Rocket Design and Fly

Stephen Chen
Student ID: 50225769
University at Buffalo
MAE 423 Final Project

May 11, 2021

Contents

Problem Statement

The purpose of this final project is to explore the process of designing and testing sounding rockets given the desired design specifications using contraints based on trajectory theory, propulsion systems, structural strength of materials, and flight stability. To build the rockets, a number of inputs will be required to define the rocket's physical shape and properties. By comparing and analyzing the flight properties of the resultant rockets with their input parameters, we shoul de able to determine how input parameters affect the properties of the rocket. To simplify and limit the infinite number of designs that are possible when designing a rocket, the following basic geometry is followed for all of the rockets built.

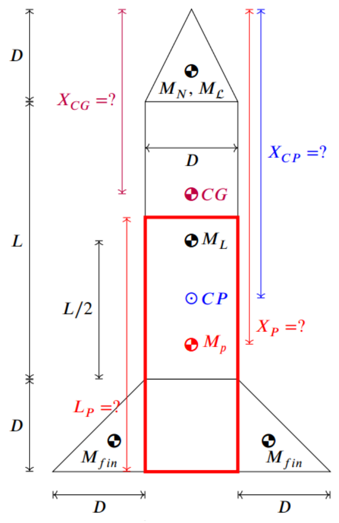

Figure 1: Basic Rocket Geometry

By following this basic geometry, a rocket's design can be defined by only two design parameters, length L, and diameter D.

Other design parameters that are necessary in defining the physical properties of the rocket are the following:

Table 1: Rocket Design Specifications

Design Specifications	Value	Description
M_L	5	Payload mass [kg]
h_{max}	10k, 20k, 30k	Maximum altitude [ft]
a^*_{max}	5, 10, 20	Normalized max. acceleration
SM	1, 2, 3	Static margin
ρ_s	1820	Shell density [$\frac{kg}{m^3}$]
ρ_p	1772	Propellant density [$\frac{kg}{m^3}$]
σ_s	80	Shell working stress [MPa]
N	4	Number of fins

From Table 1, we can see that some values remain constant while others include a selection. The constant values include the payload mass located in the cone of the rocket, the shell density, propellant density, shell working stress, and number of fins. The values that include a selection are the values that will be changed when designing the various rocket cases.

To configure the rocket for each case, we must decide on a diameter and length for the rocket. Since the rockets can be built in any size with any precision, this could also result in an infinite number of rockets. To limit and choose the best rocket configuration for each case, we will be iteratively testing each rocket sizing for flight stability and maximizing the payload coefficient.

A computer program was written in Python to automatically create and test rockets for each design case by creating millions of rockets with varying diameter and lengths. The diameter of the rockets ranged from 0.001m to 3m in 0.001m increments. The length of the rockets ranged from 0.001m to 5m in 0.001m increments. With these sets of design parameters, the program could create 15 million different sizing configurations for each case.

With the specification options shown in Table 1, we can see that a total of 27 configurations can be created. However, to make the data more simple and easier to analyze, a total of 9 cases will be generated instead. To accomplish this, when selecting the parameters from the selections, the other two parameters are set to the medium option. For example, if we want to generate a design case of $h_{max} = 30k$, then $a^*_{max} = 10$ and $SM = 2$. The cases conducted are shown in Table 2.

Table 2: Rocket Design Specifications

Cases	h_{max}	a^*_{max}	**SM**
1	10k	10	2
2	20k	10	2
3	30k	10	2
4	20k	5	2
5	20k	10	2
6	20k	20	2
7	20k	10	1
8	20k	10	2
9	20k	10	3

METHOD OF SOLUTION

The program starts by calculating R_{opt}, W_{eq}, and t_b with the design specification variables using the equations below. These equations are viable because we are assuming a linear burn with the chemical propellant.

$$R_{opt} = a^*_{max} + 1 \tag{1}$$

$$W_{eq} = \sqrt{\frac{h_{max} g}{\frac{ln(R_{opt})}{2}[ln(R_{opt} - 2)] + \frac{R_{opt}-1}{R_{opt}}}} \tag{2}$$

$$t_b = \frac{(R_{opt} - 1)W_{eq}}{gR_{opt}} \tag{3}$$

where R_{opt} is the optimal mass ratio, representing the ratio of initial to burnout mass of the rocket, W_{eq} is the equivalent velocity of the rocket, and t_b is the burnout time. These values can then be used to calculate the equivalent Mach number M_{eq} and total pressure P_0 within the rocket's combustion chamber.

$$M_{eq} = \frac{W_{eq}}{a} \tag{4}$$

$$P_0 = \frac{P_a}{(1 + \frac{\gamma-1}{2} M_{eq}^2)^{\frac{-\gamma}{\gamma-1}}} \tag{5}$$

where $\gamma = 1.4$ is the specific heat ratio of air, a is the speed of sound, and P_a is the atmospheric pressure. The speed of sound and atmospheric pressure was found assuming that the rocket is fired and flies near the surface of the Earth.

Then, after guessing the diameter of the rocket, the thickness of the rocket shell can be calculated using the hoop stress equation shown below with σ_s being the working stress of the rocket shell.

$$\delta = \frac{DP_0}{2\sigma_s} \tag{6}$$

After guessing the length of the diameter, we can calculate the mass of the rocket shell, mass of the rocket propellant, and length of the propellant tank. The equations used to calculate them are shown below. The mass of the rocket shell is calculated by simplifications including assuming the thickness of the shell is much less than the diameter ($\sigma_s << D$). We also assume the fins are the same thickness as the shell.

$$M_{s,cone} = \frac{\pi D}{2}\left(\frac{D}{2} + \sqrt{D^2 + (\frac{D}{2})^2}\right)\delta\rho_s \tag{7}$$

$$M_{s,body} = \pi D(L + D)\delta\rho_s \tag{8}$$

$$M_{s,fin} = \frac{D^2}{2}\delta\rho_s \tag{9}$$

$$M_s = M_{s,cone} + M_{s,body} + NM_{s,fin} \tag{10}$$

$$M_p = (R_{opt} - 1)(M_s + M_L) \tag{11}$$

$$L_p = \frac{4M_p}{\pi D^2 \rho_p} \tag{12}$$

The location of the center of pressure X_{cp} and center of gravity X_{cg} are calculated with the following equations. See Appendix for derivation of X_{cp} and X_{cg}.

$$X_{cp} = \frac{\frac{4}{3}\left[L + \left[\frac{16}{1+\sqrt{1+(2.5D)^2}}\right](L + \frac{3D}{2})\right]}{2 + \frac{4}{3}\left[\frac{16}{1+\sqrt{1+(2.5D)^2}}\right]} \tag{13}$$

$$X_{cg} = \frac{(M_{s,cone} + M_L)(\frac{2D}{3}) + M_{s,body}(D + \frac{L+D}{2}) + M_p((2D + L) - \frac{L_p}{2}) + NM_{s,fin}(L + \frac{5D}{3})}{M_{s,cone} + M_{s,body} + M_p + NM_{s,fin}} \tag{14}$$

The payload ratio λ and structural ratio ϵ was calculating using the following equations.

$$\lambda = \frac{M_L}{M_p + M_s} \tag{15}$$

$$\epsilon = \frac{M_s}{M_s + M_p} \tag{16}$$

The overall mass of the rocket could be found by summing all of the mass of the rocket's components.

$$M_o = M_s + M_L + M_p \tag{17}$$

Discussion of Results

After successfully configuring the 9 rocket cases, the results can be seen in Table 3 and Table 4 below.

Table 3: Rocket Case Configuration Results 1

Cases	R_{opt}	W_{eq} [m/s]	t_b [s]	P_o/P_a	δ/D	D [m]	L/D
1	11	146.871	13.611	1.132	7.168e-4	0.168	8.226
2	11	207.708	19.248	1.276	8.079e-4	0.165	10.073
3	11	254.389	23.574	1.432	9.071e-4	0.172	6.401
4	6	304.075	25.830	1.653	1.047e-3	0.139	8.151
5	11	207.708	19.248	1.276	8.079e-4	0.165	10.073
6	21	153.368	14.889	1.144	7.247e-4	0.212	7.604
7	11	207.708	19.248	1.276	8.079e-4	0.203	3.975
8	11	207.708	19.248	1.276	8.079e-4	0.165	10.73
9	11	207.708	19.248	1.276	8.079e-4	0.143	18.434

Table 4: Rocket Case Configuration Results 2

Cases	X_{CG}	X_{CP}	M_p	M_s	M_o	λ	ϵ
1	0.972	1.308	52.074	0.207	57.281	9.564e-2	3.967e-3
2	1.194	1.524	52.598	0.260	57.858	9.460e-2	4.915e-3
3	0.749	1.093	52.335	0.233	57.568	9.512e-2	4.441e-3
4	0.794	1.072	25.852	0.170	26.022	1.921e-1	6.547e-3
5	1.194	1.524	52.598	0.260	57.858	9.460e-2	4.915e-3
6	1.127	1.551	107.936	0.397	113.333	4.615e-2	3.663e-3
7	0.701	0.904	52.481	0.248	57.730	9.482e-2	4.706e-3
8	1.194	1.524	52.598	0.256	57.858	9.459e-2	4.915e-3
9	1.828	2.257	52.821	0.282	58.093	9.416e-2	5.311e-3

We can see from the tables above that changing the design specification parameters affected multiple properties of the resultant rocket. Cases 1-3 show us that by increasing the h_{max}, the equivalent velocity W_{eq}, burnout time t_b, pressure ratio P_o/P_a, and thickness ratio δ/D also increases. This trend makes sense because to reach higher velocities, the rocket must be propelled for longer. This causes the equivalent velocity, burnout time and pressure ratio to increase. Then, to contain the increased pressure of the combustion chamber, the thickness of the shell must also increase.

From analyzing cases 4-6, we can see that by increasing the normalized acceleration a^*_{max}, the mass ratio R_{opt}, diameter D, propellant mass M_p, and overall mass M_o of the rocket also increased. We can also see that δ/D, W_{eq}, t_b, P_o/P_a, payload ratio λ, and structural ratio ϵ decreased with increased acceleration. These results make sense because to get increased thrust and acceleration, the burnout time of the rocket must decrease along with the equivalent velocity. The decreased equivalent velocity means lower pressure ratio which leads to thinner shell thickness. The mass of the propellant must increase drastically, with the highest acceleration case having more than four times the propellant compared to the lowest case. Because of this, the rocket also requires more shell material (more than double) which further increased the total mass of the rocket. We can also see that due to the increased mass while payload mass remains constant, the payload ratio drastically decreased to about 25%.

Analyzing cases 7-9, we can see that changing the static margin SM does not affect the rocket's flight properties but instead affect the rocket's size. By increasing the static margin of the rocket, the length, CG location, CP location, and structural ratio also increased. The rocket's diameter and payload ratio decreased however with increasing static margin. This result tells us that by increasing the static margin of the rocket, the resultant rocket will becomes skinnier, with larger L and smaller D. Because of this, the location of the center of gravity and center of pressure must move with the increased length. The mass of the propellant also inreased slightly (~0.6%), most likely due to the slightly increased shell mass and overall mass due to the longer rocket body. The structural ratio increased by a larger percentage (~12.9%) because the shell mass increased more than the propellant mass, ~13.7% compared to ~0.6%.

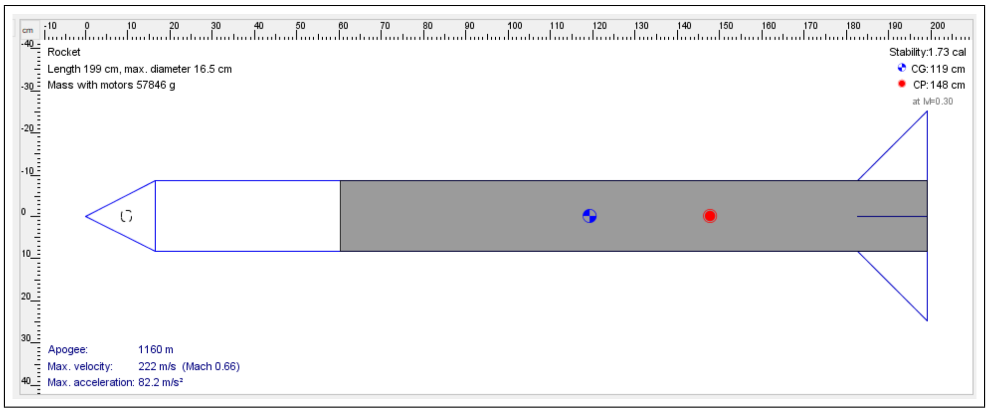
Rocket
Length 199 cm, max. diameter 16.5 cm
Mass with motors 57846 g
Stability:1.73 cal
CG:119 cm
CP:148 cm
at M=0.30
Apogee: 1160 m
Max. velocity: 222 m/s (Mach 0.66)
Max. acceleration: 82.2 m/s²

Figure 2: OpenRocket Design Based on Configuration Case 5

Figure 2 above shows the design of the rocket for case 5 created in the OpenRocket software. By comparing the CG and CP location from the figure to that of Table 4, we can see that the center of gravity location is off by ~0.3% and center of pressure location is off by ~2.97%. This tells us that our program calculates the CG and CP locations correctly and matches the OpenRocket software very closely. The slight error in the CP location may be because we used the method prescribed in the Centuri report where pressure of the rocket body is ignored but OpenRocket may have used a different method where it is not.

Figure 3 shows the simulated ideal flight of the OpenRocket model where drag is ignored. The simulation results tells us that the maximum altitude achieved by the rocket is 6134 meters which closely matches the expected altitude of 6096 meters, resulting in a 0.60% error. The expected maximum velocity of the rocket was calculated to be 309.24 $\frac{m}{s}$. The simulated flight returned a maximum velocity of 309.84 $\frac{m}{s}$, resulting in an error of 0.19%. The maximum normalized acceleration from the simulated flight was 9.89 which is very close to the expected value of 10, resulting in an error of 1.09%. This result tells us that the ideal flight case simulated in OpenRocket very closely matches our expected results. This accuracy is due to us ignoring the drag component of the rocket's flight.

Figure 3 shows the simulated flight of the OpenRocket model where drag is not ignored. Comparing this figure to Fig. **??**, we can see that the drag drastically affected the rocket's maximum height, velocity, and acceleration. The simulated maximum altitude of the drag affected flight was only 1874 meters, 69.26% less than the expected

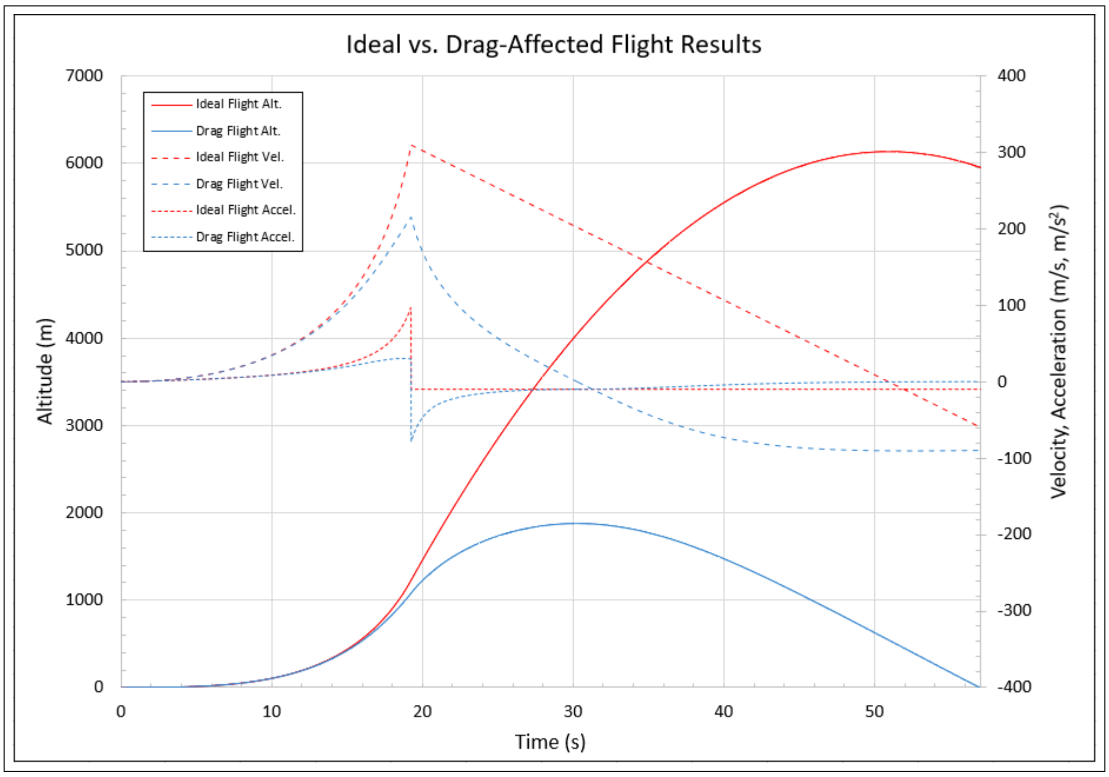

Figure 3: Simulated Flight Results for Ideal and Drag-Affected Rocket

value and 69.49% less than the idealized case. The maximum velocity of the drag affected flight was also lower at 215.31 $\frac{m}{s}$, 30.37% less than the expected value and 30.51% less than the idealized case. The maximum normalized acceleration of the drag affected flight was not only lower at 7.96 but also in the negative direction due to drag stopping the rocket from climbing. The maximum normalized acceleration of the flight was 20.39% less than the expected value and 19.50% less than the idealized case. These results tell us that drag severely impacted the rocket's ability to fly, inhibiting the rocket's ability to reach even 50% of the expected maximum altitude. Because of this, this rocket does not satisfy baseline case conditions. This result is expected because the equations used when configuring the rocket ignores the affect of drag.

To achieve greater max. altitude without increasing the amount of propellant, improvements were made to the geometry of the rocket. One of the improvements made to the rocket that also had the greatest impact was to switch the nose cone for one of a more aerodynamic shape. Of the five choices available in OpenRocket, the Haack Series cone shape appears to improve the rocket the most. From Fig. 4, we can see that by switching the conical nose cone out for the Haack Series nose cone, the max. altitude of the rocket increased from 1874 meters to 2235 meters, an increase of 19.26%. Another improvement made to the rocket was to give the fins of the rocket an airfoil cross-section and removing one of the fins, resulting in a three-fin design. Although doing so changed the center of pressure and center of mass of the rocket, the change was very small at only 3.04%. With the changes in the fins, the max. altitude of the rocket increased to 2253 meters, an improvement of 0.81% from the previous and a total improvement of 20.22% over the base design. Like the max. altitude, the max. velocity of the rocket increased from 215.31 $\frac{m}{s}$ to 242.73 $\frac{m}{s}$ (+12.74%). The max. acceleration of the rocket

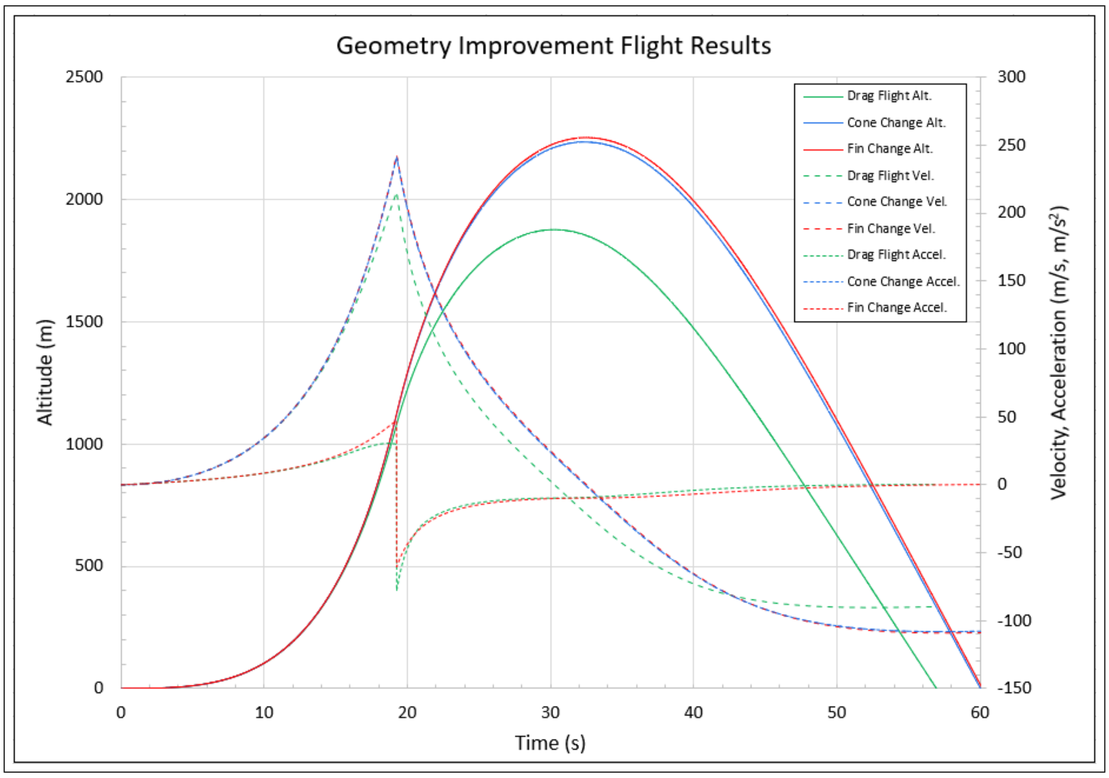

Figure 4: Flight Results for Geometrically Improved Rocket

actually decreased due to the more aerodynamic design, decreasing from 7.96 to 6.16 (-22.76%).

To further improve our rocket, the rocket's material was also changed. The purpose of changing the rocket's material is that lighter rockets require less force to accelerate. We will assume that the new material will be strong enough to withstand the internal pressures of the rocket since the shell mass is only a small portion of the rocket's total mass anyways. Instead of carbon-epoxy, weighing at about 1.82 $\frac{g}{cm^3}$, the material for the rocket shell was changed to be balsa, which only weighs 0.17 $\frac{g}{cm^3}$, roughly 10 times less. We can see from Fig. 5 that changing the material of the rocket increased the maximum altitude of the rocket from 2253 meters to 2264 meters, an increase of 0.49%. This increase in maximum altitude is very small because the propellant was what contributed to most of the rocket's weight, not its shell. The max. velocity and max. acceleration of the rocket increased slightly from 242.73 $\frac{m}{s}$ to 246.64 $\frac{m}{s}$ (+1.61%) and 6.16 to 6.57 (+6.48%) respectively.

The final improvement made to the rocket was to alter its burn-profile to make it more efficient. A new burn profile was created where instead of a linear burn throughout the entire burn time, the propellant was burnt faster towards the launch and slower as the burn carries on. Figure 6 shows the burn profile of this new engine. Although the new engine has the same impulse, propellant mass, and burn time as the old one, the new engine should be more efficient as less propellant is wasted trying to accelerate the rocket after it has gained speed, reducing fuel waste due to drag. We can see from Fig. 5 that with the new burn profile, the rocket's max. altitude increased from 2264 meters to 2639 meters, an increase of 16.12%. The max. velocity and max. acceleration decreased from 246.64 $\frac{m}{s}$ to 213.64 $\frac{m}{s}$ (-13.38%) and 6.57 to 6.65 (-44.45%). Although the max. velocity decreased,

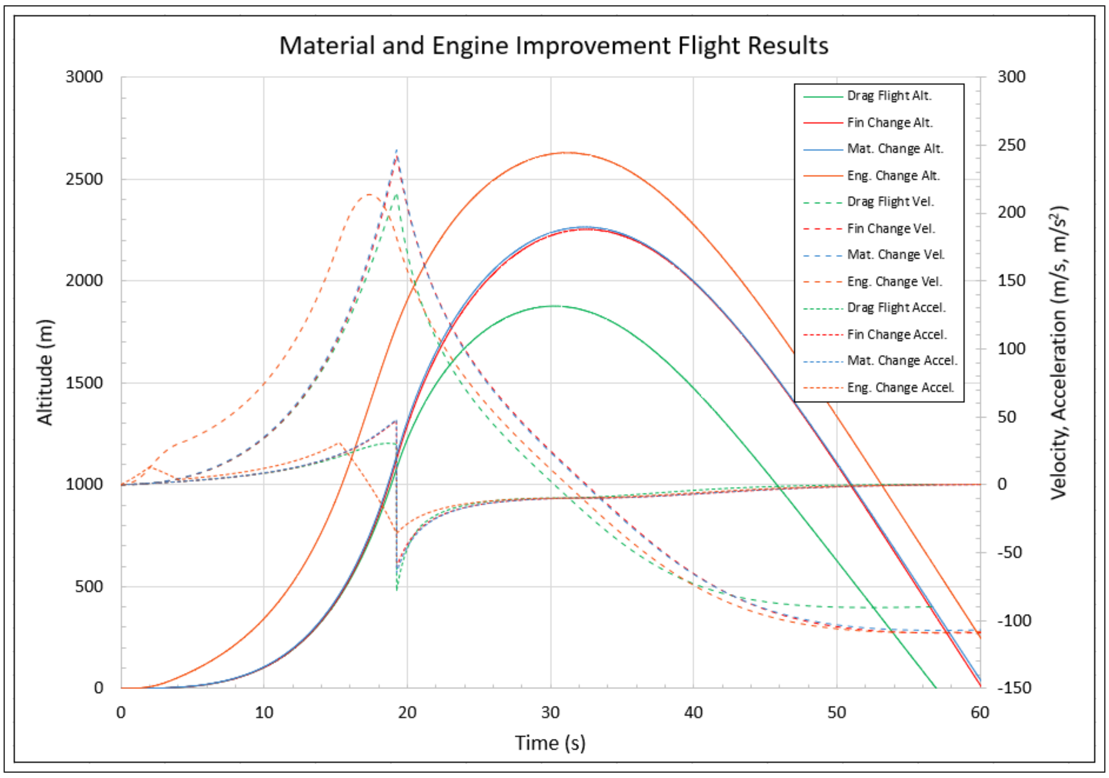

Figure 5: Flight Results for Material and Engine Improved Rocket

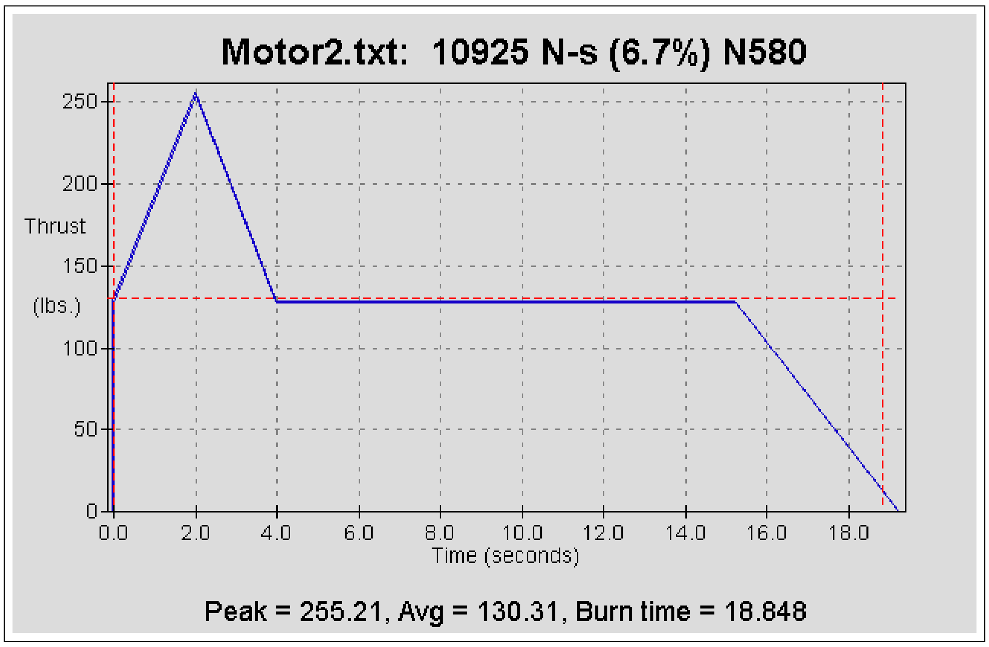

Figure 6: Burn Profile of New Engine

the rocket increased in velocity much sooner, allowing it to travel at fast speed for longer, resulting in greater altitude.

SUMMARY AND CONCLUSION

When designing a rocket, there are many parameters and variables that can affect the overall performance. Even with the simple rocket design studied in this project, a simple change in one input affects a number of other rocket properties. From part I of this project, we learned that to increase the max. altitude of a rocket, its burnout time and equivalent velocity must increase. This leads to increased internal pressure and requires a thicker shell to contain the pressure. To increase the normalized acceleration of the rocket, we must also drastically increase the amount of propellant while also decreasing the burnout time, equivalent velocity, and internal pressure. Finally, if we want to increase the static margin of the rocket, we must make the rocket longer and thinner.

The OpenRocket software very helpful and allowed us to experiment by test-flying different rocket configurations. From the experimentation, we found out that drag plays a very big role in rocket performance. By considering drag, the maximum altitude reached by our rocket decreased by 69.26%, resulting in less than half the height. However, by changing certain design parameters of our rocket, we were able to improve our rockets performance. By changing the conical nose cone for a Haack Series nose cone, we increased our max. altitude by 19.26%. We found out that altering the fin geometry does not seem to have a drastic effect on the performance although it would have a larger effect on rockets that do not fly straight up. Changing the material of the rocket shell also did not seem to have much of an effect on performance because the propellant was responsible for most of the rocket's weight, not its shell. Altering the rocket engine's burn profile however did make a big impact on the rocket's performance. By letting the rocket burn faster towards the beginning of launch and decreasing the burn towards the end, we were able to increase the rocket's max. altitude by 16.12%. In total, with all of the improvements on the rocket, we were able to increase it's max. altitude from 1874 meters to 2639 meters, resulting in an increase of 40.82%.

From this study, the most important thing we learned is that drag cannot be ignored for a rocket due to its high speed. Although a rocket is already aerodynamic, drag scales exponentially as velocity increases. Therefore, much attention should be placed on the aerodynamic properties of a rocket during its design. Another very important aspect to pay attention to is the burn profile of the rocket motor. By altering the burn profile, even rockets with the same propellant mass can reach much higher altitudes.

APPENDIX

Appendix A

This portion contains the equations used to derive the final equation used to calculate the location of center of pressure and center of gravity on the rocket.

$$(C_{N\alpha})_n = 2$$

$$\bar{X}_n = \frac{2L}{3}$$

$$(C_{N\alpha})_f = \frac{16}{1+\sqrt{1+(2.5D)^2}}$$

$$\bar{X}_f = L + D + \frac{D}{2}$$

$$k_{fb} = 1 + \frac{D}{3D}$$

$$X_{cp} = \frac{(C_{N\alpha})_n \bar{X}_n + k_{fb}(C_{N\alpha})_f \bar{X}_f}{(C_{N\alpha})_n + k_{fb(C_{N\alpha})_f}}$$

$$X_{cg} = \frac{\sum M_i \bar{X}_i}{\sum M_i}$$

Appendix B

This portion contains the Python code for the rocket configuration program.

```
# Stephen Chen
# MAE423: Intro to Propulsion
# Created: 4/23/2021

import math

# Defining inputs
Ml = 5   # Payload Mass [kg]
ps = 1820  # Shell  Density (carbon−epoxy) [1kg/m^3]
pp = 1772  # Propellant Density (66 − 78% AP, 18% organic polymer, 4 − 20% Al) [kg/m^3]
sigs  = 80000000 # Shell  Working Stress [Pa]
N = 4   # Number of fins
hmaxList = [3048, 6096, 9144] # Maximum Altitude [m] (10k, 20k, 30k ft)
amaxList = [5, 10,  20]   # Normalized Max Acceleration
SMList = [1,  2,  3]   # Static  Margin (Xcp−Xcg)/D
g = 9.81   # Acceleration due to Gravity [m/s^2]

# Defining constants
gamma = 1.4 #
pa = 101325 # Atmospheric pressure [Pa]
a  = 346         # Speed of sound at sea−level [m/s]

rocketList =  []
lambdas = []
newRockets = []


class Rocket():
    def __init__(self,  h,  a,  sm):
        self.hmax = h
        self.amax = a
        self.SM = sm
        self.CGmatch = False
        self.deltaD = 0.001
        self.deltaL = 0.001
        self.tol  = 10**−5
        self.lMax = 5  # Max length of rocket [m]
        self.dMax = 3 # Max diameter of rocket [m]
        self.lambdaMax = 0
        self.e = 0
        self.mdot = 0
        self.thrust = 0
        self.ropt, self.weq, self.tb, self.Meq, self.p0, self.delta, self.d, self.l, self.Xcp, self.Xcg,
            self.lambd, self.Mp, self.Ms, self.Lp = 0, 0, 0, 0, 0, 0, 0.001, 0.001, 0, 0, 0, 0, 0, 0

    def __str__(self):
        return "lambda: "+ str(self.lambd) + " d: " + str(self.d) + " l: " + str(self.l) + " delta:
            "+str(self.delta) + " Ms: "+str(self.Ms) + " Mp: "+str(self.Mp) + '\n' \
          + " Lp: "+str(self.Lp) + " Xcg: " + str(self.Xcg) + " Xcp: " + str(self.Xcp) + " Ropt: " +
              str(self.ropt) + " Weq: " + str(self.weq) + " tb:" + str(self.tb) + '\n' \
            + " P0/Pa" + str(self.p0/pa) + " e: " + str(self.e) + " hmax: " + str(self.hmax) + " amax: "
                + str(self.amax) + " SM: " + str(self.SM) + " T: " + str(self.thrust)

    # Step 1, calculate Ropt, Wep, tb, P0
    def Eq1(self):  # returns: [ropt, weq, tb]
        self.ropt = self.amax + 1
        self.weq = math.sqrt((self.hmax * g) / ((math.log(self.ropt) / 2) * (math.log(self.ropt) - 2) +
            ((self.ropt - 1) / self.ropt)))
        self.tb = ((self.ropt - 1) * self.weq) / (g * self.ropt)
        self.Meq = self.weq / a
        self.p0 = pa / ((1 + (gamma - 1) * 0.5 * self.Meq ** 2) ** (-gamma / (gamma - 1)))
        # return [ropt, weq, tb, p0]

    # Step 2, after guess D and L, calculate Ms, Mp, Lp, Xcp, Xcg
    def Eq2(self):
        self.delta = self.d * self.p0 / (2 * sigs)  # Thickness of the shell due to material limitations

        # Calculating mass of the rocket (shell and propellant)
        SAbody = math.pi * self.d * (self.l + self.d)  # Surface area of the body tube
        Afin = 0.5 * (self.d ** 2)  # Cross-sectional area of one fin
        SAcone = 0.5 * math.pi * self.d * (0.5 * self.d + math.sqrt(self.d ** 2 + 0.25 * self.d ** 2))  #
            Surface area of the rocket cone

        Msbody = SAbody * self.delta * ps # Assuming cylindrical shell and shell thickness << diameter
        Msfin = Afin * self.delta * ps  # Assuming fin thickness is equal to shell thickness
        Mscone = SAcone * self.delta * ps # Calculated using surface area of cone

        self.Ms = Msbody + N * Msfin + Mscone # Total mass of the rocket shell
        self.Mp = (self.ropt - 1) * (self.Ms + Ml) # Mass of propellant
        self.Lp = self.Mp / (math.pi * self.d * self.d * pp * 0.25)  # Length of the propellant tube
        # Calculating Xcp
        # For nose cone
        Cncone = 2
        xconebar = (2/3)*self.d

        # For fin
        a = self.d
        m = self.d
        b = 0
        l = math.sqrt((0.5*self.d)**2+self.d**2)
        s = self.d
        Cnfin = (4*N*(s/self.d)**2)/(1+math.sqrt(1+((2*l)/(a+b))**2))
        xfbar = self.l+self.d+(m*(a+2*b))/(3*(a+b))+(1/6)*(a+b-(a*b)/(a+b))
        kfb = 1+(self.d/2)/s+(self.d/2)
        Cnfinb = Cnfin*kfb

        self.Xcp = (Cncone*xconebar+Cnfinb*xfbar)/(Cncone+Cnfinb)

        # Calculating Xcg
        mcg = (Mscone + Ml) * (2 / 3) * self.d + Msbody * (self.d + (self.l + self.d) * 0.5) + self.Mp * ((2
            * self.d + self.l) - 0.5 * self.Lp) + N * Msfin * (
                self.d + self.l + (2 / 3) * self.d)
        # print(mcg)
        self.Xcg = mcg / (self.Ms + Ml + self.Mp)
        # return [Xcp, Xcg]

    def UpdateD(self):
        self.d += self.deltaD

    def UpdateL(self):
        self.l += self.deltaL

    def ResetL(self):
        self.l = 0.01

    def CheckMaxD(self):
        if self.d < self.dMax:
            return False
        else:
            return True
    def CheckMaxL(self):
        if  self.l  < self.lMax:
            return False
        else:
            return True

    def CheckCGMatch(self):
        if  abs(self.Xcp - self.Xcg - self.d *  self.SM) < self.tol:
            return True
        else:
            return False

    def CalcLambda(self):
        self.lambd = round(Ml / (self.Mp + self.Ms), 6)
        self.e = round(self.Ms/(self.Ms+self.Mp), 6)
        self.mdot = self.Mp/self.tb
        self.thrust =  self.weq*self.mdot

    def LpValid(self):
        if  self.Lp > self.l+self.d:
            return False
        else:
            return True


# Creates 27 different  rockets  with different  input parameters
def CreateRockets27():
    for h in hmaxList:
        for amax in amaxList:
            for sm in SMList:
                rocket = Rocket(h, amax, sm)
                rocket.Eq1()
                rocketList.append(rocket)


# Creates 9 rockets by using medium value for unfocused parameters
def CreateRockets9():
    for h in hmaxList:
        rocket = Rocket(h, amaxList[1], SMList[1])
        rocket.Eq1()
        rocketList.append(rocket)
    for amax in amaxList:
        rocket = Rocket(hmaxList[1], amax, SMList[1])
        rocket.Eq1()
        rocketList.append(rocket)

    for sm in SMList:
        rocket = Rocket(hmaxList[1], amaxList[1], sm)
        rocket.Eq1()
        rocketList.append(rocket)


# Iteratively configures the rocket's size
def ConfigureRockets():
    global newRockets
    for rocket in rocketList:
        print("Testing rocket for hmax = " + str(rocket.hmax) + ", amax = " + str(rocket.amax) + ", SM = "
            + str(rocket.SM))
        validRockets = {}
        bestLambda = 0
        while not (rocket.CheckMaxD() and rocket.CheckMaxL()):
            rocket.Eq2()
            if rocket.LpValid():
                if rocket.CheckCGMatch():
                    rocket.CalcLambda()
                    lambdas.append(rocket.lambd)
                    if rocket.lambd > bestLambda:
                        validRockets[rocket.lambd] = (rocket.l, rocket.d)
                        bestLambda = rocket.lambd
            if not rocket.CheckMaxL():
                rocket.UpdateL()
            else:
                rocket.ResetL()
                rocket.UpdateD()
        validMax = max(validRockets.keys())
        (maxl, maxd) = validRockets[validMax]
        newRock = Rocket(rocket.hmax, rocket.amax, rocket.SM)
        newRock.l = maxl
        newRock.d = maxd
        newRock.Eq1()
        newRock.Eq2()
        newRock.CalcLambda()
        newRockets.append(newRock)
```

```
CreateRockets9()
ConfigureRockets()

print(lambdas)
print(len(lambdas))
print(max(lambdas))
for each in newRockets:
    print(each)
    print('')
```