Induction Challenge Maths

Monash High-Powered Rocketry

Mechanical & Aerospace Engineering
Monash University
Australia
August 6, 2018

Contents

1 Introduction **2**

2 Drag **2**

3 Thrust **3**

4 Mass **3**

5 Gravitation (≠ Gravity) **3**

6 Integration **3**

1 Introduction

To give everyone a better appreciation of how simulations work from a holistic perspective, you will be creating your own 1 degree-of-freedom (DOF) rocket simulation. This document describes all of the maths you will need for this task, with a few artificial functions included to avoid having to interpolate from real data.

Your aim in this challenge is to create a simulation that is both **accurate** and **efficient**. Prizes will be awarded to those whose simulations are either the fastest or the most accurate. Feel free to include your own custom equations in lieu of those described in this document to improve the accuracy of this simulation.

2 Drag

We will model drag with the (hopefully familiar?) equation

$$D = \frac{1}{2}\rho V^2 S C_D \tag{1}$$

where you can approximate the wetted area by the rocket's diameter of $D = 98\ mm$ and length of $L = 2.5\ m$. The coefficient of drag is a function of Mach number is given by the following (completely made up) equation

$$C_D(M) = e^{-1.2M} \sin M + \frac{M}{6} \log_{10}(M + 1) \tag{2}$$

Density can be found using the state equation

$$p = \rho R T \tag{3}$$

combined with the standard atmosphere equations

$$T = T_0 - Lh \tag{4}$$

$$p = p_0 \left(\frac{T}{T_0}\right)^{\frac{-g_0}{LR}} \tag{5}$$

where $L = -0.0065\ Km^{-1}$ is the lapse rate, $g_0 = 9.80665 ms^{-2}$ is the mean gravitational acceleration at sea level, $R = 287.16$ is the specific gas constant for air, and $p_0 = 101325\ Pa$ and $T_0 = 288.15\ K$ are the standard sea level pressure and temperature, respectively.

You can calculate Mach number using the equation

$$M = \frac{V}{\sqrt{\gamma R T}} \tag{6}$$

where $\gamma = 1.4$ for air.

3 Thrust

Thrust will be modelled by another made up equation (although its profile approximates a real motor)

$$T(t) = T_0 \left(1 - 10^{-5} e^{\frac{ln(10^5)}{t_B} t}\right) \tag{7}$$

where we will use a peak thrust of $T_0 = 3000\ N$ and a burnout time of $t_B = 4.5\ s$. This equation mimics the behaviour of the AeroTech N3300 motor that will likely be used to launch to 30,000 ft.

4 Mass

As your rocket motor burns, the total mass of the rocket will fall. For simplicity, we will assume a quadratic mass burn rate while the motor is running.

$$\dot{m}_f = -\frac{m_f}{t_B} \tag{8}$$

The total mass of your rocket should be $m = 25\ kg$, which includes $m_f = 9\ kg$ of fuel. This is a rough estimate of the weight breakdown of a 30 000 ft rocket.

5 Gravitation ($\neq$ Gravity)

We will use a simple inverse square model of gravitation

$$g = \frac{GM}{(R_E + h)^2} \tag{9}$$

where the standard gravitational parameter $GM = 3.986 \times 10^{14}$ and the radius of the Earth can be assumed to be $R_E = 6378137\ m$ (independent of latitude).

6 Integration

Writing the integrator is one of the most challenging parts of the simulator. I will describe Euler's method here, however those of you who know more advanced techniques should use them!

We need to solve the 2nd ODE (Newton's law)

$$\frac{d^2 s}{dt^2} = F/m = f(t, s, v) \tag{10}$$

where F/m has been replaced by the general function $f(t, s, v)$ for brevity. Note that it is a function of time, position (altitude), and velocity. Take a moment to think about which forces require each of these variables. Do you think we need any more information? Don't forget that mass is not constant for a rocket!

To solve this ODE numerically, we need to reduce it to a pair of 1st order ODEs. This is done by substituting velocity, leaving us with the pair of equations

$$\frac{ds}{dt} = v \tag{11}$$

$$\frac{dv}{dt} = f(t, s, v) \tag{12}$$

The general formula for Euler's method is

$$y_{i+1} = y_i + dtf(t_i, y_i) \tag{13}$$

where the subscript represents the solution estimate (ie. it estimates the new state $i + 1$ using the current state i). To be more specific to our system of ODEs, we have

$$s_{i+1} = s_i + dt \cdot v_i \tag{14}$$

$$v_{i+1} = v_i + dt \cdot f(t_i, s_i, v_i) \tag{15}$$

Euler's method has a global truncation error of $O(dt)$, so you will need to decrease the step size in your simulation to obtain good results (although this will take longer).

As you can see from the maths above, your integrator function will require many inputs, and will have to store the new position and velocity before returning them. One way to do this is to pass a class instance to the function which contains all of the information it needs. You could use the *State* class for this.

If you have trouble with anything described above ask **questions!** This exercise should be challenging, but ultimately fun.