# FINAL PROJECT:
# ROCKET DEISGN AND FLY

## MAE 423: Introduction to Propulsion

Instructor:
Dr. Paul E. DesJardin

Ayesha Khatun
Date: 05/12/2020

**Problem Statement:**

The project aims to design a sounding rocket given a set of design parameters, with the focus of maximizing the efficiency. The input parameters also reflect constraints on the propulsion system, material properties (strength of the material) of structural material, and aerodynamic performance, etc. The first part of the project requires getting a set of outputs parameters given the set of input parameters. The second part requires the simulation of designed rockets using the OpenRocket software, which provides the validity of satisfying design requirements as well as an in-depth understanding of important aspects of the rocket design.

The given set of input parameters and the expected output parameters are listed below subsequently in Table 1 and 2, along with a short description of each parameter listed:

**Table 1: Design Input Parameters**

| Parameter | Description of Parameter | Assigned value |
|---|---|---|
| $M_L$ | Payload mass | 1 Kg |
| $h_{max}$ | Maximum altitude | 10k, 20k, 20k |
| $a^*_{max}$ | Normalized max. acceleration | 5, 10, 20 |
| SM | Static margin = $(X_{CP} - X_{CG})/\text{D}$ | 1, 2, 3 |
| $\rho_s$ | Shell (Aluminum) density | 2700 $Kg/m^3$ |
| $\rho_p$ | Propellant density | 1772 $Kg/m^3$ |
| $\sigma_s$ | Shell working stress | 60 MPa |
| N | Number of fins | 3 |

## Table 2: Design Output Parameters

| Parameter | Description of Parameter |
|---|---|
| R | Initial to burnout mass ratio $\frac{M_L+M_P+M_S}{M_L+M_S} = \frac{M_o}{M_b}$ |
| $W_{eq}$ | Equivalent/ effective velocity of gas at nozzle exhaust |
| $t_b$ | Burn time |
| $P_0$ | Exit Pressure |
| D | Diameter of the rocket tube/ Height of rocket nose and fins |
| L | Length of the rocket tube |
| $\delta/\mathrm{D}$ | Thickness to D ratio |
| $X_{CG}$ | Center of mass/gravity |
| $X_{CP}$ | Center of pressure |
| $M_P$ | Propellant mass |
| $M_o$ | Total mass, $M_o = M_P + M_L + M_S$ |
| $\lambda$ | The ratio of the specific heats, $C_P/C_V$ |
| $\epsilon$ | The ratio of mass, $\frac{M_S}{M_P+M_S}$ |

The basic rocket geometry below in Fig. 1 is also provided for the project. The goal for the first part therefore becomes to determine the parameters D and L to begin with and then to determine other required output parameters using the given constrains.

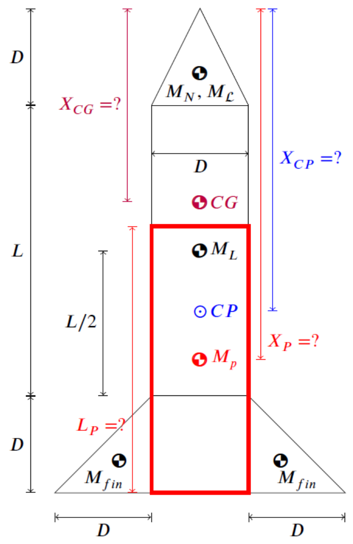


**Figure 1:** Parameterized Rocket Geometry.

## Method of Solution:

An instructional flow diagram was provided by the instructor to approach solving for output parameters using the given parameters while maintaining the required constraints.

To calculate R,

$$R = \frac{M_L + M_P + M_s}{M_L + M_s}$$

Where, $M_L =$ propellant mass, $M_P = \mathit{propellant\ mass\ and}\ M_s = \mathit{mass\ of\ the\ rocket\ structure}.$

Using R,

$$W_{eq} = \frac{h_{\max} * g}{\left(\frac{\ln(R_{\max})}{2}\right)(\ln(R_{\max}) - 2)\,((R_{\max} - 1)/R_{\max})}$$

To calculate the burn time, $t_b$

$$t_b = \frac{(R_{max} - 1)\,W_{eq}}{g\,R_{max}}$$

The exit pressure $P_o$, can be calculated using the isentropic relation:

$$\frac{P_o}{P_a} = \left[1 + \frac{\gamma - 1}{2} M_{eq}^2\right]^{\gamma/\gamma - 1}$$

Where, $P_a = 101325\ Pa$, and $\gamma = 1.4$

$M_{eq}$ is the equivalent Mach number which is calculated using the following equation:

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

Here, a is the airspeed calculated using the atmospheric temperature, T = 298K as,

$$a = \sqrt{(\gamma RT)}$$

To calculate the mass of the total rocket, the weight of different components needs to be added together. The following formulas were collected from the century notes to use:

**Nose:** $M_{S,nose} = \rho_s\,\delta\,\pi \left(1 + \sqrt{\left(\frac{5}{4}\right)}\right) D^2;\ X_{CG,nose} = \frac{2D}{3};\ X_{CP,nose} = \frac{2D}{3};\ C_{P,nose} = 2$

**Main tube:** $M_{S,maintube} = \rho_s\,\delta\,\pi\,D\,L;\ X_{CG,maintube} = D + \frac{L}{2};\ X_{CP,maintube} = D + \frac{L}{2};\ C_{P,maintube} = 0$

**Fin tube:** $M_{S,fintube} = \rho_s\,\delta\,\pi\,D^2;\ X_{CG,fintube} = D + L + \frac{D}{2};\ X_{CP,fintube} = D + L + \frac{D}{2};\ C_{P,fintube} = 0$

**Fins:** $M_{S,fins} = 0.5\,N\,\delta\,\rho_s\,D^2;\ X_{CG,fins} = D + L + \frac{2D}{3};\ X_{CP,fins} = L + \frac{3D}{2};\ C_{P,fins} = 4.6384$

So, the mass of total structure,

$$M_s = M_{S,nose} + M_{S,maintube} + M_{S,fintube} + M_{S,fins}$$

The center of gravity of total structure,

$$X_{CG} = X_{CG,nose} + X_{CG,maintube} + X_{CG,fintube} + X_{CG,fins}$$

The center of pressure of total structure,

$$X_{CP} = X_{CP,nose} + X_{CP,maintube} + X_{CP,fintube} + X_{CP,fins}$$

Then, the other required output parameter was calculated using following equations:

Mass of propellant,

$$M_p = (R-1)(M_s + M_L)$$

The thickness of the structure,

$$\delta = \frac{DP_o}{2\sigma_s}$$

The length of the rocket body,

$$L_p = M_p / \left( (\pi \rho_p D^2)/4 \right)$$

The constrains to find appropriate values of the rocket diameter, D and the length L are:

$$constraint\ 1, \qquad X_{CP} - X_{CG} - D * SM = 0$$

$$constraint\ 2, \qquad \lambda = \lambda_{\max}$$

Finally, the ratio of the specific heats,

$$\lambda = \frac{M_L}{(M_s + M_p)}$$

And, the ratio of the mass,

$$\epsilon = \frac{M_s}{(M_s + M_p)}$$

The calculations were done using python and the results attained from the code in appendix [] operation is tabulated in Table 3. One hand calculation is also carried out to assure the functionality of the code and validity of the results.

## Discussion of Results:

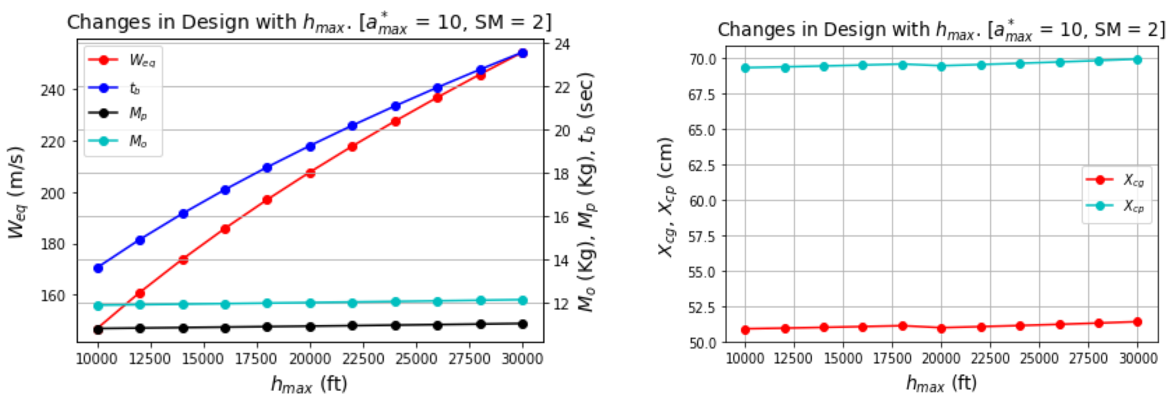
## Part I

1) Table 3 summarizes the output parameters for the inputs under different cases of consideration:

**Table 3: Design Analysis Summary**

| Output /Input | $h_{max}(ft) = 10k$ $a_{max} = 10$ SM = 2 | $h_{max}(ft) = 20k$ $a_{max} = 10$ SM = 2 | $h_{max}(ft) = 30k$ $a_{max} = 10$ SM = 2 | $h_{max}(ft) = 20k$ $a_{max} = 5$ SM = 2 | $h_{max}(ft) = 20k$ $a_{max} = 10$ SM = 2 | $h_{max}(ft) = 20k$ $a_{max} = 20$ SM = 2 | $h_{max}(ft) = 20k$ $a_{max} = 10$ SM =1 | $h_{max}(ft) = 20k$ $a_{max} = 10$ SM =2 | $h_{max}(ft) = 20k$ $a_{max} = 10$ SM=3 |
|---|---|---|---|---|---|---|---|---|---|
| **R** | 11 | 11 | 11 | 6 | 11 | 21 | 11 | 11 | 11 |
| $W_{eq}\ (\frac{m}{s})$ | 146.9 | 207.7 | 254.4 | 304.1 | 207.7 | 153.4 | 207.7 | 207.7 | 207.7 |
| $t_b\ (sec)$ | 13.6 | 19.2 | 23.6 | 25.8 | 19.2 | 14.9 | 19.2 | 19.2 | 19.2 |
| $P_0 (MPa)$ | 0.115 | 0.129 | 0.145 | 0.168 | 0.129 | 0.116 | 0.129 | 0.129 | 0.129 |
| $\delta/D$ | 0.00096 | 0.00108 | 0.00121 | 0.0014 | 0.00108 | 0.00097 | 0.00108 | 0.00108 | 0.00108 |
| **D (m)** | 0.0919 | 0.0922 | 0.0924 | 0.0764 | 0.0922 | 0.1149 | 0.1107 | 0.0922 | 0.0819 |
| **L (m)** | 0.8279 | 0.8293 | 0.8356 | 0.5764 | 0.8293 | 1.1582 | 0.5374 | 0.8293 | 1.0798 |
| $X_{CG}\ (cm)$ | 50.94 | 51.02 | 51.44 | 34.53 | 51.02 | 72.29 | 40.31 | 51.02 | 61.10 |
| $X_{CP}\ (cm)$ | 69.32 | 69.46 | 69.93 | 49.81 | 69.46 | 95.28 | 51.38 | 69.46 | 85.68 |
| $M_P\ (Kg)$ | 10.79 | 10.90 | 11.02 | 5.29 | 10.90 | 23.40 | 11.05 | 10.90 | 10.84 |
| $M_0\ (Kg)$ | 11.87 | 11.99 | 12.13 | 6.35 | 11.99 | 24.57 | 12.15 | 11.99 | 11.93 |
| $\lambda$ | 0.0920 | 0.0910 | 0.0899 | 0.1868 | 0.0910 | 0.0424 | 0.0897 | 0.0910 | 0.0915 |
| $\epsilon$ | 0.0073 | 0.0082 | 0.0092 | 0.0110 | 0.0082 | 0.0072 | 0.0094 | 0.0082 | 0.0077 |

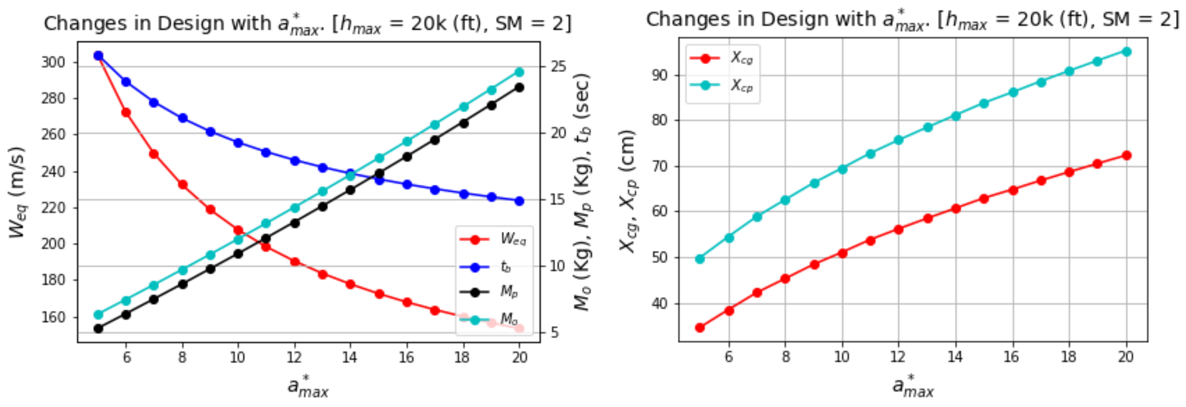
2) **Design changes with $h_{max}$ variation:** Figure 2 (a & b) below shows that for the increase in maximum height from10,000 ft to 30,000 ft, the required burn time and velocity increases by around 40%. While the burn time and equivalent velocity are so impactful to attain a higher destination, the $X_{CG}$ $and$ $X_{CP}$does not change drastically for the higher altitude. The changes in the $X_{CG}$ $and$ $X_{CP}$ are in fact less than 1% for variations in $h_{max}$.



**(a)** **(b)**

**Figure 2:** Design changes for $h_{max}$ variation

3) **Design changes with $a^*_{max}$ variation:** The lower value of 5 and the upper value of 20 was under consideration to understand the effect of $a^*_{max}$ change. For higher acceleration, the burn time and velocity logarithmically decrease in around 50%. Therefore, it shows that to achieve a higher acceleration for a rocket, it is very crucial to select a propellant and rocket structure that can efficiently minimize burn time. Also, the weight of the propellant, as well as the overall structure has a negligible (around 2%) effect on acceleration.



**(a)** **(b)**

**Figure 3:** Design changes for $a_{max}$ variation

4) **Design changes with SM variation:** The comparison shows that changes in static margin or factor of safety are not significantly influenced by the burn time, rocket velocity (Fig. 4a) and mass of the propellant or overall structure, however, the center of pressure and center of gravity are impacted by the increase in SM, by around 40% and 35% respectively.

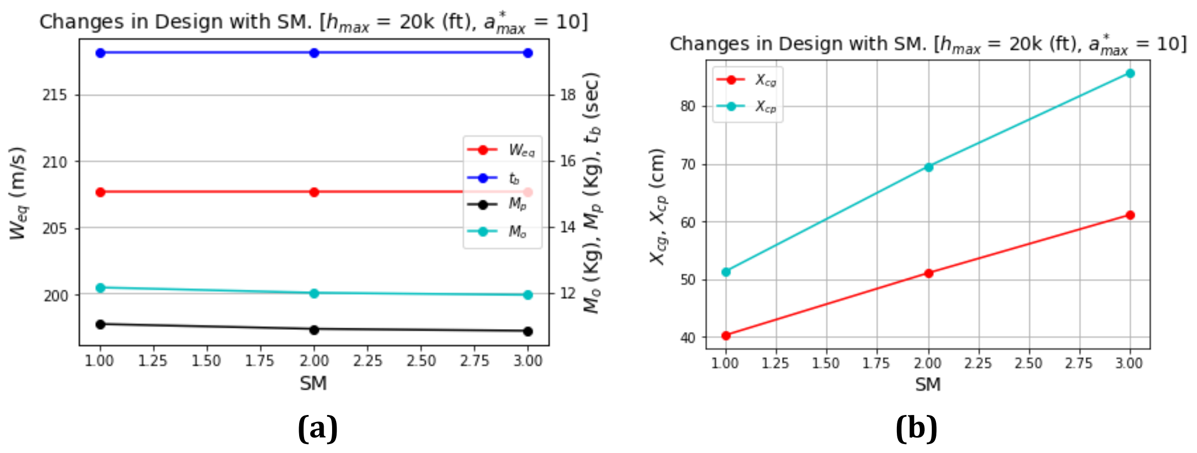
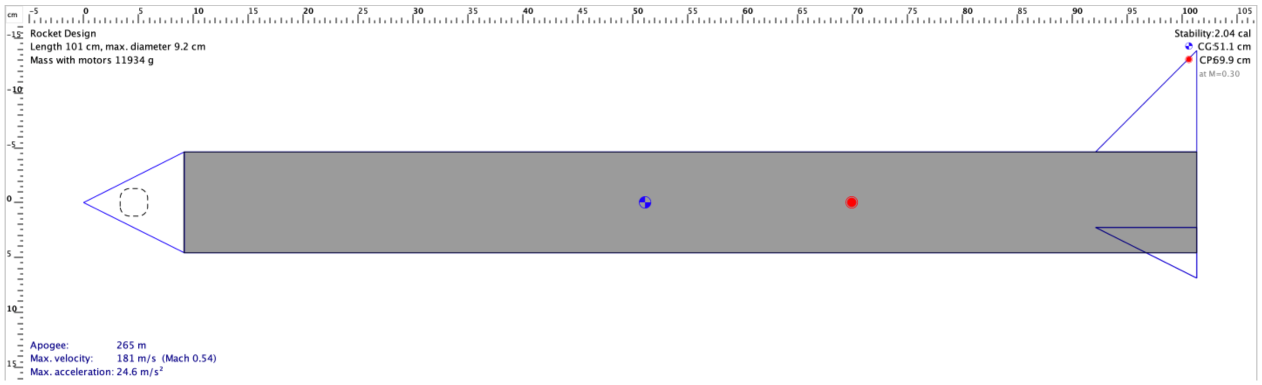


**Figure 4:** Design changes for $SM$ variation

## Part II

The designed rocket was simulated with the baseline condition $h_{\max} = 20{,}000\,ft, a^*_{\max} = 10, SM = 2$

1) $\boldsymbol{X_{CP}}$ ***and*** $\boldsymbol{X_{CG}}$ **Comparison:** The simulation result can be seen from the screenshot below, which fully agrees with the calculated values,$X_{CP} = 69.46\ cm, X_{CG} = 51.02\ cm$(table 3):



Rocket Design
Length 101 cm, max. diameter 9.2 cm
Mass with motors 11934 g

Stability:2.04 cal
CG:51.1 cm
CP:69.9 cm
at M=0.30

Apogee: 265 m
Max. velocity: 181 m/s (Mach 0.54)
Max. acceleration: 24.6 m/s²

**Figure 5:** $X_{CP}$ and $X_{CG}$ values from OpenRocket design.

So, the error between the calculated and simulated values is less than 0.5%.

2) **Ideal Vertical Case:** For linear burn rate, $I_{sp} = constant$. In an ideal situation we neglect the drag effect, which allows assuming ideal case with maximum efficiencies. So, the expected performance is much higher than the realistic condition.
For idealized case, the $h_{max}, V_{r,max}\ and\ a^*_{max}$ , with respect to the flight time is shown in the Fig.3 below:

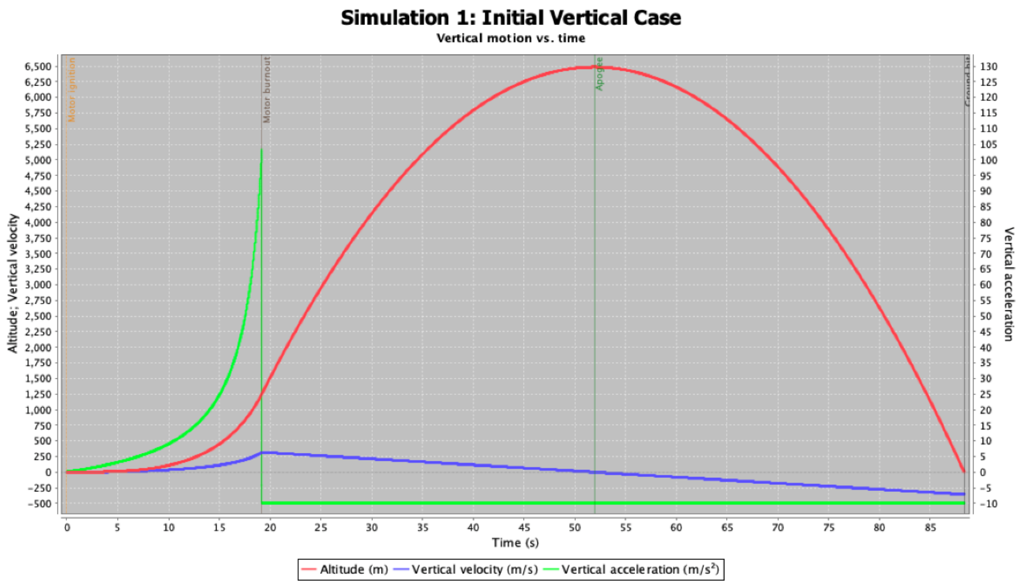


**Figure 6:** Rocket performance profile for ideal case ($C_D = 0$).

The simulation agrees to the calculated value of with 6% error in maximum height and 3% in acceleration. The rocket also satisfies the baseline conditions with approximately 7% error. This result makes sense as the performance profile agrees with the theoretical height, velocity, and acceleration values.

3) **Drag Effects:** Figure 7 below shows the realistic performance profile for the rocket under the drag effects. If compared to Fig. 6, the results show a significant decrease in performance as expected. This result because the effect of drag is very impactful in rocket performance as opposed to ideal conditions with no drag. So, with the addition of drag effects, the rocket with same design shows around 74% performance decay from the baseline condition under ideal case consideration.

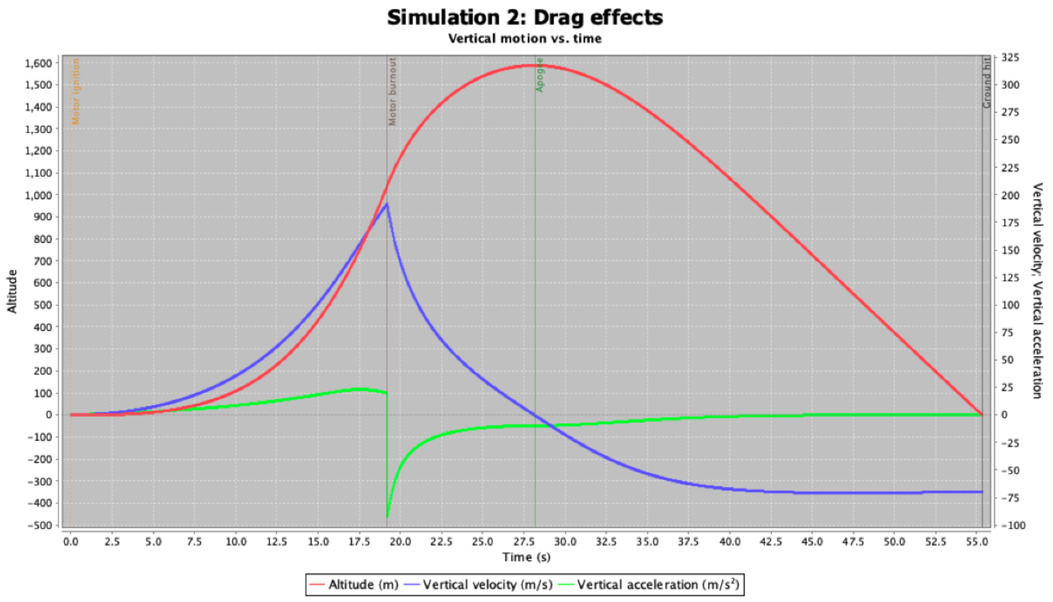


**Figure 7:** Rocket performance profile under drag effects.

This result makes sense because, realistically rockets can have around 75% efficiency, and the simulation results show a similar result.

4) **Geometry Selection**: Under the effect of drag, changing the geometry doesn't get to the closest performance as under ideal case. However, changing the geometry does help for better performance. In this case, changing the nose cone shape from conical to ogive helped the rocket to attain a higher altitude using the same amount of propellant by approximately 16%, as compared to the performance profile in Fig. 7.

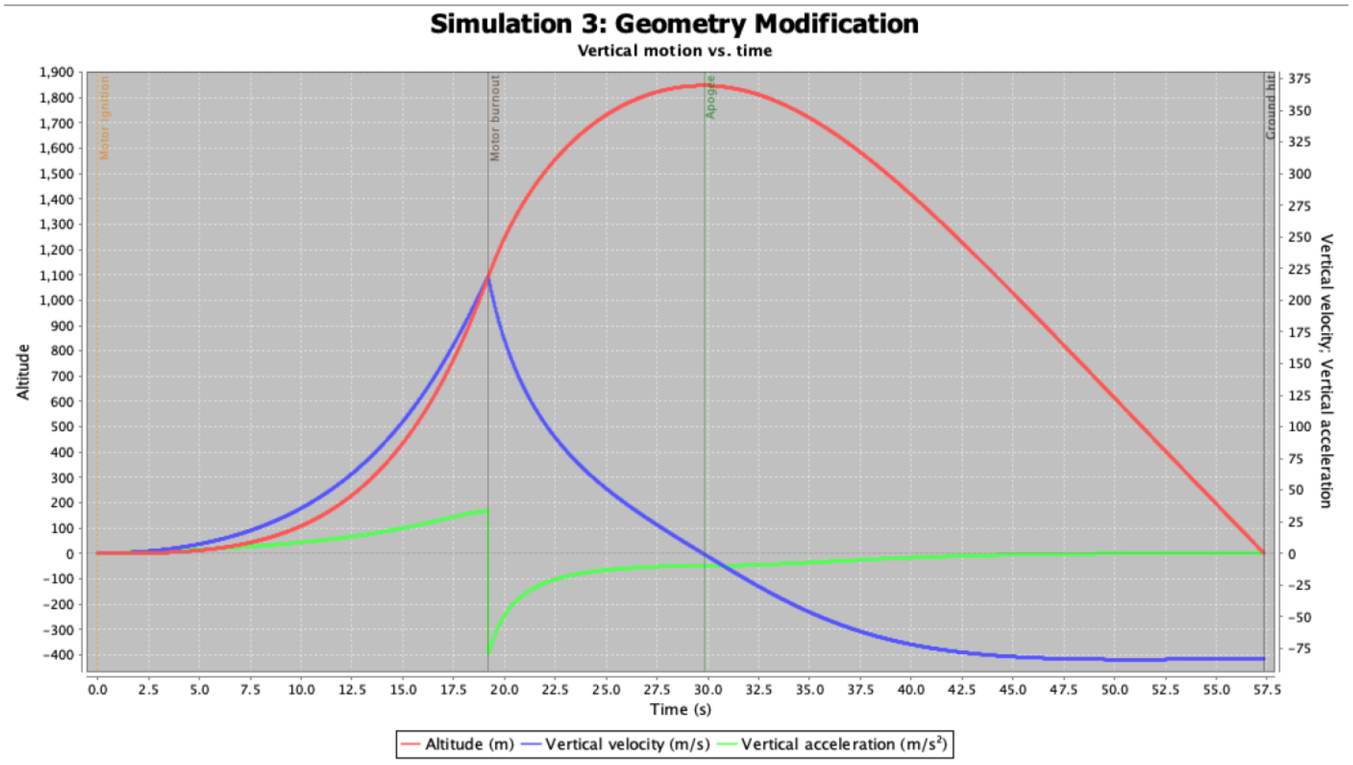


**Figure 8:** Rocket performance profile under geometry modification.

1) **Material Selection**: Changing the structural material does not appear to have a strong influence in increasing the rocket performance profile as it can be seen from the comparison between Fig.7 and Fig. 8. Here, the change of structural material from Aluminum (density 2.7 g/cm$^3$) to Polycarbonate (density 1.2 g/cm$^3$) increases the efficiency of only around 0.1%. Polycarbonate has a tensile strength of 65 MPa, a little stronger than Aluminum, that is used previously, which maintains the strength requirement of rocket design. Therefore, the result shows that making light-weight rocket is not the most effective way to increase the efficiency of the rocket.

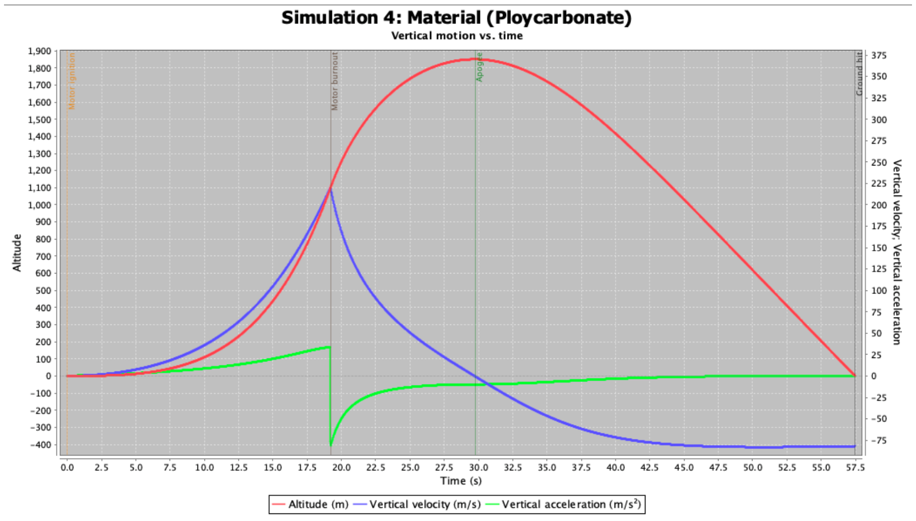


**Figure 9:** Rocket performance profile under material modification.

2) **Propellant Selection:** Figure 10 shows increasing the burn-time only by second can increase the highest altitude and velocity by 6%. In the Fig. 10, the burn time was increased to 20.2 second from 19.2 seconds as illustrated in Fig. 9. Therefore, it is evident that by selecting an alternative burn-profile, the rocket performance can be enhanced efficiently.

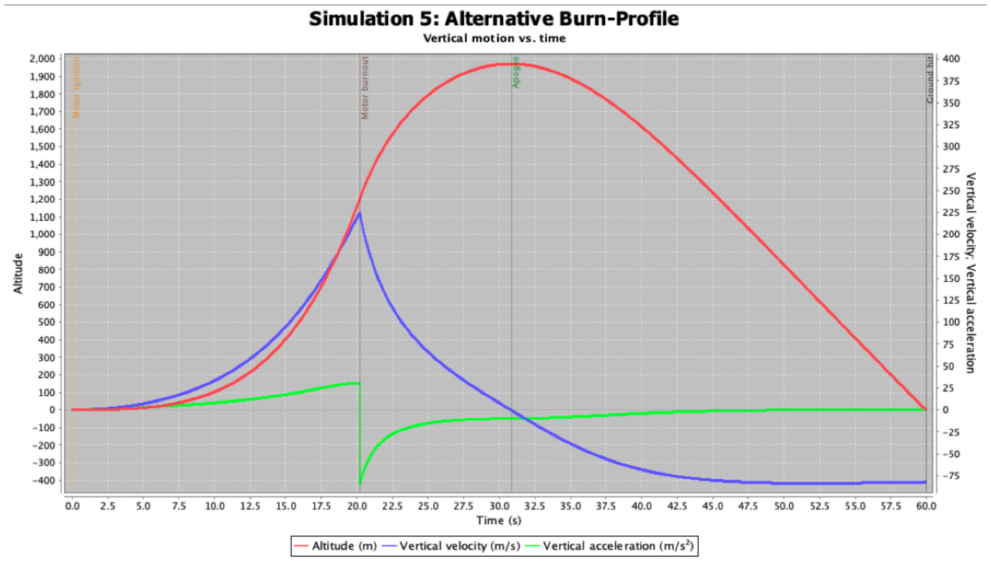


**Figure 10:** Rocket performance profile under

The summary of the simulations is also attached in the appendix [] which tabulates the effect of different simulations. Comparing the table values for each case, the above explained effects can be more evident.

## Summary and Conclusion:

The overall project helped us to learn the aspects of theoretical rocket design and the realistic rocket performance with the environmental and structural limitation involved as opposed to the performance under ideal environment. It helps us to understand how the practical scenarios do not always greatly agree with the theory. There are multiple sources of loss involved in reality. For example, the parts of rocket simulation where the non-idealized case was studied with the addition of drag force ($C_D \neq 0$), the rocket performance result showed about 72% deviation in from ideal case with no drag force involved. From case studies, we know that rockets usually have around 75% efficiencies. So, the project overall gave us the complete overview of rocket design and performance.

## Reference:

[1] Paul Desjardin. "Class Lecture - MAE 423: Introduction to Propulsion, Spring 2020." https://buffalo.app.box.com/s/8om64mmkqq3qflveaxufp90ofymasz17. Accessed 12 May 2020.

[2] James Barrowman. "Calculating the Center of Pressure of a Model Rocket." http://www.rockets4schools.org/images/Calculating.pdf. Accessed 12 May 2020.

## Appendices:

### (a) Python Code:

```python
#====================================================
# Project: Rocket Design and Fly (MAE 423)
# Author: Ayesha Khatun
# Python: 3.7.7
# Submission Date: 05/12/2020
#====================================================

# Required libraries
import numpy as np
from scipy import optimize
import matplotlib.pyplot as plt

### Parameter Description
# M_L = Payload mass
# h_max = Maximum Altitude
# a_star_max = Normalized max acceleration
# SM = static margin
# rho_s = shell density (aluminum)
# rho_p = Propellant density
# sigma_s = Shell working stress
# N = Number of fins

# g = acceleration due to gravity
# P_a = Atmospheric Pressure at sea level (Pa)
# gamma = Ratio of the specific heats (C_p/C_v)
# T_a = Atmospheric Temperature(K)
# R_gas = gas constant


# Given input values in the project
M_L = 1
rho_s = 2700
rho_p = 1772
sigma_s = 60e6
N = 3

g = 9.81
P_a = 101325.
gamma = 1.4
T_a = 298.
R_gas = 287.

# 'M_func', 'Xcp_func', and 'Xcg_func' functions calculate
# M, X_cp, and X_cg of Rocket's Nose, Main Tube, Fin Tube and Fins respectively.
def M_func(delta, rho_s, D, L, N):
    M_nose = rho_s*delta*np.pi*(1 + np.sqrt(5/4))* D**2
    M_maintube = rho_s*delta*np.pi*D*L
    M_fintube = rho_s*delta*np.pi*D**2
    M_fin = 0.5*N*delta*rho_s*D**2

    return np.array([M_nose, M_maintube, M_fintube, M_fin])


def Xcp_func(D, L):
    Xcp_nose = D*2./3.
    Xcp_maintube = D + L/2.
    Xcp_fintube = D + L + D/2.
    Xcp_fin = (D*3)/2 + L

    return np.array([Xcp_nose, Xcp_maintube, Xcp_fintube, Xcp_fin])


def Xcg_func(D, L):
    Xcg_nose = D*2./3.
    Xcg_maintube = D + L/2.
    Xcg_fintube = D + L + D/2.
    Xcg_fin = D + L + 2.*D/3.

    return np.array([Xcg_nose, Xcg_maintube, Xcg_fintube, Xcg_fin])

# 'L_func' is used to check if the condition ('dif') satisfies to get L in the algorithm.
def L_func(L, delta, rho_s, D, N, SM, R):
    M_vec_tmp = M_func(delta, rho_s, D, L, N)
    Xcp_vec_tmp = Xcp_func(D, L)
    Xcg_vec_tmp = Xcg_func(D,L)

    M_s_tmp = sum(M_vec_tmp)
    M_p_tmp = (R - 1)*(M_s_tmp + M_L)
    L_p_tmp = M_p_tmp/((np.pi * rho_p * D**2)/4.)
    X_cp_tmp = (sum(Xcp_vec_tmp*Cp_vec))/sum(Cp_vec)
    X_cg_tmp = ((sum(Xcg_vec_tmp*M_vec_tmp) + (.5*D*M_L) + (2*D + L - L_p_tmp/2)*M_p_tmp))/(su
m(M_vec_tmp) + M_L + M_p_tmp)

    dif = X_cp_tmp - X_cg_tmp - D*SM
    return (dif)

```

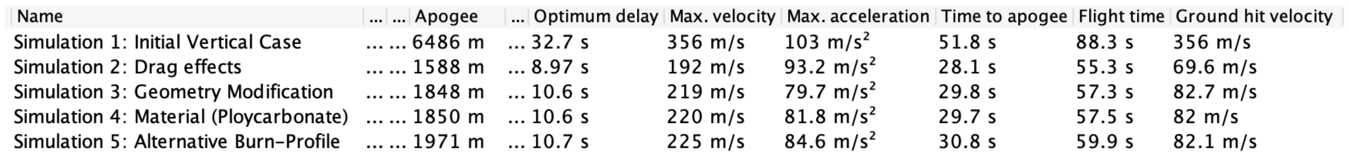
```python
# calculated from the handout
Cp_vec = np.array([2,0,0,4.6384])

def D_func(D, P_o, R, SM, h_max, a_star_max, W_eq, M_eq, t_b):
    delta = D*P_o/2/sigma_s

    M_nose = delta*rho_s*np.pi*(1+np.sqrt(5/4))* D**2
    M_fintube = delta*rho_s*np.pi* D**2
    M_fin = 0.5*N*delta*rho_s* D**2
    L_lower = ((R - 1)*(M_L + M_nose + M_fin + M_fintube) - np.pi* D**3 *rho_p/4.)/(np.pi* D**2 *rho_p/4. - (R - 1)*np.pi*D*rho_s*delta)
    L_upper = 1000.*D
    sol_L_lower = L_func(L_lower, delta, rho_s, D, N, SM, R)
    sol_L_upper = L_func(L_upper, delta, rho_s, D, N, SM, R)
    sol_prod = sol_L_lower*sol_L_upper

           parameter_list = ["D", "L", "L_p", "M_s", "M_p", "M_o", "X_cp", "X_cg", "delta", "h_max", "a_star_max", "SM", "W_eq", "M_eq", "t_b"]
           if (sol_prod<0):
               L = optimize.brentq(L_func, L_lower, L_upper, args=(delta, rho_s, D, N, SM, R))
               M_vec = M_func(delta, rho_s, D, L, N)
               Xcp_vec = Xcp_func(D, L)
               Xcg_vec = Xcg_func(D,L)

               M_s = sum(M_vec)
               M_p = (R-1)*(M_s+M_L)
               L_p = M_p/((np.pi * rho_p * D**2)/4.)
               X_cp = (sum(Xcp_vec*Cp_vec))/sum(Cp_vec)
               X_cg = ((sum(Xcg_vec*M_vec)+(.5*D*M_L)+(2*D+L-L_p/2)*M_p))/(sum(M_vec)+M_L+M_p)

               M_o = M_s+ M_p+ M_L

               out_array = np.array([D, L, L_p, M_s, M_p, M_o, X_cp, X_cg, delta, h_max, a_star_max, SM, W_eq, M_eq, t_b])
               output = dict(zip(parameter_list, out_array))

           else:
               out_array = np.zeros(len(parameter_list))
               output = dict(zip(parameter_list, out_array))

           return output

       ## The following inputs changes; here median inputs are used.
       # h_max = 10k, 20k, 30k ft
       # a_star_max = 5, 10, 20
       # SM = 1, 2, 3
       h_max = 20000 * 0.3048
       a_star_max = 10
       SM = 2

       R = a_star_max + 1.
       W_eq = np.sqrt((h_max * g)/(((np.log(R)/2) * (np.log(R) - 2)) + ((R - 1)/R)))
       a = np.sqrt(gamma*R_gas*T_a)
       M_eq = W_eq/a
       P_o = P_a*(1. + (((gamma-1.)/2.)*M_eq*M_eq)**(gamma/(gamma-1.)))
       t_b = ((R - 1) * W_eq)/(g * R)

       D_values = np.arange(0.01, .201, .00005)
       lmbda = np.zeros(len(D_values))
```

```python
        Output_list = []
        i=0
        for D in D_values:
            Output = D_func(D, P_o, R, SM, h_max, a_star_max, W_eq, M_eq, t_b)
            Output_list.append(Output)

            if Output["M_s"] == 0.:
                lmbda[i] = 0.
            else:
                lmbda[i] = M_L/(Output["M_s"]+Output["M_p"])
            i += 1

        lmbda_max = np.max(lmbda)
        index = np.where(lmbda==lmbda_max)[0][0]
        Final_output = Output_list[index]
        Final_output["lmbda"] = lmbda_max
        Final_output["epsilon"] = Final_output["M_s"]/(Final_output["M_p"]+Final_output["M_s"])

        print('Part I: (1): Summary Table.')
        print('---------------------------')
        print(' ')
        print("---------------------------------***INPUT***------------------------------------")
        print(f'h_max      = {h_max/.3048:.1f} ft')
        print(f'a_star_max = {a_star_max:.1f}')
        print(f'SM         = {SM}')
        print(' ')
        print('-----------------------------------***OUTPUT***---------------------------------')
        print(f'R          = {R}')
        print(f'W_eq       = {W_eq:.1f} m/s')
        print(f't_b        = {t_b:.1f} sec')
        print(f'p_o        = {P_o/1000000:.3f} MPa')
        print(f'delta/D    = {(Final_output["delta"]/Final_output["D"]):.5f}')
        print(f'D          = {Final_output["D"]:.4f} m')
        print(f'L          = {Final_output["L"]:.4f} m')
        print(f'X_cg       = {(100*Final_output["X_cg"]):.2f} cm')
        print(f'X_cp       = {(100*Final_output["X_cp"]):.2f} cm')
        print(f'M_s        = {(Final_output["M_s"]):.3f} kg')
        print(f'M_p        = {(Final_output["M_p"]):.2f} kg')
        print(f'M_o        = {(Final_output["M_o"]):.2f} kg')
        print(f'Lambda     = {(Final_output["lmbda"]):.4f}')
        print(f'Epsilon    = {(Final_output["epsilon"]):.4f}')


        ### Part I: (2,3,4): Changes in Design with h_max, a_star_max and SM

        # 'plot_data_func' generates the data for comparison for a given parameter variation,choosing
        # the “median option” for the other two parameters, resulting in 9 design cases in total.
        h_values = np.arange(10000, 30001, 2000)
        a_values = np.arange(5,21,1.)
        SM_values = np.array([1.,2.,3.])

        def plot_data_func(values, hmax, amax, SM):
            data = []
            for val in values:

                if hmax:
```

```
                h_max = val * 0.3048
                a_star_max = 10
                SM = 2
            elif amax:
                h_max = 20000 * 0.3048
                a_star_max = val
                SM = 2
            elif SM:
                h_max = 20000 * 0.3048
                a_star_max = 10
                SM = val

            R = a_star_max + 1.
            W_eq = np.sqrt((h_max * g)/(((np.log(R)/2) * (np.log(R) - 2)) + ((R - 1)/R)))
            a = np.sqrt(gamma*R_gas*T_a)
            M_eq = W_eq/a
            P_o = P_a*(1. + (((gamma-1.)/2.)*M_eq*M_eq)**(gamma/(gamma-1.)))
            t_b = ((R - 1) * W_eq)/(g * R)
            D_values = np.arange(0.01, .201, .00005)
            lmbda = np.zeros(len(D_values))

            Output_list = []
            i=0
            for D in D_values:
                Output = D_func(D, P_o, R, SM, h_max, a_star_max, W_eq, M_eq, t_b)
                Output_list.append(Output)

                if Output["M_s"] == 0.:
                    lmbda[i] = 0.
                else:
                    lmbda[i] = M_L/(Output["M_s"]+Output["M_p"])
                i += 1

            lmbda_max = np.max(lmbda)
            index = np.where(lmbda==lmbda_max)[0][0]
            Final_out = Output_list[index]
            Final_out["lmbda"] = lmbda_max
            Final_out["epsilon"] = Final_out["M_s"]/(Final_out["M_p"]+Final_out["M_s"])
            data.append(Final_out)

        return data

    # 'h_data' for changes in design with h_max;
    # 'a_data' for changes in design with a_star_max;
    # 'SM_data' for changes in design with SM;
    h_data = plot_data_func(values=h_values, hmax=True, amax=False, SM=False)
    a_data = plot_data_func(values=a_values, hmax=False, amax=True, SM=False)
    SM_data = plot_data_func(values=SM_values, hmax=False, amax=False, SM=True)

    # 'plot_func' generates plots for changes in design with h_max, a_start_max, and SM.
    def plot_func(plot_data, title, xlabel, hmax, amax, SM):

        if hmax:
            h_max = [sub['h_max'] for sub in plot_data]
            h_max = np.rint(np.multiply(h_max, 3.28084))
            x = h_max
        elif amax:
            a_star_max = [sub['a_star_max'] for sub in plot_data]
            x = a_star_max
        elif SM:
            SM = [sub['SM'] for sub in plot_data]
```

```python
            x = SM

        W_eq = [sub['W_eq'] for sub in plot_data] # in m/s
        t_b = [sub['t_b'] for sub in plot_data] # in sec
        M_p = [sub['M_p'] for sub in plot_data] # in Kg
        M_o = [sub['M_o'] for sub in plot_data] # in Kg
        X_cg = np.multiply([sub['X_cg'] for sub in plot_data], 100) # in cm
        X_cp = np.multiply([sub['X_cp'] for sub in plot_data], 100) # in cm

        ### Plot W_eq, t_b, M_p, M_o
        fig, ax = plt.subplots()
        plt.title(title, size = 14)
        plt.plot(x, W_eq, '-or', label='$W_{eq}$')
        ax.set_xlabel(xlabel, size = 14)
        ax.set_ylabel('$W_{eq}$ (m/s)', size = 14)

        # Get second Y axis
        ax2 = ax.twinx()
        plt.plot(x, t_b, '-ob', label='$t_b$')
        plt.plot(x, M_p, '-ok', label='$M_p$')
        plt.plot(x, M_o, '-oc', label='$M_o$')
        ax2.set_ylabel('$M_o$ (Kg), $M_p$ (Kg), $t_b$ (sec)', size=14)
        handles,labels = [],[]
        for ax in fig.axes:
            for h,l in zip(*ax.get_legend_handles_labels()):
                handles.append(h)
                labels.append(l)

        plt.legend(handles,labels)
        plt.grid()
        plt.show()

        ### plot X_cg and X_cp
        fig, ax = plt.subplots(1,1)
        plt.title(title, size = 14)
        plt.plot(x, X_cg, '-or', label='$X_{cg}$')
        plt.plot(x, X_cp, '-oc', label='$X_{cp}$')
        ax.set_xlabel(xlabel, size = 14)
        ax.set_ylabel('$X_{cg}$, $X_{cp}$ (cm)', size = 14)
        plt.legend()
        plt.grid()
        plt.show()

    print('Part I: (2): Changes in Design with h_max')
    plot_func(plot_data = h_data, title = 'Changes in Design with $h_{max}$. [$a^*_{max}$ = 10, SM = 2]', xlabel = '$h_{max}$ (ft)', hmax=True, amax=False, SM=False)

    print('Part I: (3): Changes in Design with a_star_max')
    plot_func(plot_data = a_data, title = 'Changes in Design with $a^*_{max}$. [$h_{max}$ = 20k (ft), SM = 2]', xlabel = '$a^*_{max}$', hmax=False, amax=True, SM=False)

    print('Part I: (4): Changes in Design with SM.')
    plot_func(plot_data = SM_data, title = 'Changes in Design with SM. [$h_{max}$ = 20k (ft), $a^*_{max}$ = 10]', xlabel = 'SM', hmax=False, amax=False, SM=True)
```

## b) Simulation Comparison:



| Name | … | … | Apogee | … | Optimum delay | Max. velocity | Max. acceleration | Time to apogee | Flight time | Ground hit velocity |
|---|---|---|---|---|---|---|---|---|---|---|
| Simulation 1: Initial Vertical Case | … | … | 6486 m | … | 32.7 s | 356 m/s | 103 $m/s^2$ | 51.8 s | 88.3 s | 356 m/s |
| Simulation 2: Drag effects | … | … | 1588 m | … | 8.97 s | 192 m/s | 93.2 $m/s^2$ | 28.1 s | 55.3 s | 69.6 m/s |
| Simulation 3: Geometry Modification | … | … | 1848 m | … | 10.6 s | 219 m/s | 79.7 $m/s^2$ | 29.8 s | 57.3 s | 82.7 m/s |
| Simulation 4: Material (Ploycarbonate) | … | … | 1850 m | … | 10.6 s | 220 m/s | 81.8 $m/s^2$ | 29.7 s | 57.5 s | 82 m/s |
| Simulation 5: Alternative Burn-Profile | … | … | 1971 m | … | 10.7 s | 225 m/s | 84.6 $m/s^2$ | 30.8 s | 59.9 s | 82.1 m/s |

**Figure 11:** Rocket simulation summary.