Overview

Rev 0.0

Tim M. Kostersitz

February 2021

The Model

In this project I decided to model a three stage rocket. The model contains the option to either eject the empty stage cells, or to keep carrying them with it. Mechanically this is not questioned, the assumption is that the previous stage does not interfere with the thrust of the next. As for the data that can be visualized, the model will produce data points for thrust, the force due to gravity, the speed of the rocket, and its height. All of these are plotted in one form or another in the following sections.

1 Functions

1.1 Mass

The mass function is to return a value for the mass of the rocket at a given time 't'. When I first wrote the program I went with a function that is capable of taking in a array. This turned out to be impractical for the use with the 'odeint' function seeing as that in itself passes through a time step to the subsequent function. So now this function is much simpler and only calculates the mass at a single time step. This model takes into account air resistance as a function of height. It also contains the option for the stage ejection:

`mass(t, 0)` - The stages are not ejected

`mass(t, 1)` - The stages are ejected

The results from stage ejection are shown below.

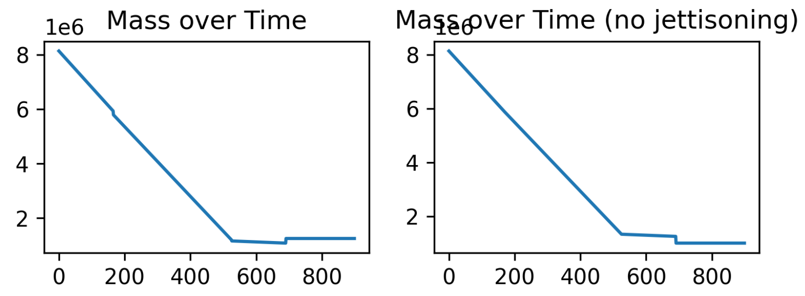

Figure 1: Mass in Kg where the left shows the result of a simulation with stage ejection and the right shows one without

From the figure it is shown that the empty shell ejection of each previous stage does not have much effect on the mass of the rocket.

1.2 temperature

This function returns the ambient temperature at a given altitude. This is used to compute the air resistance. I could only find a basic atmospheric shell model that assumes a linear change in temperature within a given atmospheric layer. Then, at any altitude above the 84,852m mark I simply assume 120K which is near the dark side temperature of the ISS. After 600,000m I assume 2.4K. I had a hard time finding a great way to model this, but thought it would be cool to try it this way since this function is not used for the major functionality of the code. The simulation results of a 15min simulation is found below.

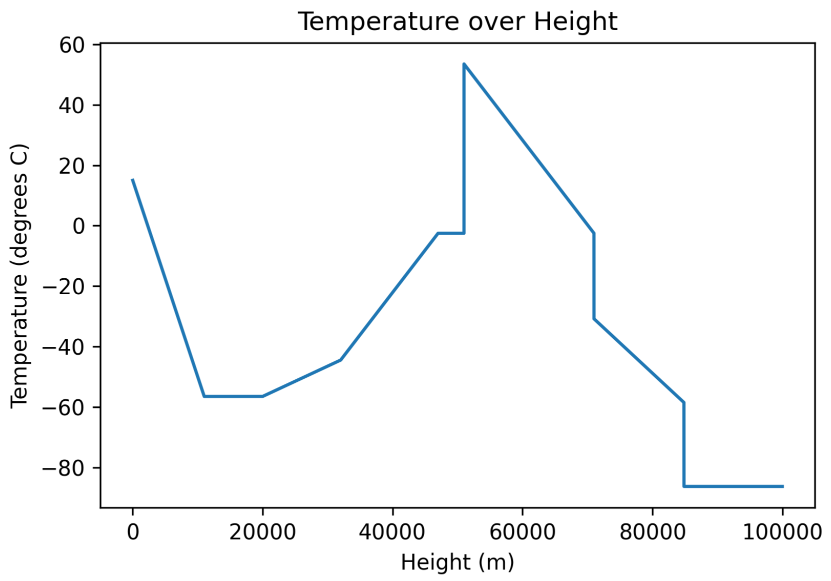

Figure 2: Temperature plotted as the height increases

The formula used for this computation was:

$$T(h) = T_b + \alpha_0(h - 0) \tag{1}$$

where where h is the height, T_b is a reference temperature and α is another variable given by the *International Standard Atmosphere* which is a mathematical model that splits the atmosphere into various layers that have defined properties. Within the layers this model assumes linear distribution of absolute temperature. From the given properties one can calculate various other properties since within the layers. I followed the assumptions that are made with this model to model all of the atmosphere related quantities. More information on how I came about these calculations is found at this webpage.

1.3 airDensity

This function returns the density of air at a given height. This also takes into account the layer of the atmosphere. The result of this function as well as the formula followed are shown below.

$$\rho = \frac{P}{(R_{sp} * T(h))} \tag{2}$$

Where

$$P = P_b * \frac{T_b}{T(h)})^{\frac{g_0 M}{R L_b}} \tag{3}$$

In this context P_b is the base static pressure of the layer b in units of Pa. T_b is the base temperature of the layer b in K. L_b is the base temperature lapse rate of the layer b in K/m. And H_b is the base geopotential height of the layer b in meters. These were all provided on this webpage.

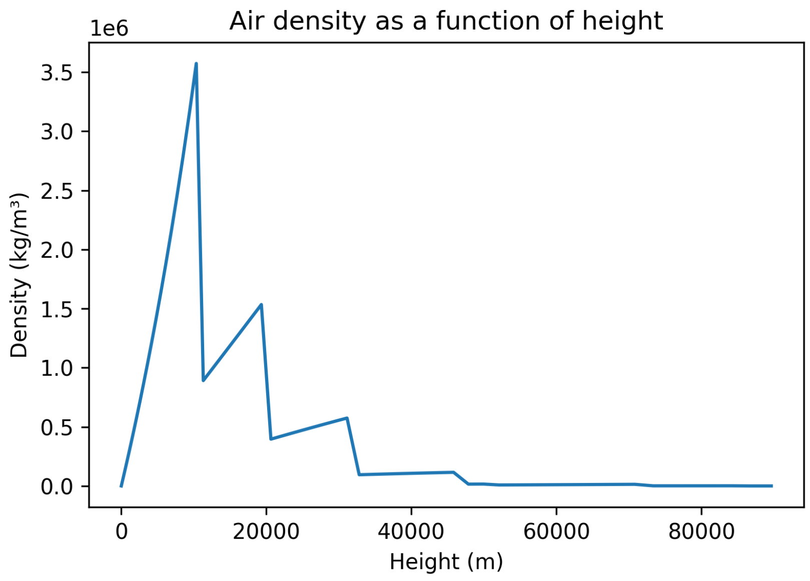

Figure 3: Density of air plotted as the height increases

1.4 airResistanceForce

This function return the force on the rocket due to air resistance (in N). This is done by using the 'airDensity' and 'airTemperature' function presented in the previous sections. The result of this function and the formula used are shown below.

$$\text{Air Resistance} = \frac{1}{2}C\rho A v^2 \tag{4}$$

here C, which is the drag coefficient, is assumed to be 0.5, ρ is the air density as given by the previous 'airDensity' function.

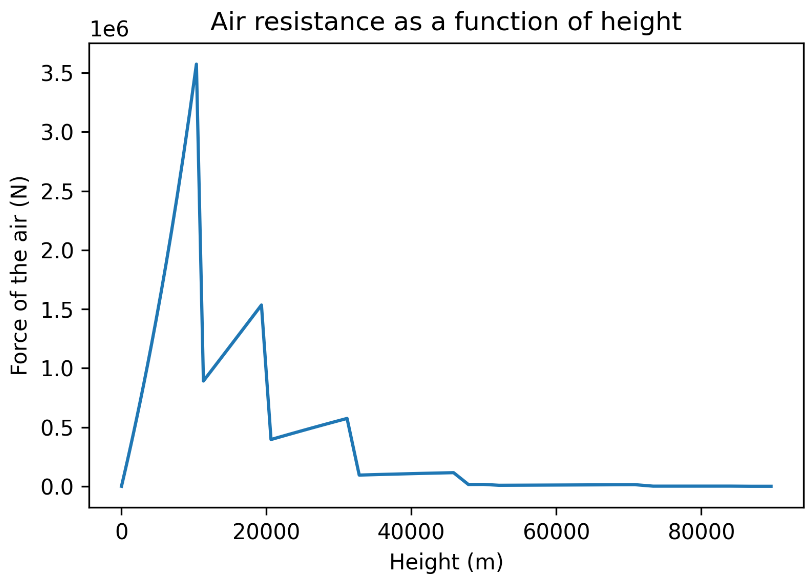

Figure 4: Air resistance plotted as the height increases

1.5 vExhaust

This function returns the exhaust velocity of the rocket that is calculated from the fuel burn rate and thrust which is converted from the given Kgf to Newtons within the function. The result of this function and its corresponding formula are shown below.

$$v_{exhaust} = \frac{thrust_{stagex} 9.80665}{BurnRate_{stagex}} \tag{5}$$

The burn rates are calculated by dividing the thrust by its burn time. The 9.80665 are the conversion from kg force to Newtons.

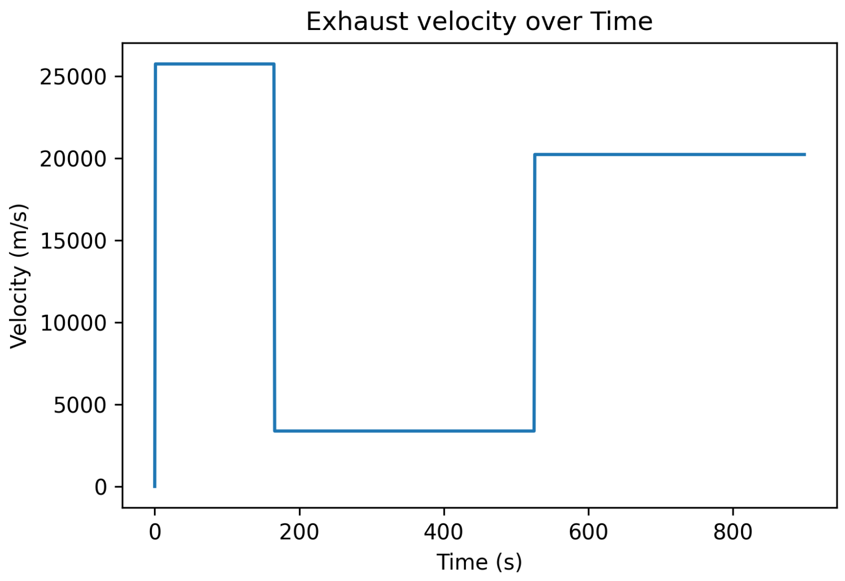

Figure 5: Exhaust velocity as calculated from the burn rate and thrust

1.6 forces

This function returns the net force on the rocket. It takes into account the force due to gravity, air resistance and thrust of the rocket engine. I found this plot to be one of the coolest out of the bunch as it utilizes most of the helper functions and results in an intuitively understandable plot. The sum of the forces is given by:

$$F = \frac{GM_eM_r(t)}{(r_e + h)^2} + F_{airResistance} - thrust_{stagex}9.80665 \tag{6}$$

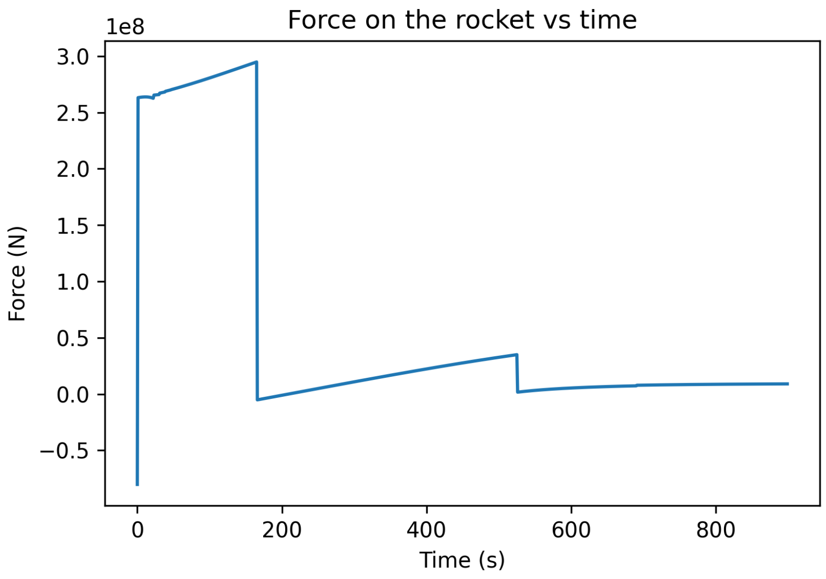

Figure 6: Net force on the rocket over time

As the fuel burns off weight, the rocket begins to see a greater upward force which results in the positive slope seen on each segment of the plot. Each segment or bar is a respective stage, the largest one being the initial burn.

1.7 velocity

This function returns the velocity of the rocket as calculated with the Tsiolkovsky rocket equation using the exhaust velocity. This function is also put through the 'odeint' function to produce altitude data. This is where my simulation becomes a bit discontinuous due to the fact that the rocket velocity equation does not take into account the air Resistance. While this was my goal starting out, it is also true that air resistance becomes quite negligible very quickly with the chosen rocket parameters.

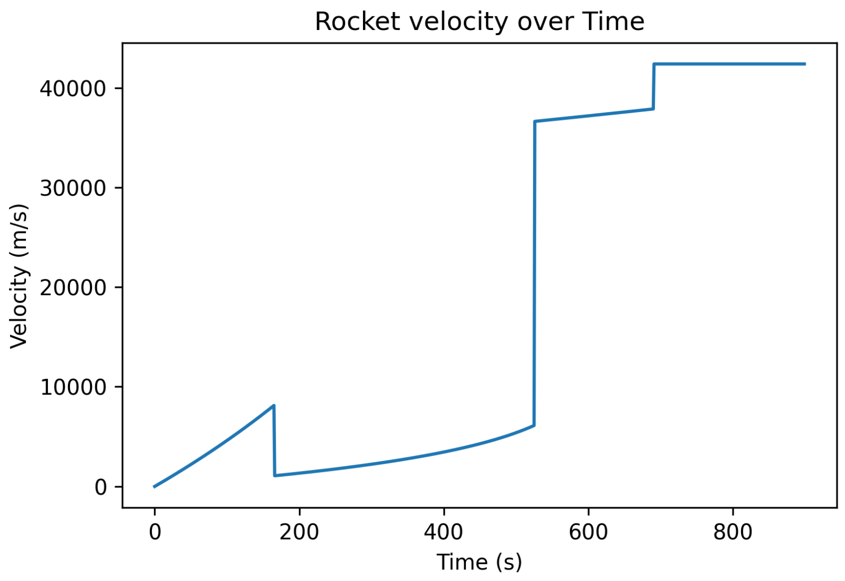

Figure 7: Rocket velocity as calculated from the burn rate and thrust

As seen above there is a dip in velocity right after the first stage separation. I was not able to figure out the reason for this dip. I do believe that this could make sense if the Tsiolkovsky equation was derived with gravitation in mind.

2 Conclusion

Up until I noticed that dip in velocity I was feeling extremely confident that my program was working, but this looks to me like a mistake in my math or understanding. Nonetheless I thought that the modeling of the atmosphere turned out extremely interesting and learning about the modeling of it was an adventure in and of itself.