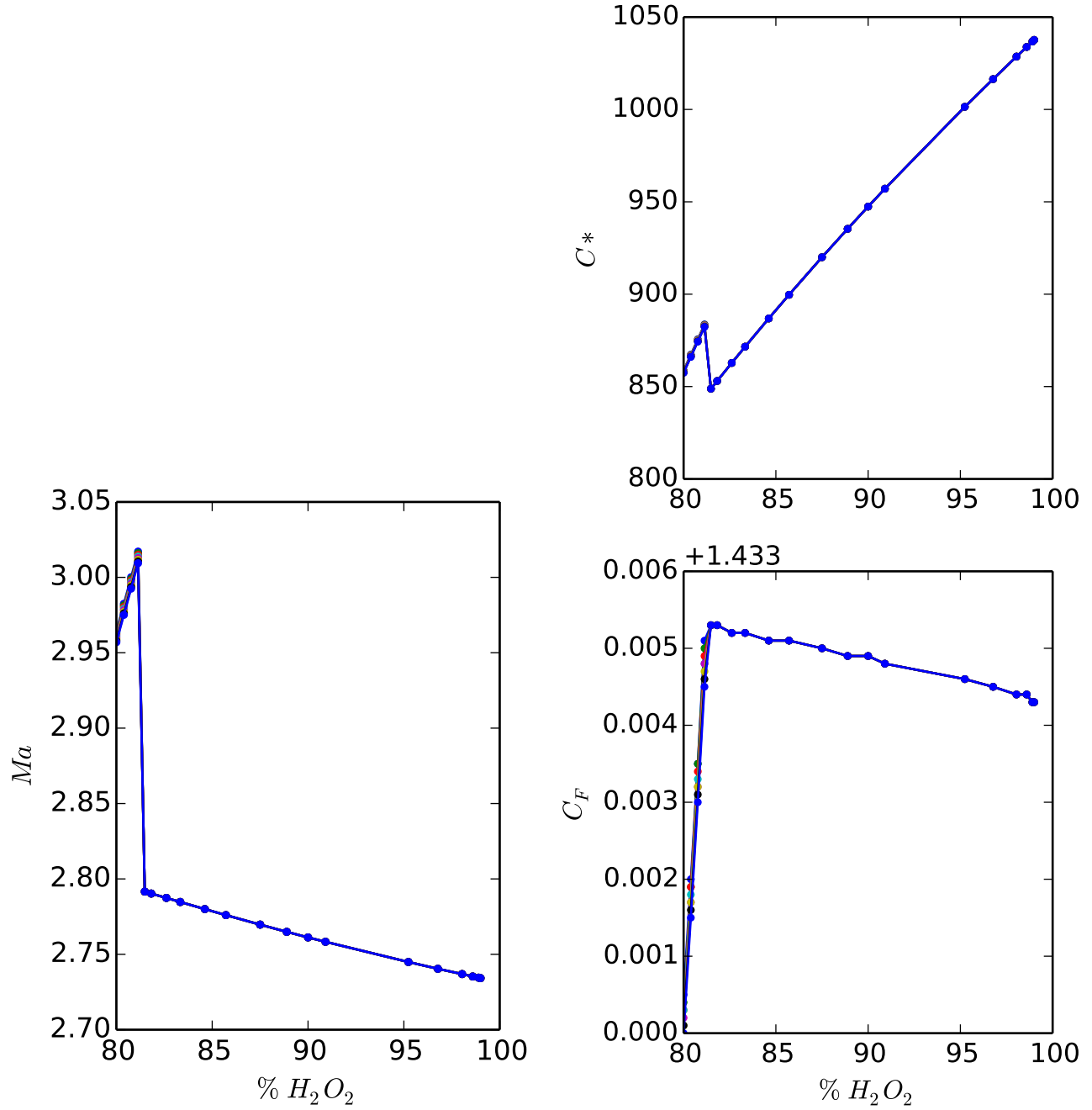


FIGURE 2. Nozzle Exit Mach and C^* and C_F values

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%

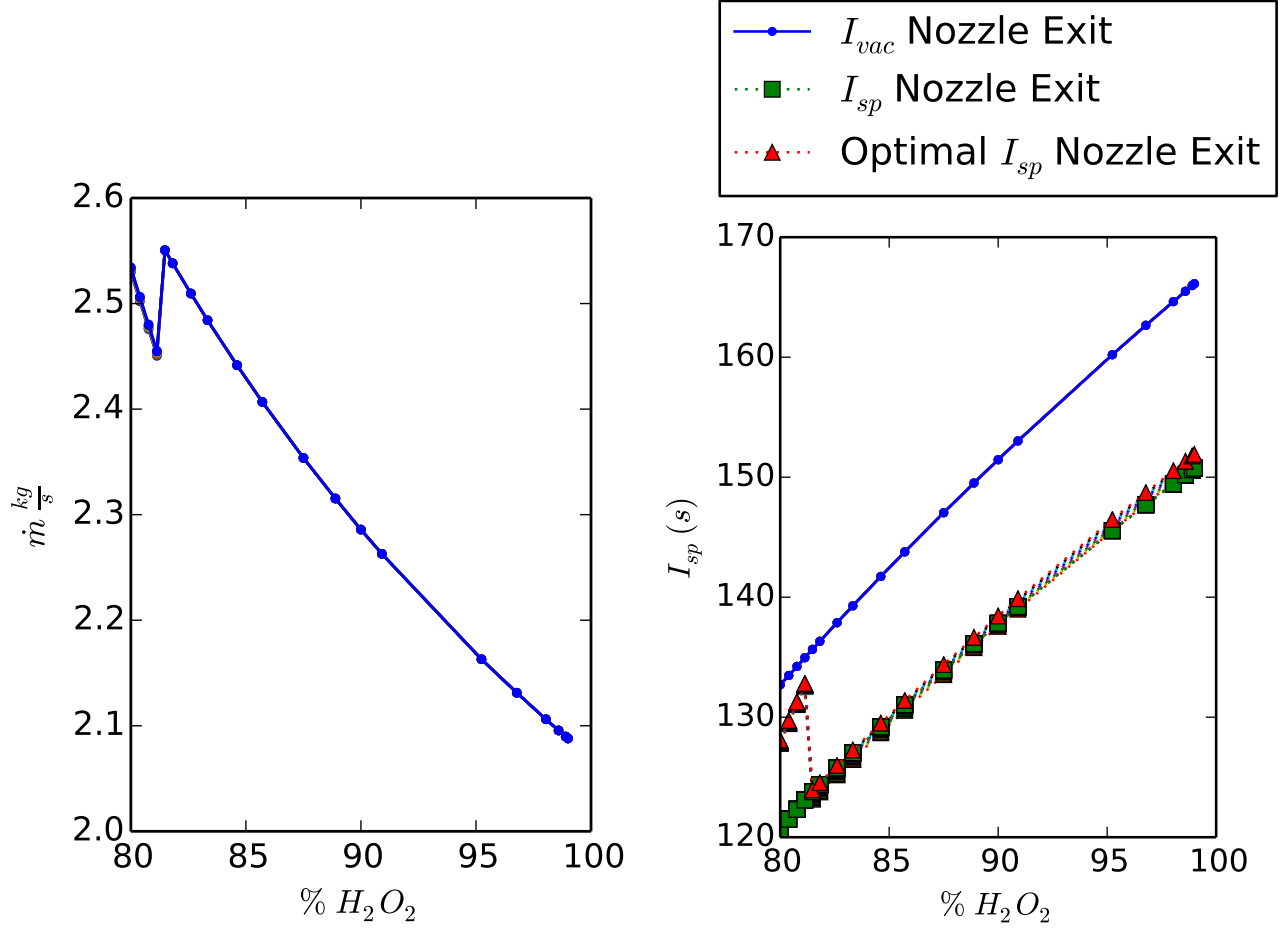


FIGURE 3. I_{sp} values at the nozzle exit

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

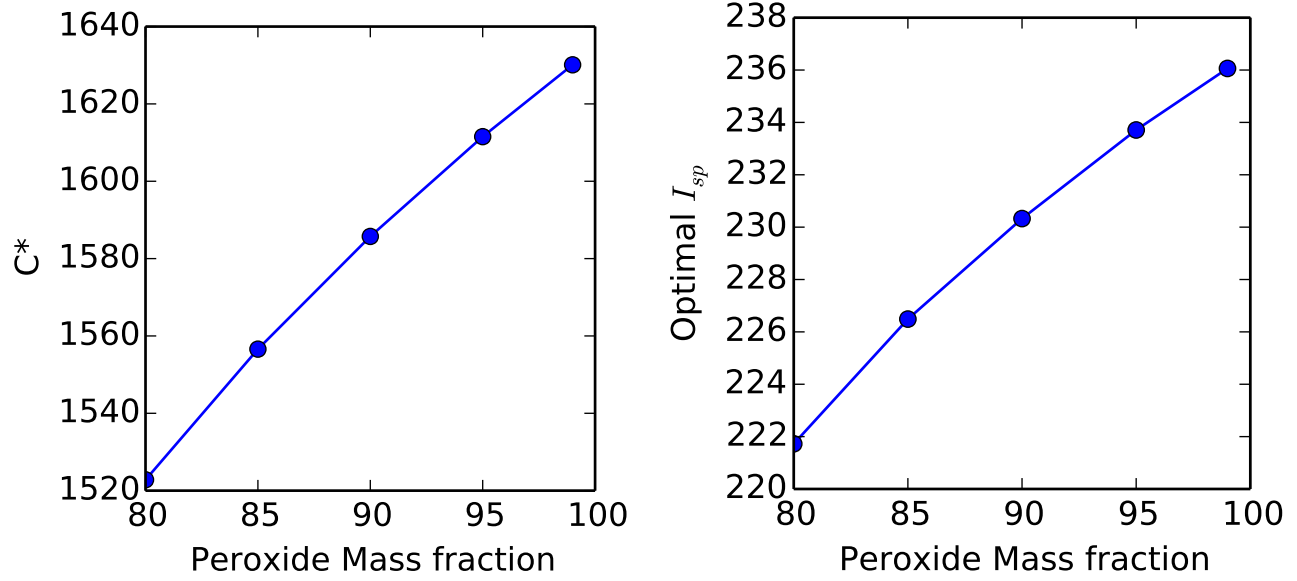


FIGURE 4. C^* and I_{sp} optimal values for each of the 5 simulations

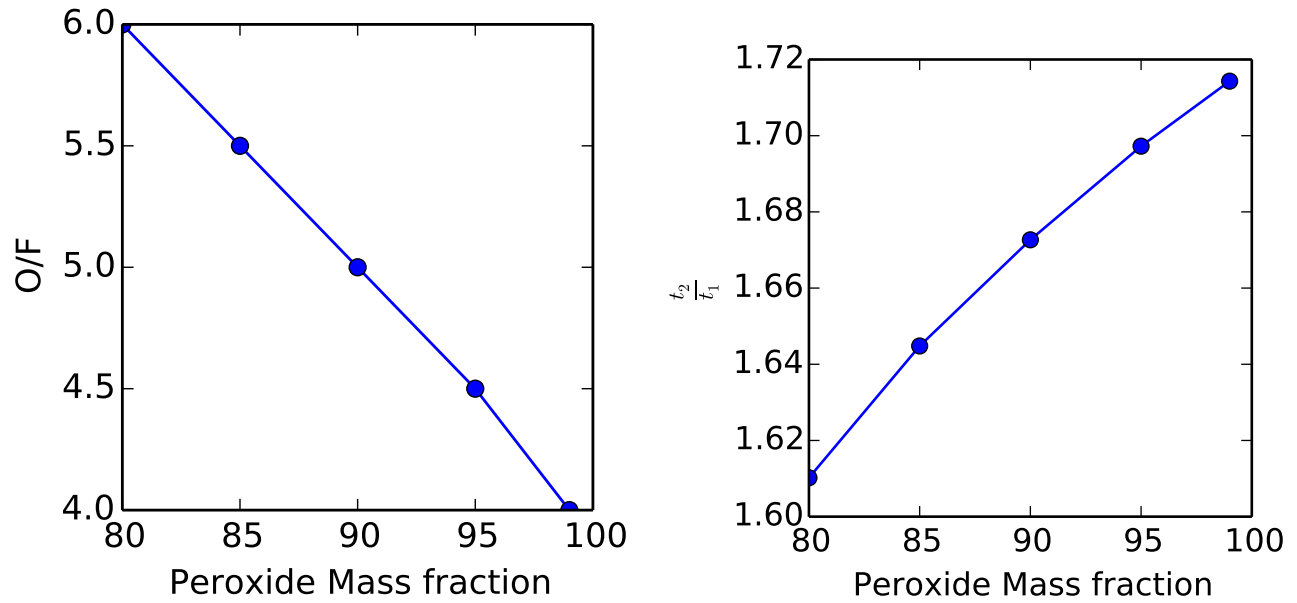


FIGURE 5. (left) The optimal O/F values are compared for each of the 5 simulations (right) shows the time ratio of the hybrid rocket over the 90% monopropellant

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 achieve better performance and longer hovering time. If this motor is able to achieve 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

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

A.2 Part 2 configuration file

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

A.3 Plotting python script

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from matplotlib import mlab,cm
4 from scipy.integrate import quad
5 import sys
6 import glob
7 import os
8
9 def get_data(filename):
10 #    header=np.genfromtxt(filename,dtype=str)[0,:]
11    data=np.genfromtxt(filename,skip_header=1)
12    return data
13 def get_data_dtypes(filename):
14    data=np.genfromtxt(filename,names=True,dtype=None)
15    return data
16
17 def calcs(thrust):
18 #    calculate C*,P0,t0,at,mdot,isp_opt
19    for i in range(0,np.size(x1['gam'])):
20        g=x1['gam'][i]
```



```

21 x1['p'][i]=x1['p'][i]*100000. # convert bar to Pa
22 x1['isp'][i]=x1['isp'][i]/9.806 # convert European Isp to American Isp
23 x1['ivac'][i]=x1['ivac'][i]/9.806 # convert European Isp to American Isp
24 p=x1['p'][i]
25 t=x1['t'][i]
26 M=x1['mach'][i]
27 mw=x1['mw'][i]
28 cf=x1['cf'][i]
29
30 t0[i]=t*(1+((g-1.)/2.)*M**2)
31 p0[i]=p*(1+((g-1.)/2.)*M**2)*(g/(g-1.))
32 #print "P = ",p,p0[i]
33 c_star[i]=(np.sqrt(g*8.3144598*1000.)/(g*np.sqrt(2./(g+1))*((g+1)/(g-1))))*np.sqrt(t0[i]/mw)
34 if cf!=0: at[i]=thrust/(p0[i]*cf)
35 p0_at[i]=p0[i]*at[i]
36 if cf!=0: mdot[i]=thrust/(c_star[i]*cf)
37
38 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
39
40 if (cf!=0): isp_opt[i]=(
41     p0_at[i]/(9.806*mdot[i])*(
42         g*np.sqrt(
43             (2./(g-1))*(2./(g+1))*((g+1)/(g-1))
44         )*np.sqrt(
45             1-(p/p0[i])*((g-1)/g))
46         +
47         x1['aeat'][i]*(p-101325.)/p0[i]
48     ))
49
50 def big():# plot big plot
51 fig=plt.figure(figsize=(24,72))
52 n=1
53 ax1=fig.add_subplot(np.size(x1.dtype.names)+7,W,1)
54 for i in x1.dtype.names:
55     for j in range(0,W):
56         ax1=fig.add_subplot(np.size(x1.dtype.names)+7,W,n)
57         n=n+1
58         for k in range(0,W*PS-(W-1),W):
59             ax1.plot(100-x1['f'][j+k::W*PS],x1[i][j+k::W*PS],'.-')
60             #if (i == 'isp'): ax1.set_xlabel(r'$\% H_{2O_2}$')
61             if (j == 0): ax1.set_ylabel(i)
62
63 # Isp optimal
64 for j in range(0,W):
65     ax1=fig.add_subplot(np.size(x1.dtype.names)+7,W,n)
66     n=n+1
67     for k in range(0,W*PS-(W-1),W):
68         ax1.plot(100-x1['f'][j+k::W*PS],isp_opt[j+k::W*PS],'.-')
69     #ax1.set_xlabel(r'$\% H_{2O_2}$')
70     if (j == 0): ax1.set_ylabel('Optimal $I_{sp}$')
71
72 # c*
73 for j in range(0,W):
74     ax1=fig.add_subplot(np.size(x1.dtype.names)+7,W,n)
75     n=n+1
76     for k in range(0,W*PS-(W-1),W):
77         ax1.plot(100-x1['f'][j+k::W*PS],c_star[j+k::W*PS],'.-')
78     #ax1.set_xlabel(r'$\% H_{2O_2}$')
79     if (j == 0): ax1.set_ylabel('c*')
80
81 # t0
82 for j in range(0,W):
83     ax1=fig.add_subplot(np.size(x1.dtype.names)+7,W,n)
84     n=n+1

```

```

85     for k in range(0,W*PS-(W-1),W):
86         ax1.plot(100-x1['f'][j+k::W*PS],t0[j+k::W*PS],'.-')
87     #ax1.set_xlabel(r'$\% \ H_2O_2$')
88     if (j == 0): ax1.set_ylabel('t0')
89
90 # p0
91 for j in range(0,W):
92     ax1=fig.add_subplot(np.size(x1.dtype.names)+7,W,n)
93     n=n+1
94     for k in range(0,W*PS-(W-1),W):
95         ax1.plot(100-x1['f'][j+k::W*PS],p0[j+k::W*PS],'.-')
96     #ax1.set_xlabel(r'$\% \ H_2O_2$')
97     if (j == 0): ax1.set_ylabel('p0')
98
99 # a_throat
100 for j in range(0,W):
101     ax1=fig.add_subplot(np.size(x1.dtype.names)+7,W,n)
102     n=n+1
103     for k in range(0,W*PS-(W-1),W):
104         ax1.plot(100-x1['f'][j+k::W*PS],at[j+k::W*PS],'.-')
105     #ax1.set_xlabel(r'$\% \ H_2O_2$')
106     if (j == 0): ax1.set_ylabel('at')
107
108 # P0*a_throat
109 for j in range(0,W):
110     ax1=fig.add_subplot(np.size(x1.dtype.names)+7,W,n)
111     n=n+1
112     for k in range(0,W*PS-(W-1),W):
113         ax1.plot(100-x1['f'][j+k::W*PS],p0_at[j+k::W*PS],'.-')
114     #ax1.set_xlabel(r'$\% \ H_2O_2$')
115     if (j == 0): ax1.set_ylabel('p0 at')
116
117 # m_dot
118 for j in range(0,W):
119     ax1=fig.add_subplot(np.size(x1.dtype.names)+7,W,n)
120     n=n+1
121     for k in range(0,W*PS-(W-1),W):
122         ax1.plot(100-x1['f'][j+k::W*PS],mdot[j+k::W*PS],'.-')
123     ax1.set_xlabel(r'$\% \ H_2O_2$')
124     if (j == 0): ax1.set_ylabel(r'$\dot{m}$')
125
126 fig.savefig('all.pdf',bbox_inches='tight')
127
128 def Part1_plots(): # Part 1 specific plots desired i
129     fig=plt.figure(figsize=(3,4))
130     ax1=fig.add_subplot(111)
131     for k in range(0,W*PS-(W-1),W):
132         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 '')
133         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
134             '')
135     ax1.set_xlabel(r'$\% \ H_2O_2$')
136     ax1.set_ylabel(r'$T$ (K)$')
137     ax1.legend(numpoints=1,loc='upper center',bbox_to_anchor=(0.5,1.3))
138     fig.savefig('Part1_i.pdf',bbox_inches='tight')
139
140 # Part 1 specific plots desired ii
141     fig=plt.figure(figsize=(3,4))
142     ax1=fig.add_subplot(111)
143     for k in range(0,W*PS-(W-1),W):
144         ax1.plot(100-x1['f'][W-2+k::W*PS],t0[W-2+k::W*PS],'.-',label='$T_0$ Stagnation Combuster' if k==0 else '')
145         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
146             '')
147     ax1.set_xlabel(r'$\% \ H_2O_2$')
148     ax1.set_ylabel(r'$T$ (K)$')

```

```

147 ax1.legend(numpoints=1,loc='upper center',bbox_to_anchor=(0.5,1.3))
148 fig.savefig('Part1.ii.pdf',bbox_inches='tight')
149
150 # Part 1 specific plots desired iii
151 fig=plt.figure(figsize=(3,4))
152 ax1=fig.add_subplot(111)
153 for k in range(0,W*PS-(W-1),W):
154     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 "")
155 ax1.set_xlabel(r'$\%$ H2O2$')
156 ax1.set_ylabel(r'$Ma$')
157 #ax1.legend(numpoints=1,loc='upper center',bbox_to_anchor=(0.5,1.2))
158 fig.savefig('Part1.iii.pdf',bbox_inches='tight')
159
160 # Part 1 specific plots desired iv
161 fig=plt.figure(figsize=(3,8))
162 ax1=fig.add_subplot(211)
163 for k in range(0,W*PS-(W-1),W):
164     ax1.plot(100-x1['f'][W-1+k::W*PS],c_star[W-1+k::W*PS],'.- ',label='$C*$ Nozzle Exit' if k==0 else "")
165 ax1.set_ylabel(r'$C*$')
166 #ax1.legend(numpoints=1,loc='upper center')
167
168 ax1=fig.add_subplot(212)
169 for k in range(0,W*PS-(W-1),W):
170     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 "")
171 ax1.set_xlabel(r'$\%$ H2O2$')
172 ax1.set_ylabel(r'$C_F$')
173 #ax1.legend(numpoints=1,loc='upper center')
174 fig.savefig('Part1.iv.pdf',bbox_inches='tight')
175
176 # Part 1 specific plots desired v
177 fig=plt.figure(figsize=(3,4))
178 ax1=fig.add_subplot(111)
179 for k in range(0,W*PS-(W-1),W):
180     ax1.plot(100-x1['f'][W-1+k::W*PS],x1['ivac'][W-1+k::W*PS],'.- ',label='$I_{vac}$ Nozzle Exit' if k==0
181         else "")
182     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
183         "")
184     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
185         else "")
186 ax1.set_xlabel(r'$\%$ H2O2$')
187 ax1.set_ylabel(r'$I_{sp}$ (s)$')
188 ax1.legend(numpoints=1,loc='upper center',bbox_to_anchor=(0.5,1.425))
189 fig.savefig('Part1.v.pdf',bbox_inches='tight')
190
191 # Part 1 specific peroxide concentration required?
192 print "Peroxide concentration requires a 5:1 Oxide to fuel ratio to vaporize all water"
193
194 # Part 1 What at is required for the optimal operating chamber pressure at 90% H2O2 concentration?,
195     corresponding Isp
196 #print 0,W*PS-(W-1),W
197 #print np.size(at[2+0::W*PS]),W*PS
198 #print "A* and optimal Isp at 90% H2O2 for 3114N thrust"
199 for k in range(0,W*PS-(W-1),W):
200     for i in range(W-1+k,np.size(at),W*PS):
201         #print i,k,x1['f'][i],at[i]
202         if (x1['f'][i]==10 and x1['p'][i]==100440.):
203             print "90% H2O2 and optimal pressure gives:"
204             print "    at = ",at[i]
205             print "    isp= ",x1['isp'][i]
206             print "    isp_opt= ",isp_opt[i]
207             print "    mdot= ",mdot[i]
208
209 # Part 1 specific plots desired vi
210 fig=plt.figure(figsize=(3,4))
211 ax1=fig.add_subplot(111)

```

```

208     for k in range(0,W*PS-(W-1),W):
209         ax1.plot(100-x1['f'][W-1+k:W*PS],mdot[W-1+k:W*PS],'.-')
210     ax1.set_xlabel(r'$\% \ H_2O_2$')
211     ax1.set_ylabel(r'$\dot{m} \ \frac{kg}{s}$')
212     fig.savefig('Part1_vi.pdf',bbox_inches='tight')
213
214 def output_file():# output file
215     f=open('output.txt','w')
216     # headings
217     f.write('# ')
218     for t in x1.dtype.names:
219         f.write(t+' ')
220     f.write('isp_opt ')
221     f.write('t0 ')
222     f.write('p0 ')
223     f.write('c_star ')
224     f.write('at ')
225     f.write('p0_at ')
226     f.write('m_dot \n')
227     #for i in x1.dtype.names:
228         #for j in np.size(x1[i]):
229     for i in range(0,np.size(x1['gam'])):
230         for t in x1.dtype.names:
231             f.write(str(x1[t][i]))
232             f.write(' ')
233             f.write(str(isp_opt[i]))
234             f.write(' ')
235             f.write(str(t0[i]))
236             f.write(' ')
237             f.write(str(p0[i]))
238             f.write(' ')
239             f.write(str(c_star[i]))
240             f.write(' ')
241             f.write(str(at[i]))
242             f.write(' ')
243             f.write(str(p0_at[i]))
244             f.write(' ')
245             f.write(str(mdot[i]))
246             f.write(' \n')
247     f.close()
248
249 def plot_data(xlabel,x,ylabel,y,filename):
250     fig=plt.figure(figsize=(3,3))
251     ax1=fig.add_subplot(111)
252     ax1.plot(x,y,'o')
253     ax1.set_xlabel(xlabel)
254     ax1.set_ylabel(ylabel)
255     #ax1.axis('equal')
256     fig.savefig(filename,bbox_inches='tight')
257
258 def example(): # example from section 7.1
259     global PS
260     global W
261     PS=5
262     W=2
263     # get argument list using sys module
264     global x1
265     x1=get_data_dtypes(filename1)
266
267     # subroutines for part 1
268     # initialize values
269     global c_star
270     global p0
271     global p0_at
272     global t0

```

```

273 global at
274 global mdot
275 global isp_opt
276
277 c_star = np.zeros(np.size(x1['gam']))
278 p0      = np.zeros(np.size(x1['gam']))
279 p0_at   = np.zeros(np.size(x1['gam']))
280 t0      = np.zeros(np.size(x1['gam']))
281 at      = np.zeros(np.size(x1['gam']))
282 mdot    = np.zeros(np.size(x1['gam']))
283 isp_opt = np.zeros(np.size(x1['gam']))
284
285 calcs(110.)
286 big()
287 #Part1-plots()
288 #output-file()
289
290
291 def Part1(): # Part 1
292     global PS
293     global W
294     PS=8
295     W=3
296     # get argument list using sys module
297     global x1
298     x1=get_data_dtypes(filename1)
299
300     # subroutines for part 1
301     # initialize values
302     global c_star
303     global p0
304     global p0_at
305     global t0
306     global at
307     global mdot
308     global isp_opt
309
310     c_star = np.zeros(np.size(x1['gam']))
311     p0      = np.zeros(np.size(x1['gam']))
312     p0_at   = np.zeros(np.size(x1['gam']))
313     t0      = np.zeros(np.size(x1['gam']))
314     at      = np.zeros(np.size(x1['gam']))
315     mdot    = np.zeros(np.size(x1['gam']))
316     isp_opt = np.zeros(np.size(x1['gam']))
317
318     calcs(3114.)
319     big()
320     Part1-plots()
321     output_file()
322
323 def Part2_iter(): # Part 2
324     global PS
325     global W
326     PS=8
327     W=3
328     # get argument list using sys module
329     global x1
330     x1=get_data_dtypes(filename1)
331
332     # subroutines for part 1
333     # initialize values
334     global c_star
335     global p0
336     global p0_at
337     global t0

```

```

338 global at
339 global mdot
340 global isp_opt
341
342 c_star = np.zeros(np.size(x1['gam']))
343 p0      = np.zeros(np.size(x1['gam']))
344 p0_at   = np.zeros(np.size(x1['gam']))
345 t0      = np.zeros(np.size(x1['gam']))
346 at      = np.zeros(np.size(x1['gam']))
347 mdot    = np.zeros(np.size(x1['gam']))
348 isp_opt = np.zeros(np.size(x1['gam']))
349
350 calcs(3114.)
351 # Part 2 specific plots for iteratin
352 fig=plt.figure(figsize=(3,4))
353 ax1=fig.add_subplot(111)
354 for k in range(0,W*PS-(W-1),W):
355     ax1.plot(100-x1['f'][W-1+k::W*PS],x1['p'][W-1+k::W*PS],'.- ',label=k)
356 ax1.set_xlabel(r'$\% \backslash$ ABSS$')
357 ax1.set_ylabel(r'$P \backslash$ (Pa)$')
358 ax1.legend(numpoints=1,loc='best')
359 fig.savefig('Part2_p.pdf',bbox_inches='tight')
360 output_file()
361
362 def Part2(): # Part 2, combined files
363     # get all txt files
364     W=3
365     PS=8
366     s= np.sort(glob.glob("./Part2/*/*.txt"))
367     max_values=np.zeros(np.size(s))
368     loc_of=np.zeros(np.size(s))
369     loc_isp_opt=np.zeros(np.size(s))
370     #loc_isp=np.zeros(np.size(s))
371     fig = plt.figure(figsize=(3,3))
372     fig2 = plt.figure(figsize=(3,3))
373     ax1=fig.add_subplot(111)
374     ax2=fig2.add_subplot(111)
375     for f in range(0,np.size(s)):
376         filename=s[f]
377         x_data=get_data_dtypes(filename)
378         max_values[f] = np.max(x_data['c_star'][2::3])
379         for i in range(0,np.size(x_data['c_star'])):
380             if (x_data['c_star'][i] == max_values[f] and x_data['isp'][i] !=0):
381                 loc_of[f]=x_data['of'][i]
382                 loc_isp_opt[f]=x_data['isp_opt'][i]
383         for k in range(0,W*PS-(W-1),W):
384             ax1.plot(100-x_data['f'][W-1+k::W*PS],x_data['isp_opt'][W-1+k::W*PS],'.- ',label=filename if k==0
385                     else '')
386             ax2.plot(100-x_data['f'][W-1+k::W*PS],x_data['c_star'][W-1+k::W*PS],'.- ',label=filename if k==0 else
387                     '')
388         ax1.set_xlabel('O/F')
389         ax2.set_xlabel('O/F')
390         ax1.set_ylabel('$I_{sp}$')
391         ax2.set_ylabel('$C*$')
392         ax1.legend(numpoints=1,loc='best')
393         ax2.legend(numpoints=1,loc='best')
394         fig.savefig('Part2_Isps.pdf',bbox_inches='tight')
395         fig2.savefig('Part2_C_stars.pdf',bbox_inches='tight')
396     # c*
397     fig = plt.figure(figsize=(3,3))
398     ax1=fig.add_subplot(111)
399     ax1.plot(np.array([80,85,90,95,99]),max_values,'o-')
400     ax1.set_xlabel('Peroxide Mass fraction')
401     ax1.set_ylabel('C*')
402     fig.savefig('Part2_C_star.pdf',bbox_inches='tight')

```

```

401
402 # O/F
403 fig = plt.figure(figsize=(3,3))
404 ax1=fig.add_subplot(111)
405 ax1.plot(np.array([80,85,90,95,99]),loc_of,'o-')
406 ax1.set_xlabel('Peroxide Mass fraction')
407 ax1.set_ylabel('O/F')
408 fig.savefig('Part2_OF.pdf',bbox_inches='tight')
409
410 # Isp
411 fig = plt.figure(figsize=(3,3))
412 ax1=fig.add_subplot(111)
413 ax1.plot(np.array([80,85,90,95,99]),loc_isp_opt[:],'o-')
414 #ax1.plot(np.array([80,85,90,95]),loc_isp[:-1:], 'o-')
415 ax1.set_xlabel('Peroxide Mass fraction')
416 ax1.set_ylabel('Optimal $I_{sp}$')
417 fig.savefig('Part2_OptimalIsp.pdf',bbox_inches='tight')
418
419 def Part3(): # Part 2, combined files
420     # get all txt files
421     s= np.sort(glob.glob("./Part2/*/*.txt"))
422     max_values=np.zeros(np.size(s))
423     loc_of=np.zeros(np.size(s))
424     loc_isp_opt=np.zeros(np.size(s))
425     loc_isp=np.zeros(np.size(s))
426     for f in range(0,np.size(s)):
427         filename=s[f]
428         x_data=get_data_dtypes(filename)
429         max_values[f] = np.max(x_data['c_star'][2::3])
430         for i in range(0,np.size(x_data['c_star'])):
431             if (x_data['c_star'][i] == max_values[f] and x_data['isp'][i]!=0):
432                 loc_of[f]=x_data['of'][i]
433                 loc_isp_opt[f]=x_data['isp_opt'][i]
434                 loc_isp[f]=x_data['isp'][i]
435
436     # ratio of Isp/Isp (hybrid/monopropellant)
437     fig = plt.figure(figsize=(3,3))
438     ax1=fig.add_subplot(111)
439     ax1.plot(np.array([80,85,90,95,99]),(loc_isp_opt[:])/137.69999,'o-')
440     ax1.set_xlabel('Peroxide Mass fraction')
441     ax1.set_ylabel(r'$\frac{t_2}{t_1}$')
442     fig.savefig('Part3.pdf',bbox_inches='tight')
443
444
445
446
447 def main():
448     #user defined values
449     #sys.argv
450     global filename1
451     filename1 = str(sys.argv[1])
452
453     # example stuff
454     #example()
455
456     # Part 1 stuff
457     #Part1()
458
459     # part 2 stuff
460     #Part2_iter()
461     Part2()
462
463     # part 3 stuff
464     #Part3()
465

```

```
466 | if __name__ == '__main__':  
467 |     main()
```