

# UNH SEDS Design Report

May 2018



Reilly Webb, Charlie Nitschelm, Thomas Collins,  
Dan Nemr, Nicholas Clegg, Kevin Bucher,  
Grace Johnston, Charles Gould, Ross Thyne, Ester Yee  
Matt Dodge, Pedro Campos, Silas Johnson, Travis Raynolds

Advisor: Doctor Todd Gross

## Table of Contents

|                                                               |           |
|---------------------------------------------------------------|-----------|
| <b>LIST OF FIGURES .....</b>                                  | <b>3</b>  |
| <b>LIST OF TABLES .....</b>                                   | <b>4</b>  |
| <b>LIST OF EQUATIONS.....</b>                                 | <b>4</b>  |
| <b>INTRODUCTION.....</b>                                      | <b>6</b>  |
| ABSTRACT .....                                                | 6         |
| OBJECTIVES OF UNH SEDS .....                                  | 6         |
| UNIVERSITY STUDENT ROCKETRY COMPETITION.....                  | 6         |
| <i>Goals</i> .....                                            | 7         |
| <i>Constraints</i> .....                                      | 7         |
| OVERALL ROCKET CONFIGURATION AND CONCEPT OF OPERATIONS .....  | 7         |
| LAUNCH OPERATIONS AND SAFETY PRECAUTION CHECKLIST.....        | 8         |
| <b>COMPARISON OF TEST RESULTS AND LAUNCH SIMULATIONS.....</b> | <b>20</b> |
| STATIC TEST FIRE RIG .....                                    | 20        |
| <i>Test Results</i> .....                                     | 20        |
| <i>Test Comparisons</i> .....                                 | 23        |
| FLIGHT TRAJECTORY .....                                       | 25        |
| <i>Aether II</i> .....                                        | 25        |
| <i>Further Testing</i> .....                                  | 27        |
| <b>REDESIGN DETAILS .....</b>                                 | <b>9</b>  |
| IMPROVEMENT CYCLE .....                                       | 9         |
| AERODYNAMIC MODELS .....                                      | 10        |
| <i>Assumptions</i> .....                                      | 11        |
| <i>Program Structure</i> .....                                | 12        |
| Stability .....                                               | 12        |
| Acceleration .....                                            | 14        |
| Thrust .....                                                  | 15        |
| Mass .....                                                    | 15        |
| Drag.....                                                     | 15        |
| Trajectory .....                                              | 17        |
| <i>Model Verification</i> .....                               | 18        |
| NONLINEAR OPTIMIZATION .....                                  | 20        |
| <i>Constraints</i> .....                                      | 29        |
| <i>Finite Element Analysis</i> .....                          | 29        |
| <b>ROCKET ITERATIONS .....</b>                                | <b>30</b> |
| MODEL ROCKET CLASS ITERATIONS.....                            | 30        |
| <i>Model Rocket v1</i> .....                                  | 30        |
| <i>Model Rocket v2</i> .....                                  | 30        |
| AETHER CLASS ITERATIONS .....                                 | 30        |
| <i>Aether I</i> .....                                         | 30        |
| <i>Aether II</i> .....                                        | 31        |
| <i>Aether III</i> .....                                       | 31        |
| <i>Aether IV</i> .....                                        | 31        |
| <i>Aether V</i> .....                                         | 32        |

|                                                             |           |
|-------------------------------------------------------------|-----------|
| CURRENT ITERATION (DETAILED).....                           | 32        |
| AETHER VI.....                                              | 32        |
| <b>RECOVERY SYSTEM.....</b>                                 | <b>34</b> |
| ONBOARD RECOVERY ELECTRONICS .....                          | 34        |
| PARACHUTE DESIGN.....                                       | 36        |
| <b>COMPETITION ROCKET AND FINAL ASSEMBLY PROCEDURE.....</b> | <b>39</b> |
| MATERIAL CHOICE .....                                       | 39        |
| ENGINE CHOICE.....                                          | 39        |
| PARACHUTE CHOICE.....                                       | 39        |
| FINAL ASSEMBLY PROCEDURE .....                              | 39        |
| LAUNCH DATES .....                                          | 41        |
| <b>BILL OF MATERIALS AND COST BREAKDOWN.....</b>            | <b>42</b> |
| OVERALL CASH FLOW .....                                     | 42        |
| DETAILED CASH OUTFLOW.....                                  | 42        |
| <b>APPENDIX.....</b>                                        | <b>43</b> |
| MATLAB SIMULATION CODE .....                                | 44        |
| <i>Overall Simulation Function.....</i>                     | <i>44</i> |
| Thrust .....                                                | 45        |
| Mass .....                                                  | 48        |
| Drag of Sustainer.....                                      | 49        |
| Drag of Booster and Sustainer.....                          | 50        |
| Acceleration of Sustainer .....                             | 54        |
| Acceleration of Booster.....                                | 55        |
| CALIBER FUNCTION CODE .....                                 | 55        |
| <b>REFERENCES.....</b>                                      | <b>43</b> |

## List of Figures

|                                                            |    |
|------------------------------------------------------------|----|
| Figure 1. Rocket Components.....                           | 7  |
| Figure 11. Improvement Cycle .....                         | 10 |
| Figure 12 Trapezoidal Fin Model.....                       | 13 |
| Figure 13. Aether IV Caliber .....                         | 14 |
| Figure 14. Aether IV Launch Simulation .....               | 18 |
| Figure 15: Model verification using Aether IV rocket ..... | 19 |
| Figure 2. STFR test of booster engine.....                 | 20 |
| Figure 3. Load cell calibration .....                      | 21 |
| Figure 4. Booster Engine Response.....                     | 22 |
| Figure 5. Sustainer Engine Response .....                  | 22 |

|                                                             |    |
|-------------------------------------------------------------|----|
| Figure 6. Engine Impulse vs Time .....                      | 23 |
| Figure 7. Experimental Data vs Cesaroni Supplied Data ..... | 24 |
| Figure 8. Aether 2 Trajectory Comparison .....              | 25 |
| Figure 9. Aether 2 Velocity Comparison.....                 | 26 |
| Figure 10. Aether 2 Stability.....                          | 27 |
| Figure 16. Optimization of Aether VI Dimensions .....       | 28 |
| Figure 17. Centering Ring FEA .....                         | 29 |
| Figure 18. Booster Fins .....                               | 33 |
| Figure 19. Sustainer Fins .....                             | 33 |
| Figure 20. Nose Cone .....                                  | 34 |
| Figure 21. Dual deployment basic design.....                | 35 |
| Figure 22. Electronics schematics of the TeleMega GPS.....  | 36 |
| Figure 23. Parachute Descent Speeds.....                    | 37 |

## List of Tables

|                                            |    |
|--------------------------------------------|----|
| Table 1. Surface Roughness.....            | 16 |
| Table 2. Aether IV Apogee Predictions..... | 19 |
| Table 3. Solid Textile Parachutes.....     | 38 |

## List of Equations

|                   |    |
|-------------------|----|
| Equation 1 .....  | 12 |
| Equation 2.....   | 12 |
| Equation 3 .....  | 13 |
| Equation 4.....   | 13 |
| Equation 5.....   | 14 |
| Equation 6.....   | 15 |
| Equation 7.....   | 15 |
| Equation 8.....   | 16 |
| Equation 9.....   | 16 |
| Equation 10.....  | 16 |
| Equation 11 ..... | 17 |

Equation 12..... 17

Equation 13..... 17

Equation 14..... 17

Equation 15..... 18

Equation 16..... 38

Equation 17..... 39

## Introduction

### Abstract

Members of UNH SEDS are designing, manufacturing, and launching a high powered multi-stage rocket for the SEDS University Student Rocketry Competition. Collegiate rocketry teams will be competing nationally in the Fall of 2018, with points awarded to the rockets that achieve the highest altitude, are fully recoverable, and are backed by the strongest design methodology. Since our team is working from the ground up, UNH SEDS has taken a "first principles" approach towards reaching these goals. Once fundamental aerodynamic relationships were studied and understood, they were implemented to create models of flight dynamics, drag, and stability. A static test fire rig was constructed to obtain experimental thrust curve data from the engines; further increasing the accuracy of our simulated trajectories. Driven by both manufacturing and competition constraints, our models were then used to optimize nose cone, body tube, and fin dimensions. Eight rocket iterations have been designed and launched. We have analyzed the flight data from each launch to continuously improve and learn important lessons out in the field that could not have been gathered from theory and simulations alone. The bulk of this report is the process in which we achieved precise simulation and optimization methods to apply to our competition rocket.

### Objectives of UNH SEDS

Students for the Exploration and Development of Space (SEDS) is a national, student-based organization that enables university students to get involved in space related projects. A chapter of SEDS has been founded at UNH in the Fall of 2017.

The mission of UNH SEDS is to provide a platform for UNH students to form multi-disciplinary teams and pursue space-focused outreach, networking events, and engineering projects.

This year's primary focus is to design a rocket to compete in the University Student Rocketry Competition in the Fall of 2018. This work was completed by senior undergraduate members of SEDS for their senior project, however undergrads were heavily involved throughout.

### University Student Rocketry Competition

The USRC is an annual competition hosted by SEDS-USA to challenge students, to design, build, and launch a multi-stage rocket with a standardized altimeter to the highest possible altitude. The judging panel includes professionals from within the aerospace industry. Winning teams will be awarded a cash prize as well as free attendance to the SEDS SpaceVision 2018 conference. Teams can launch at a field close to their university as long as they are witnessed by an independent party. However, teams can also meet up to organize a regional launch. Points are awarded by the judges based on the following criteria:

## Goals

1. Design and launch a high-powered rocket to achieve **maximum altitude** (at least 3000 feet)
2. Implement a comprehensive recovery system, such that the rocket is reusable

## Constraints

1. Total combined engine impulse must not exceed **640.0 N-s**
2. The rocket must have *at least* two propulsive stages
3. Time: Launch window closes **October 12<sup>th</sup>, 2018**
4. Budget: \$4563.0 from the UNH ME department and Parents Association

## Overall Rocket Configuration and Concept of Operations

A section-view model of the rocket configuration is shown below in Figure 1. This is a high-level description of the major components that will be frequently referenced throughout the remainder of the report.

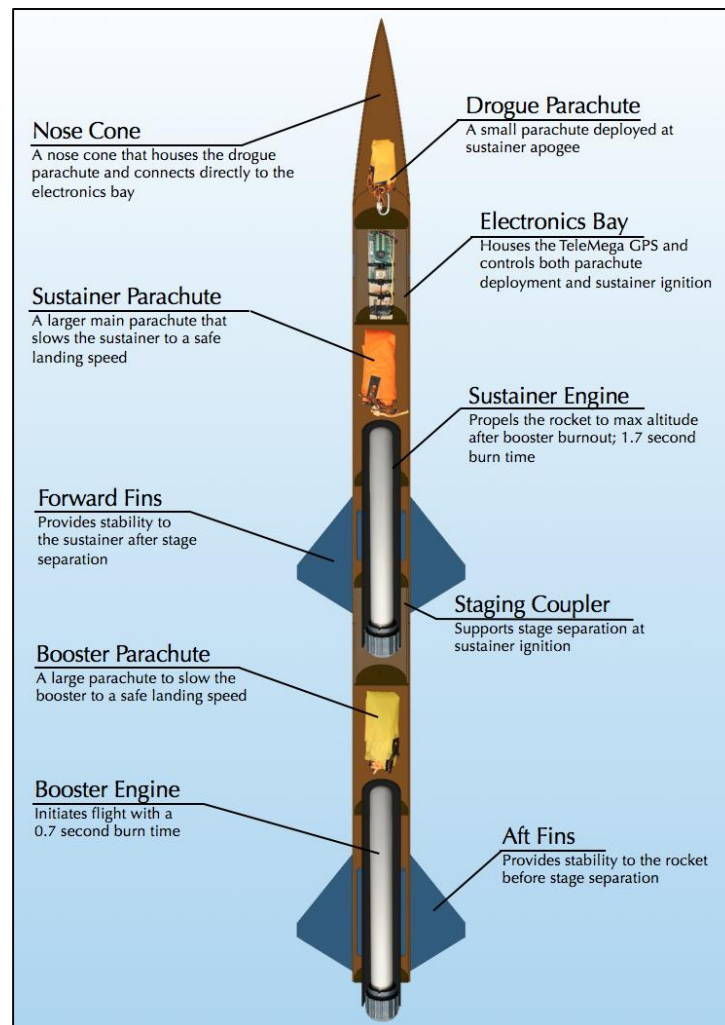


Figure 1. Rocket Components

Both engines are solid-propellant rocket motors manufactured by Cesaroni. Solid engines were chosen for simplicity and were purchased instead of being custom built. Next year UNH SEDS will attempt to manufacture their own hybrid rocket engines.

The TeleMega GPS/Altimeter system used will be referred to as the flight computer. The flight computer, located in the electrical bay (e-bay), sends electrical signals at various preprogrammed events. Current is sent to three individual ignitors; one that initiates the firing of the sustainer engine and two that light ejection charges for deploying the parachutes.

The booster engine is the first engine to fire; ignited manually from a safe distance using a custom ignition switch. Booster burn-out is followed by stage separation, then sustainer ignition. Decoupling the two stages is achieved by drag separation, with increased pressure from sustainer ignition acting as a fallback. The booster parachute then deploys, carrying the booster body tube and booster engine safely to the ground.

The sustainer body tube continues upward until the rocket reaches a maximum altitude, where the flight computer triggers an ejection charge at the appropriate time by evaluating altimeter data. The ejection charge pressurizes the area above the electrical bay, forcing off the nose cone and deploying the drogue parachute. The drogue is a small parachute that works to slow the descent of the rocket to a controlled speed. At a predetermined altitude, the flight computer will send a signal to another ejection charge located in the aft end of the e-bay. This separates the sustainer body tube from the e-bay and deploys the main parachute; a much larger parachute that slows the rocket components considerably. This prevents significant damage from ground impact.

## Launch Operations and Safety Precaution Checklist

1. Assemble Launch Pad in desired location given the rocket flight path, wind direction and time of day
  - 1.1. If the wind is strong in a certain direction, and angle of attack can be useful to compensate for the calculated drift the rocket will ensure on descent
2. Set up launch rail stopper to raise the rocket roughly 6 inches above the launch pad base
3. Slide rocket into position on rail
4. Turn on the TeleMega GPS to begin satellite connection and flight configuration
5. Insert the booster engine assembly into the booster
  - 5.1. Ensure that there are no possible situations that can ignite the booster engine when configuring on the launch pad
6. Screw rocket aft retainer onto engine tube
7. Remove yellow safety cap to begin igniter installation
  - 7.1. Ensure that the range safety officer and the launch director are the only ones at the pad to limit the amount of people around the rocket
  - 7.2. All other members and observers must be 100 feet away from the rocket
8. Install igniter leads into engine by inserting through the nozzle and up the grain until it reaches the top
9. Screw engine casing retainer onto the booster engine tube to constrain the engine in both the positive and negative y directions



10. Attach igniter leads onto the launch system alligator leads.
  - 10.1. Check that the other igniter launch leads are not connected to the battery and the launch director has the launch key
11. Perform final checks on the rocket
  - 11.1. Fin alignment between stages
  - 11.2. Proper retainment on all engines
  - 11.3. Correct pin installations if needed
12. Run from launch pad (It is UNH SEDS tradition to always run from the launch pad before countdown)
13. Verify TeleMega connection, continuity on all igniters and GPS tracking
14. Setup launch controller to the battery for launch
15. Verify surroundings and safety of all people
  - 15.1. Launch safety officer must give the okay for launch
16. Check continuity
17. Countdown --> Launch

Redesign Cycles and the Approach to Accurate Models  
Improvement Cycle

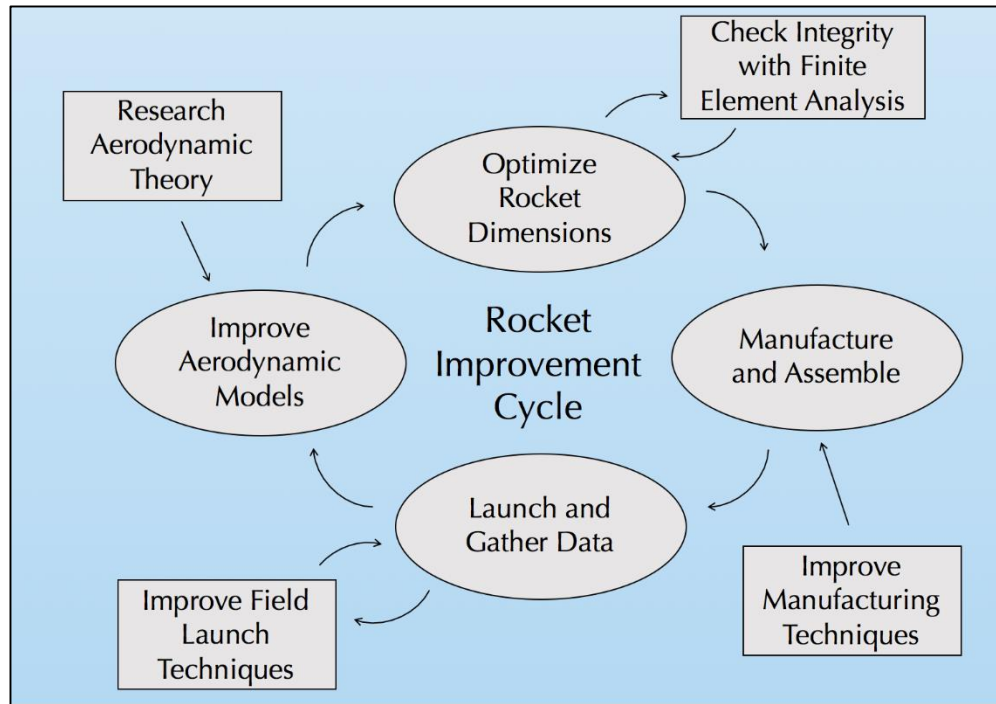


Figure 2. Improvement Cycle

Our team utilized an improvement cycle to ensure progression and improvement on each rocket iteration. The cycle begins with research of aerodynamic theory to enhance our aerodynamic models used for flight simulations. When the flight simulations are validated with experimental data, rocket dimensions are optimized for max altitude. This is done using a nonlinear optimization program in MATLAB. These dimensions are constantly checked with finite element analysis to ensure structural integrity with a minimum factor of safety of 3. The rocket is then manufactured with the optimal dimensions, while attempting to employ better techniques from last build. After the rocket is manufactured, it is launched, and data is gathered. We are always pursuing better launch techniques to ensure recovery of all components. The data from the flight is reviewed and compared to our flight simulations, where the process then repeats.

UNH SEDS is currently on its 8<sup>th</sup> iteration of this cycle; Aether VI with a launch date of May 7<sup>th</sup>. The launch was a perfect end to the 2017-2018 school year with a successful two stage flight with nominal dual deployment and recovery.

### Aerodynamic Models

Obtaining a trustworthy model for the trajectory of a given rocket is essential in rocketry design. It allows for a prediction of location and velocity of rocket at any given point in its flight path. This includes an estimation of the maximum altitude of the rocket, which allows us to quantify the quality of a given rocket design that is directly related to our primary goal.

An accurate model can also provide approximate landing distance from the launch pad for a given crosswind velocity. This is essential in designing the recovery system of the rocket, as it confirms that the rocket will land within the launch field.

A number of commercial rocketry simulation programs currently exist, however many of them are pricy and made to be a “black box”. These programs have been proven to give accurate solutions, but there is typically no access to internal program to gain an understanding how these simulations are achieved. This is not desirable for SEDS, as achieving a fundamental understanding of the physics behind a rocket’s flight takes top priority.

Therefore, our team has created our own aerodynamic models using MATLAB. The program is essentially a numerical simulation of the rocket that calculates altitude and velocity changes over incremental time periods. This required creating accurate stability, thrust, drag, and atmospheric models that can be called upon at each time increment.

## Assumptions

1. The rocket is treated as a combined rigid body that separates at stage separation
  - a. Drag force calculations change to correct for stage separations and parachute deployment
2. Flow over the rocket is steady state with no vortices
3. The attitude of the rocket is constant throughout flight, with nose cone pointing up
  - a. Although this is not always the case, it is too difficult to try and predict changes to attitude with our simulations
4. Fins are flat plates with no cant angle
  - a. Allows for simplification of pressure drag forces on fins
5. The rocket is axially symmetric
  - a. Current manufacturing procedures get us close, but alignment of fins is never perfect
6. Significant drag components considered are base, pressure, and skin friction drag
  - a. These are typically considered the big three for rocketry
7. The pressure drag component from the ogive nose cone at subsonic speeds is negligible, due to its aerodynamic geometry.
  - a. The resulting pressure drag force is significantly smaller than the skin friction drag on the ogive nose cone
  - b. If the joint between the nose cone and body tube is smooth, flow separation will not occur and pressure drag will be negligible in our analysis
8. A rocket can be considered passively stable if it has a caliber above 1 [1]
  - a. Caliber is factored into the design of the rocket to ensure the rocket will be stable through the flight
9. Only pressure drag was considered after parachute deployment
  - a. When the parachute is deployed, the main drag force component is that of pressure drag
10. Crosswind speed is constant for a given launch day
  - a. In actuality, wind speeds may vary at altitude; however, we would have great difficulty predicting this
11. Deviation due to the Coriolis effect from the Earth is negligible
  - a. The rocket flight path covers too small of a distance and the event occurs over a short period of time; the Coriolis effect on the rocket will be quite small relative to other forces

## 12. Gravitational acceleration is constant

- a. For the altitude our rocket is projected to reach, acceleration due to gravity changes by less than 0.1%

## Program Structure

### Stability

In rocketry, caliber is a measure of the stability of the rocket. It is well known that the ideal caliber for a stable rocket is between 1 and 3. Caliber is calculated with Equation 1:

*Equation 1*

$$\text{Caliber} = \frac{C_p - C_g}{D}$$

D is the diameter of the rocket,  $C_g$  is the distance between the tip of the nosecone to the *center of gravity* of the rocket, and  $C_p$  is the distance between the tip of the nosecone to the *center of pressure* of the rocket.

The center of gravity is the average location of the mass of the rocket. This can be estimated by summing the center of gravity of each individual component of the rocket and dividing it by the total mass.

*Equation 2*

$$C_g = \frac{1}{m_{tot}} \sum_i m_i x_i$$

The center of pressure is average location of the pressure on the rocket. In other terms, it is the point where the total sum of a pressure field acts on the rocket. This is important to determine, as the total drag force vector is the value of this integrated pressure field acting through this point. Calculating the  $C_p$  for a rocket design is outlined in a paper by Barrowman (1966) [1]

Fin design is the primary contributor to the location of the center of pressure. We decided to go with a trapezoidal model, mostly for ease of calculation. The trapezoidal fin model is also versatile; as triangular fins can be modeled when  $C_t$  is zero (see Figure 3). Square fins can also be represented as  $X_r$  goes to zero and  $C_t = C_r$ .

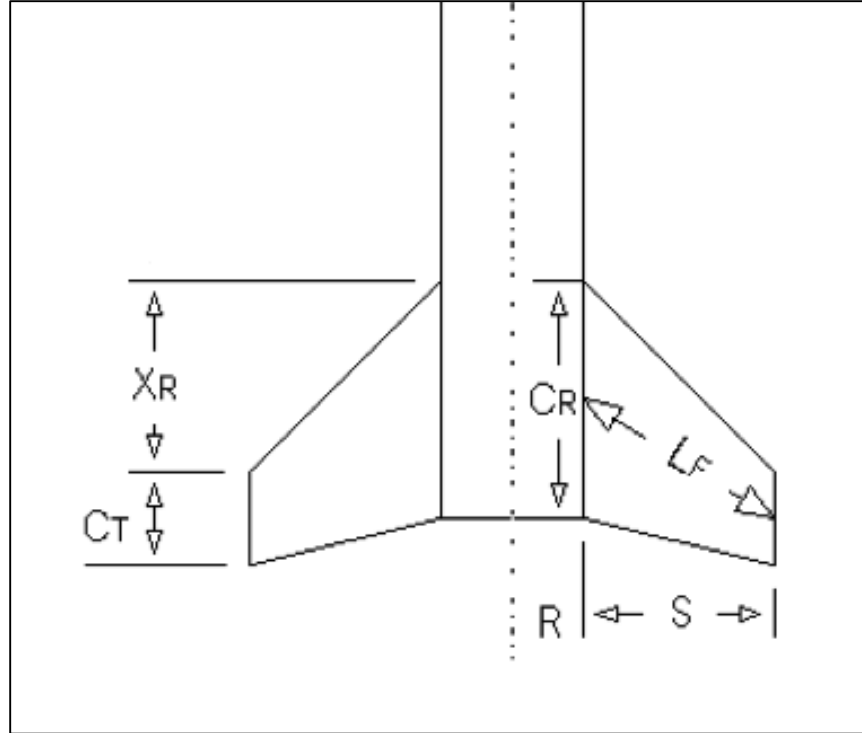


Figure 3 Trapezoidal Fin Model

The following equations are taken from Barrowman (1966) [1]. The approach uses force coefficients based on the relative geometry of the nosecone, body, and fins to approximate the average location of the forces.

Equation 3

$$C_p = (C_{nn} * X_n + C_{nt} * X_t + C_{nf} * X_f) / (C_{nn} + C_{nt} + C_{nf})$$

Where  $C_{nn}$  is the force coefficient for the nose cone, which is experimentally found to be 2 for the ogive shape that we are using for our rockets.  $C_{nt}$  is the coefficient that arises if the diameter of the rocket varies, and in our case it doesn't so it equals zero. The X variables in the equation are the distances between the centroid of the shape and the tip of the nose cone.  $C_{nf}$  is the coefficient that is affected by the fins, and is calculated as follows:

Equation 4

$$C_{nf} = \frac{4 * \text{Number of fins} * (\frac{S}{2 * R})^2}{1 + \sqrt{1 + \frac{2 * L_f}{C_R + C_T}}}$$

Where all variables are defined in Figure 3 Trapezoidal Fin Model

It is essential to make sure that the  $C_p$  is at the very least aft of the  $C_g$ . If the  $C_p$  was in front of the  $C_g$ , the rocket would be extremely unstable and any applied moment from cross-winds would flip the rocket. This

form of instability is similar to instability of an inverted pendulum. Therefore, to drive the  $C_p$  towards the aft end of the rocket, more stabilizing surfaces can be added to the aft end. This is why fins are typically located as far aft of the rocket as possible.

Stability is first calculated in the rocket simulation, as the rocket will not fly if it is outside of the 1-3 caliber range. For multistage rockets, caliber calculation is performed both before and after separation. Caliber also changes with respect to time due to mass loss as the propellant burns. This increases the caliber as the engine fires because it drives the center of gravity forward, as demonstrated in the *Aether IV* stability simulation (Figure 4).

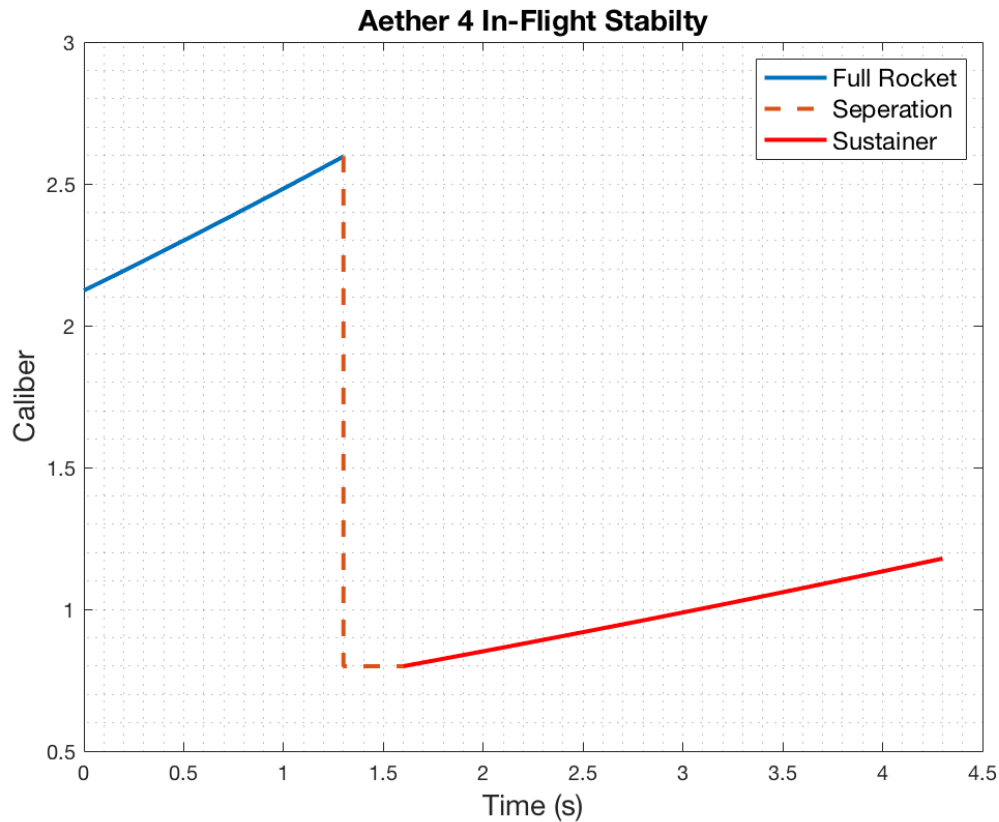


Figure 4. Aether IV Caliber

#### Acceleration

Starting from ignition on the launch pad, the program calculates the acceleration of the rocket at each time step. It accomplishes this by applying Newton's 2<sup>nd</sup> law to the rocket: subtracting the aerodynamic drag force (D) and gravity from the engine thrust (T), and dividing by the mass at that instant.

Equation 5

$$a_i = \frac{T_i - m_i g - D_i}{m_i}$$

### Thrust

The engine thrust is taken from thrust curve data,  $T(t_i)$ . This is supplied by the engine manufacturer; however, we can also use experimental data from the STFR if we are testing a rocket with our competition engines.

### Mass

The mass at a given time,  $m_i$ , is found by subtracting the total lost mass at that instant from the initial mass. Total lost mass considers the amount of propellant burned as well as mass lost from stage separation. Mass of propellant burned at a given time is found by multiplying burn time by the average burn rate of each engine.

#### Equation 6

$$m_{lost} = (t_i - t_{ignition}) \left( \frac{m_{full} - m_{empty}}{t_{burn}} \right)$$

Where  $m_{lost}$  is the expended propellant mass of a given engine,  $t_{ignition}$  is the time at which the engine ignites,  $m_{full}$  is the mass of the engine pre-ignition,  $m_{empty}$  is the mass of the empty engine casing, and  $t_{burn}$  is the total burn time of the engine.

### Drag

The instantaneous drag,  $D_i$ , is calculated by calling a function called GetDrag. GetDrag calculates the combined drag force on the rocket, which is dependent on instantaneous velocity ( $v_i$ ) and height ( $h_i$ ) and time ( $t_i$ ).

Air density has a large effect on the resulting drag, and it changes significantly as the rocket flies into the upper atmosphere. We incorporated an atmospheric model that incorporates the scale height of the Earth to find density at a given altitude. Scale height,  $H$ , is the vertical distance over which the density and pressure of the atmosphere fall by a factor of 1/e. [2]

#### Equation 7

$$\rho_i = \rho_o e^{-\frac{h_i}{H}}$$

Where  $\rho_o$  is the air density at sea level (1.225 kg/m<sup>3</sup>), and  $H$  is the scale height of the earth (8400 m). If  $v_i < 0$ , then the parachute has deployed and the drag from each parachute is also included in the calculation.

At  $v_i > 0$ , the function takes into consideration pressure drag, skin friction drag, and base drag on all rocket components and sums the drag forces to find the total instantaneous force of drag. As the booster body tube separates from the sustainer body tube the function corrects itself; no longer accounting for the full rocket length and two fin sets. To calculate skin friction drag coefficients, we use Reynold's number ( $R_e$ ) and surface roughness ( $R_s$ ).

The following equations 8-12 for drag were taken from Barrowman, 1966 [1].

Equation 8

$$R_e = \frac{vl}{\mu}$$

The surface roughness was estimated using Table 1 and used to find the critical Reynold's number ( $R_{crit}$ ).

Table 1. Surface Roughness

| Type of surface                    | Height / $\mu\text{m}$ |
|------------------------------------|------------------------|
| Average glass                      | 0.1                    |
| Finished and polished surface      | 0.5                    |
| Optimum paint-sprayed surface      | 5                      |
| Planed wooden boards               | 15                     |
| Paint in aircraft mass production  | 20                     |
| Smooth cement surface              | 50                     |
| Dip-galvanized metal surface       | 150                    |
| Incorrectly sprayed aircraft paint | 200                    |
| Raw wooden boards                  | 500                    |
| Average concrete surface           | 1000                   |

Equation 9

$$R_{crit} = 51\left(\frac{R_s}{L}\right)^{-1.039}$$

Using Reynold's number and critical Reynold's number, the coefficient of drag due to skin friction was calculated using Barrowman's empirically derived formulas.

Equation 10

$$\begin{aligned} \text{if } R_e < 10^4: C_{sf} &= 1.48 \times 10^{-2} \\ \text{if } 10^4 < R_e < R_{crit}: C_{sf} &= \frac{1}{(1.5 \ln R - 5.6)^2} \\ \text{if } R_e > R_{crit}: C_{sf} &= 0.032 \left(\frac{R_s}{L}\right)^2 \end{aligned}$$

Compressibility effects were also taken into consideration for subsonic and supersonic speeds. Skin friction drag effects were found for the body tubes and fin sets.

For the specific geometry used, pressure drag effects on the nose cone were considered negligible. To find the coefficient of drag for the fins, the leading-edge angle of the fins and their frontal area were taken into consideration. Again, drag coefficients were found with Barrowman's experimentally derived equations depending on Mach number.



Equation 11

$$\begin{aligned}
 \text{for } M < 0.9: C_D &= [(1 - M^2)^{-0.417} - 1] \cos^2(LEA) \\
 \text{for } 0.9 < M < 1: C_D &= [1 - 1.785(M - 0.9)] \cos^2(LEA) \\
 \text{for } M > 1: C_D &= \left[ 1.214 - \frac{0.502}{M^2} + \frac{0.1095}{M^4} \right] \cos^2(LEA)
 \end{aligned}$$

Finally, base drag was considered using the cross-sectional area of the body tube and relationships with current Mach number.

Equation 12

$$\begin{aligned}
 \text{for } M < 1: C_D &= 0.12 + 0.13M^2 \\
 \text{for } M > 1: C_D &= \frac{0.25}{M}
 \end{aligned}$$

The components of drag were then each calculated with the following drag equation using respective reference areas and drag coefficients.

Equation 13

$$F_D = C_D \frac{1}{2} \rho v^2 A$$

Total instantaneous force of drag was calculated from a summation of skin friction drag, pressure drag, and base drag.

*Trajectory*

From here the velocity and height can be found by simply integrating the acceleration and adding to the values at the previous time step.

Equation 14

$$\begin{aligned}
 v_i &= v_{i-1} + \Delta t * a_i \\
 h_i &= h_{i-1} + \Delta t * v_i
 \end{aligned}$$

A time step resolution of  $\Delta t = 0.01$  was found to be acceptable, as the results converged to a consistent solution. Figure 5 demonstrates a completed simulation for the *Aether IV* design.

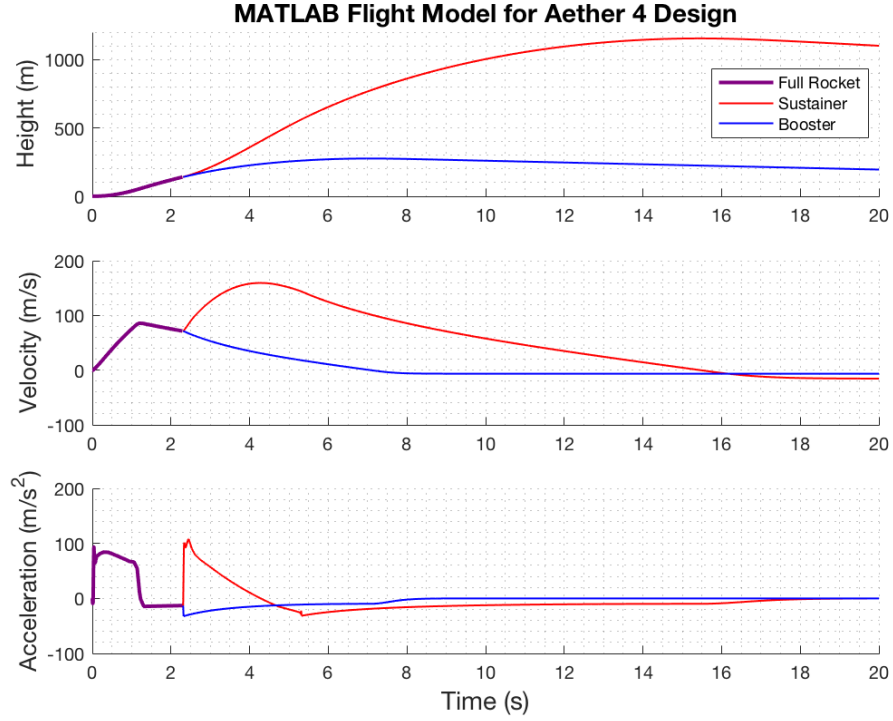


Figure 5. Aether IV Launch Simulation

Landing distance from launch pad,  $D_{LZ}$ , can then be estimated as follows:

Equation 15

$$D_{LZ} = t_{land} v_{crosswind}$$

Where  $t_{land}$  is the time the rocket hits the ground, and  $v_{crosswind}$  is the estimated wind speed on the day of launch taken from weather.gov.

## Model Verification

The MATLAB model can then be verified by comparing simulated results to the experimental results for a given design. Additionally, we modeled the flight our rockets with OpenRocket. OpenRocket is a widely accepted, open-source rocket simulation program which performs similar calculations to our MATLAB model. Significant data points to compare are the apogee and descent velocity. Verification using the *Aether IV* design can be seen in Figure 6.

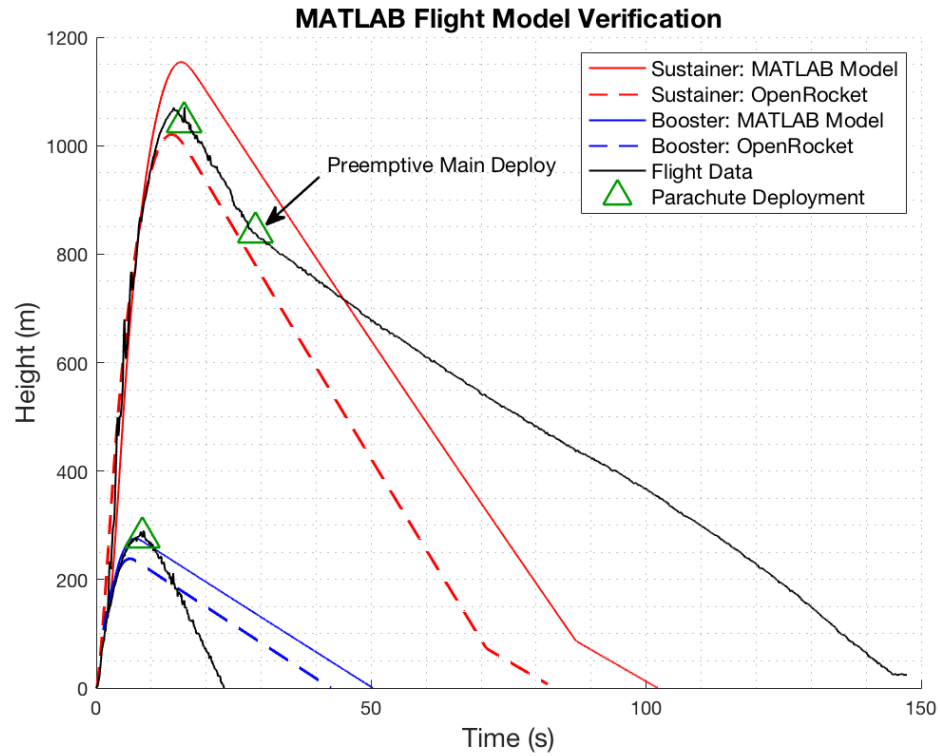


Figure 6: Model verification using Aether IV rocket

The booster deployed its parachute successfully and reached the ground safely, but it fell much faster than predicted. This is because of our estimations used for the parachute in our drag function. Additionally, during test the main deployed pre-emptively, causing the sustainer components to drift further and fall slower. Since these tests were performed we have worked to create a tighter fit between the nose cone and e-bay, while allowing for a loose enough fit that the ejection charge is able to separate the components. This is discussed in detail in the Rocket Iterations section.

The apogee results for both simulations and the experimental data are shown below in Table 2.

Table 2. Aether IV Apogee Predictions

|                         | Flight Data | OpenRocket Model | MATLAB Model |
|-------------------------|-------------|------------------|--------------|
| <b>Sustainer Apogee</b> | 1071.1 m    | 1020.6 m         | 1154.1 m     |
| <b>Booster Apogee</b>   | 290.0 m     | 238.6 m          | 276.3 m      |

Considering the assumptions used in our calculations, the modelled trajectories were decent projections of the experimental flight data. There are many uncertainties that we cannot account for in our simulations, such as varying wind speeds at altitude, friction between the lugs on the rocket and the launch rail, and increases in base drag while the engine is not firing; to name a few. The booster apogee was more accurately predicted by the MATLAB model but the sustainer apogee was better simulated by OpenRocket. For the typical unpredictability of a rocket launch, we were satisfied with our estimations for apogee. Moving forward, the focus for *Aether V* and *Aether VI* has been on successfully triggering events with the flight computer; both main parachute deployment and sustainer engine ignition.

## COMPARISON OF TEST RESULTS and Launch Simulations

### Static Test Fire Rig

#### Test Results

Both the booster stage and sustainer stage engines for the competition rocket were tested to ensure that the rocket will meet the required specifications. A static test fire rig (STFR) was used to experimentally obtain the thrust output of the engines (Figure 7). Engine data is available online from the manufacturer, but we wish to test the engines ourselves. In order to guarantee that the engines match the specifications online, we will compare the experimental data to the expected data.



*Figure 7. STFR test of booster engine*

The test setup consisted of a custom STFR, two 12V batteries, a 250 lb load cell, an AD620 Amplifier, two 100 k $\Omega$  resistors, a DATAQ and a laptop with SignalExpress software. The batteries supply -12V and

+12V to the amplifier, and 6V to the load cell. The load cell was calibrated in lab using known weights, later allowing for a conversion from voltage to thrust (Figure 8).

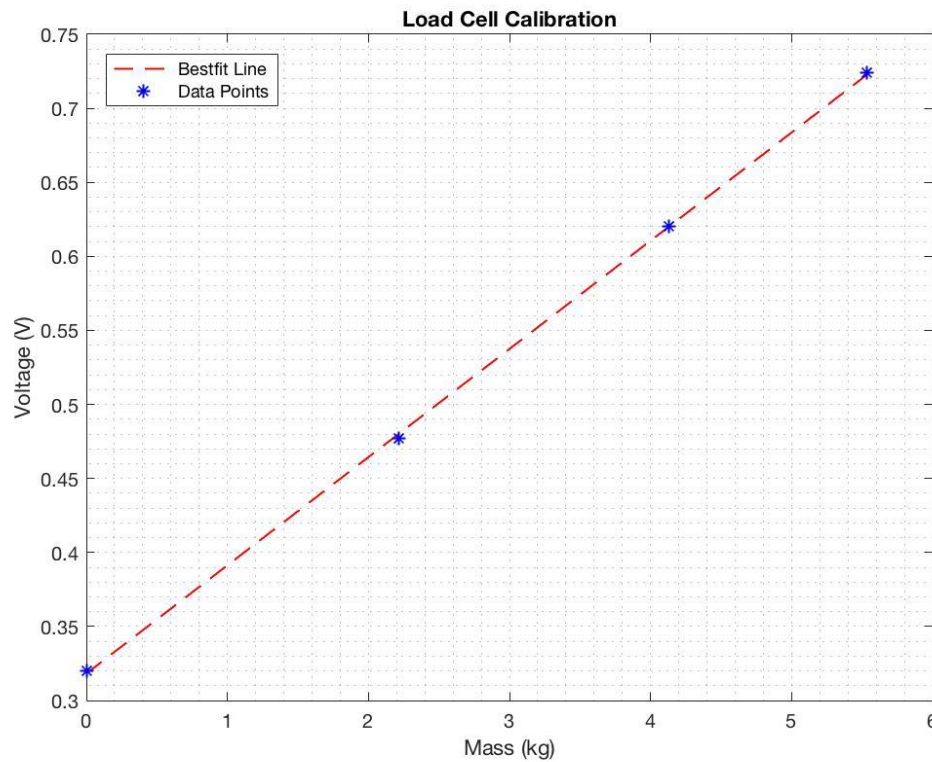


Figure 8. Load cell calibration

The test was conducted on Burley-Demeritt farm in Lee, NH. The STFR was placed on the ground and made as level as possible. This was made difficult by the surrounding snow. The electronics were placed on cardboard to keep them dry and undamaged.

Both the booster and sustainer engines were tested. Each time, the engine ignitor was inserted into the engine with the ignitor leads wired to the ignition controller and the controller to a power supply. The set screws along the test fire rig were tightened and the engine was made concentric with the cylindrical fixture. As one person recorded the data via Signal Express, the other ignited the engine. Both tests were successful. The booster thrust output can be seen in Figure 9.

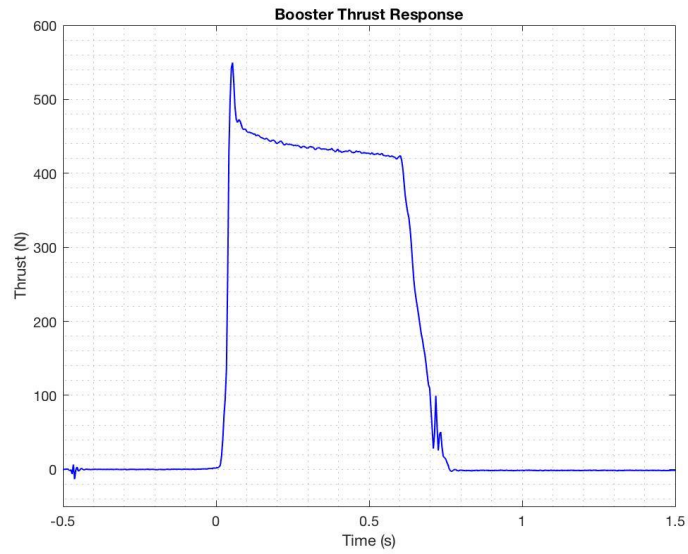


Figure 9. Booster Engine Response

The booster engine maxed out a 549.6 N. The booster engine used was a Cesaroni H399, which is rated by the manufacturer to have a maximum thrust of 545.8 N. The response was fairly smooth which leads us to believe our data acquisition setup was sufficient. The sustainer engine data, shown in Figure 10, was not quite as accurate.

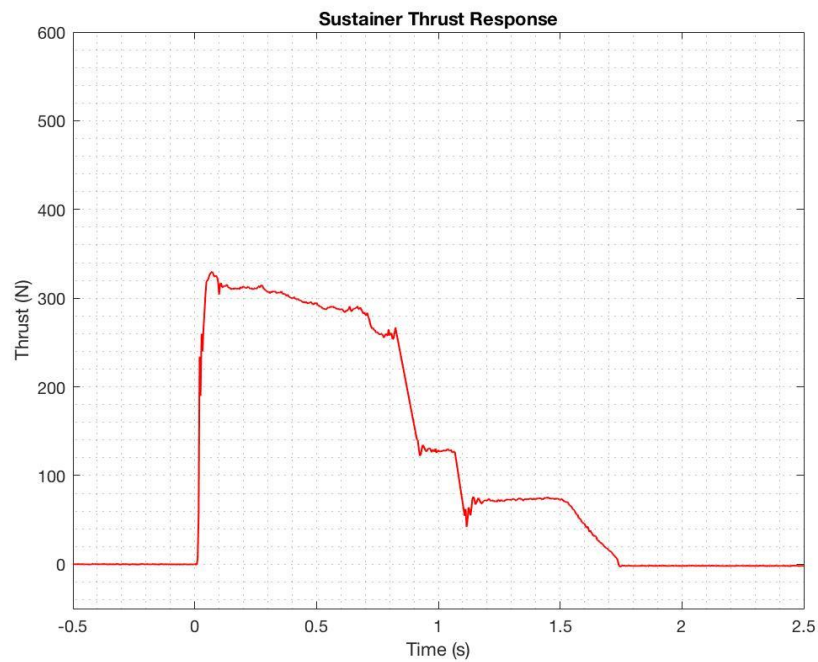


Figure 10. Sustainer Engine Response

The sustainer engine reached a max thrust of 330 N. The sustainer engine used was a Cesaroni I204 and is rated to have a maximum thrust of 356.8 N. The max thrust was off by about 27 N, and the remainder of the response contained a few spikes in the data that were treated as noise and filtered out.

The maximum combined impulse of our rocket cannot exceed 640 N-s as specified by competition guidelines. By integrating the thrust data, we acquired impulse as a function of time. Figure 11 shows the measured impulse of both engine types.

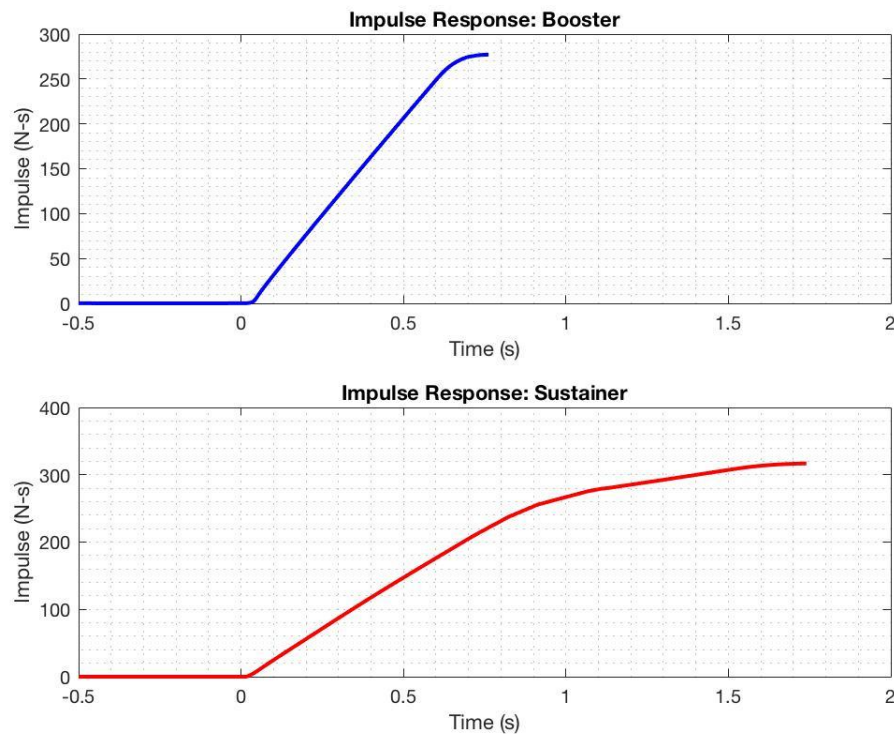


Figure 11. Engine Impulse vs Time

The booster engine was calculated to have a maximum impulse of 277.1 N-s while the sustainer engine was calculated to have a maximum impulse of 316.7 N-s. This would mean our total impulse is 593.8 N-s; well within competition guidelines.

## Test Comparisons

Data for both engine types is supplied by Cesaroni, which can then be compared to our experimental data. The compared thrust responses are displayed in the first two subplots of Figure 12.

Producing an analytical model for the thrust curve of our engines was attempted at multiple points throughout the year. However, after discussion with various UNH ME faculty members, it proved to be outside the scope of our project. Next year an analytical model will be valuable, especially as SEDS moves forward with designing custom hybrid engines.

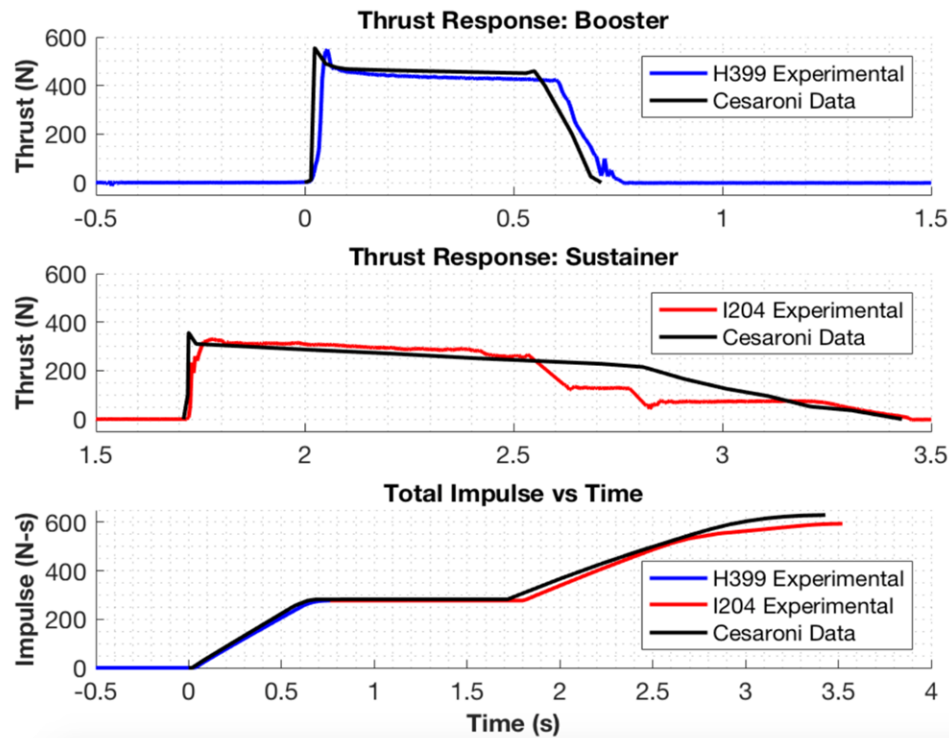


Figure 12. Experimental Data vs Cesaroni Supplied Data

The thrust comparison between the H399 booster engine data sets is noticeably similar. The measured maximum thrust was off by only about 5 N, and both burn times were approximately 0.7 s. The supplied data response seems to show a nearly instantaneous maximum thrust, whereas the measured data shows a slightly more delayed response. This could be due to limitations of the load cell, or slight variations in the rocket engines tested. Small differences in the packing of the solid propellant can have major effects on the characteristics of the engine.

The I204 sustainer engine burned for about 1.7 s during the tests; the time period for which it is rated to burn by Cesaroni. In addition to the aforementioned variations in maximum thrust, the rest of the measured sustainer response was not entirely accurate. In the first 0.75 s, the engine performed close to expected, gradually decreasing in thrust after reaching peak force. At about one second, two spikes in the data were seen that were clearly uncharacteristic of the engine response and the data points were filtered out. Still, the data showed an overall inaccurate response. While the engine was firing, it was noticed that there was a dip in the flame at about one second; leading the team to believe that there was a fault in the particular engine used. The exact same experimental method was used for both engines, and only the sustainer engine had these issues.

The third subplot in Figure 12 shows stacked booster and sustainer impulse data compared alongside Cesaroni data. The H399 booster and I204 sustainer are rated at 282.2 N-s and 347.7 N-s, respectively. This means that total impulse for the rocket was estimated to be 629.9 N-s. As mentioned previously, the experimental total impulse was found to be 593.8 N-s. Since the impulse is simply the integration of the



thrust data, the inaccurate sustainer thrust threw off the total sustainer impulse value. Due to both time and money constraints, the team could not test another sustainer engine. There are plans to purchase more engines for testing in the future, and the STFR setup will be used to acquire a more accurate representation of the I204 sustainer thrust response and total impulse. The tested engine data was used directly in launch simulations.

## Flight Trajectory

### Aether II

On the same day as the STFR experiments, the *Aether II* rocket was launched. The *Aether II* was the first rocket to give the team flight data. Previous rocket builds are highlighted in the Redesign Details section of the report. This was a single-stage launch, with the goal of perfecting the dual deploy recovery technique. The primary objective of *Aether II* was to accomplish this method of recovery, as it's an effective way to prevent excessive crosswind induced drift for rockets with a high-altitude apogee. Based off of the MATLAB trajectory model, expected altitude at apogee was around 760 m, making this rocket the ideal proof of concept test for dual deployment recovery. All *Aether* series rockets use Cesaroni G54 engines for the booster stage; a cheaper, lower impulse engine used for proof of concept.

The onboard electronics bay housed a RRC3 barometric altimeter. This altimeter records height data from ground launch altitude with respect to time. A comparison of our MATLAB trajectory model and the recorded flight data can be seen in Figure 13.

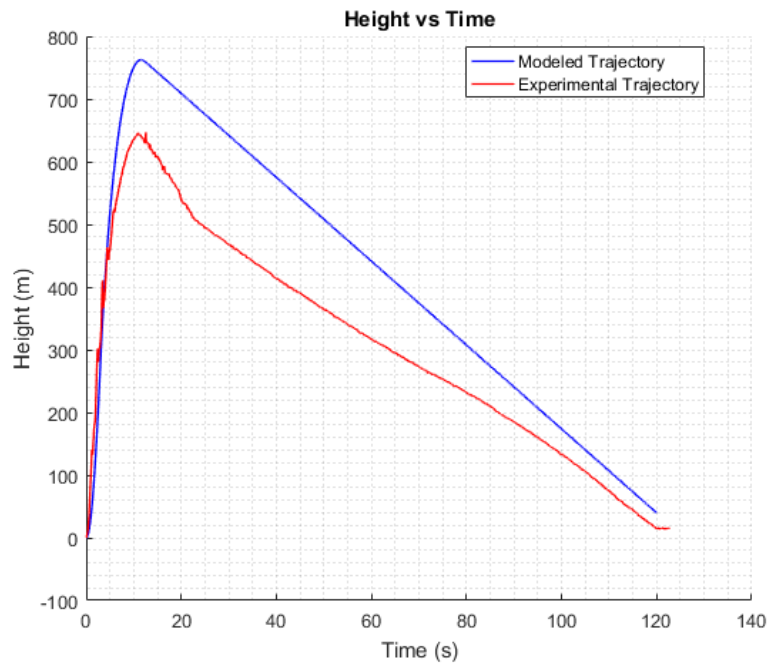


Figure 13. Aether 2 Trajectory Comparison

This data confirmed suspicions that the dual deployment failed. At about 20 seconds into the launch, the ejection charge that is supposed to deploy only the drogue parachute also deployed the main parachute. This increased the drift distance of the rocket significantly (so much so that it drifted away from the field and got stuck in a tree). A comparison of the modeled velocity and recorded velocity in Figure 14 also shows that both parachutes deployed in a single event, as there is a constant falling velocity of around 7 m/s when there should be two distinct falling velocities for each deployment.

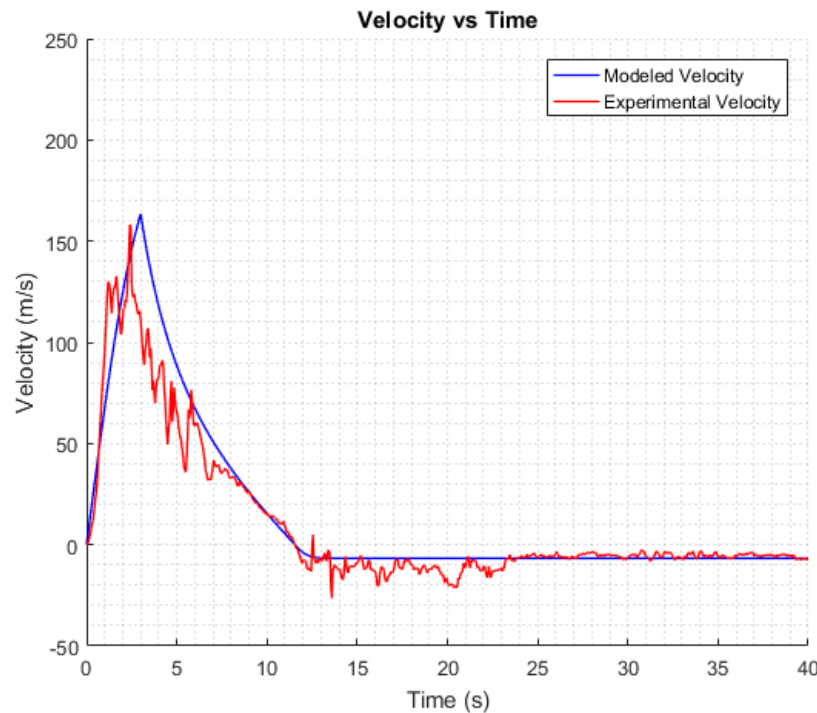


Figure 14. Aether 2 Velocity Comparison

The *Aether II* trajectory model makes use of fundamental aerodynamic relationships between rocket drag, stability, and atmospheric conditions. A number of assumptions were made in constructing this model such as ignoring skin friction drag, crosswinds and updrafts, and assuming laminar flow. Also, the thrust data that was used to determine the acceleration during the engine burn was supplied by the manufacturer, and had not yet been confirmed experimentally by the static test fire rig. The STFR was used to test the thrust of the higher impulse, competition grade engines. These factors may be the cause of deviations between the model and experimental data; especially in estimation of maximum altitude. Otherwise, the implementation of even the most fundamental aerodynamic relationships appears to give a reasonable estimation of rocket dynamics in-flight. Still, there was plenty of room for improvement of simulations.

Predicting the stability of the rocket in flight was also a concern. A model to confirm passive stability for *Aether II* was also created in MATLAB. This was achieved by calculating the locations of the center of gravity and the center of pressure for the entire rocket. The center of gravity is the average location of the

weight of the rocket, and the center of pressure is the point where the total sum of the surrounding pressure field acts on the rocket. The distance between the location of these two points determines the rocket's passive stability, which is a measure of the rocket sensitivity to external aerodynamics forces such as cross-wind. The ideal spacing between these two points is to have the center of pressure 1-2 rocket diameters (calibers) aft of the center of gravity. Since the center of gravity changes as the fuel is consumed, the stability of the rocket also changes. This change is demonstrated by the change in caliber over time in Figure 15.

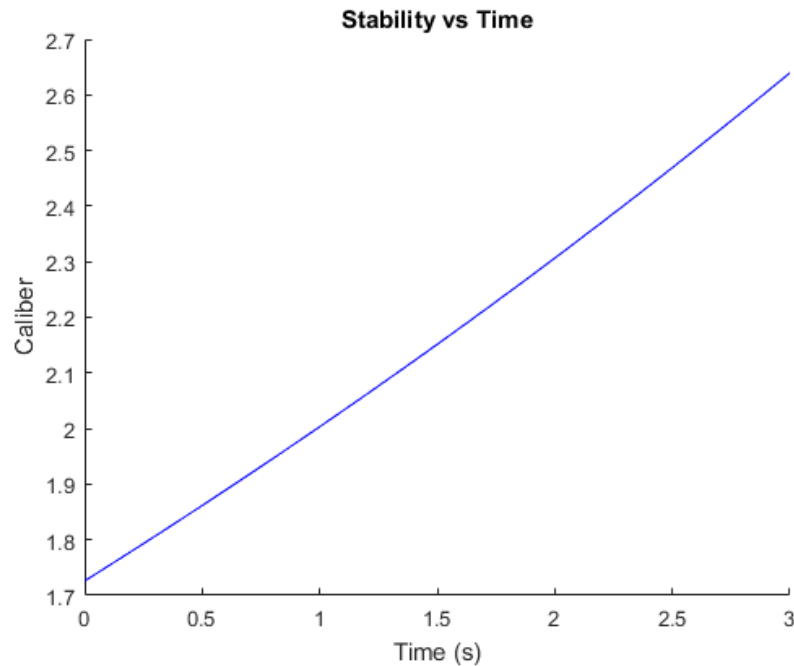


Figure 15. Aether 2 Stability

Since there is no way to quantify the stability of the rocket during flight, the best way we can confirm our model is to observe changes in the rocket's attitude visually. *Aether II* appeared to fly relatively straight while leaving the launch rail, with little change to attitude during the engine burn. This observation suggests that our stability model was accurate, as it was predicted to have a high enough caliber to be conservatively stable.

### Further Testing

After the flight of *Aether II*, the team began development of a multi-stage rocket. *Aether III* was the first iteration to consist of dual-deployment and both booster and sustainer stages. Additionally, significant improvements were made to our flight simulations. Details of each *Aether* model are discussed in the Aether Class Iterations section of Redesign Details. The *Aether IV* launch results and simulations were highlighted on our URC poster. *Aether IV*'s simulation was the first to utilize our nonlinear optimization

techniques that determined ideal dimensions to maximize altitude. Since the URC, we have constantly been working to improve, launching and recovering *Aether V* and later this week launching *Aether VI*.

The following section of the report will highlight each individual rocket iteration, describe our developmental process, and explain, in great detail, our improved MATLAB simulations. Unlike many senior design projects, rocketry requires constant redesign; testing and analyzing results each time, learning from successes and failures alike. Redesign Details will continue to discuss flight test results and how they compared to our MATLAB and OpenRocket models.

## Nonlinear Optimization

Once it has been verified that the aerodynamic models are reasonably trustworthy, optimal dimensions for a given rocket configuration can be determined by using the built-in MATLAB nonlinear programming solver, *fmincon*. This solver finds the minimum of a nonlinear multivariable function, and can be constrained by both linear and nonlinear relations. It accomplishes this using an interior-point algorithm that, in simplified terms, constantly varies each variable at each iteration and follows the gradient of the output.

To utilize this solver, all of the aerodynamic models had to be combined into a single function. The inputs of this function are the various dimensions of each component of the rocket and the output is the simulated maximum altitude.

The results of the solver applied to our *Aether VI* design can be seen in Figure 16.

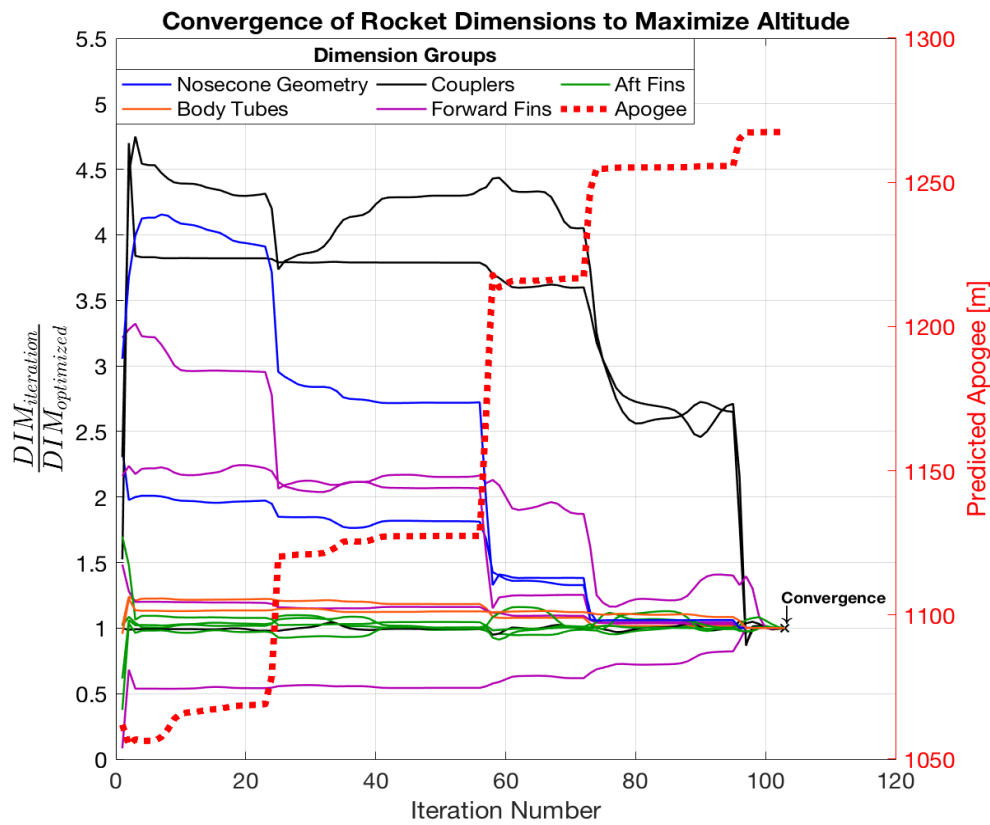


Figure 16. Optimization of *Aether VI* Dimensions

The dotted red line shows that approximately 200 m was added to the modeled apogee the initial guess of the dimensions after optimization. The solid colored lines demonstrate how every dimension that is being varied converges to a single set of dimensions that results in the highest predicted apogee. This particular optimization had 104 iterations before it converged on a function result that is non-decreasing in all feasible directions. Specifically, the amount of change from the final iteration was below the stop-tolerance of the program.

### Constraints

The stability model is used as a constraint in this solver. The relation between rocket dimensions and stability is implemented such that the final solution must also have dimensions that result in an initial caliber of 1.5, both at the launch pad and after stage separation.

Manufacturing limitations dictate the upper and lower bounds for each rocket dimension. These include limits for coupler tolerances, required clearance between fins and motor, and the thickness of available materials.

### Finite Element Analysis

The final dimensions are then used to build 3D models in Solidworks and perform finite element analysis to confirm structural integrity of parts that are prone to failure, such as the engine centering ring seen in Figure 17. A factor of safety of 3 is typically used in aerospace vehicles [3], so this is the minimum we use to ensure structural integrity during flight.

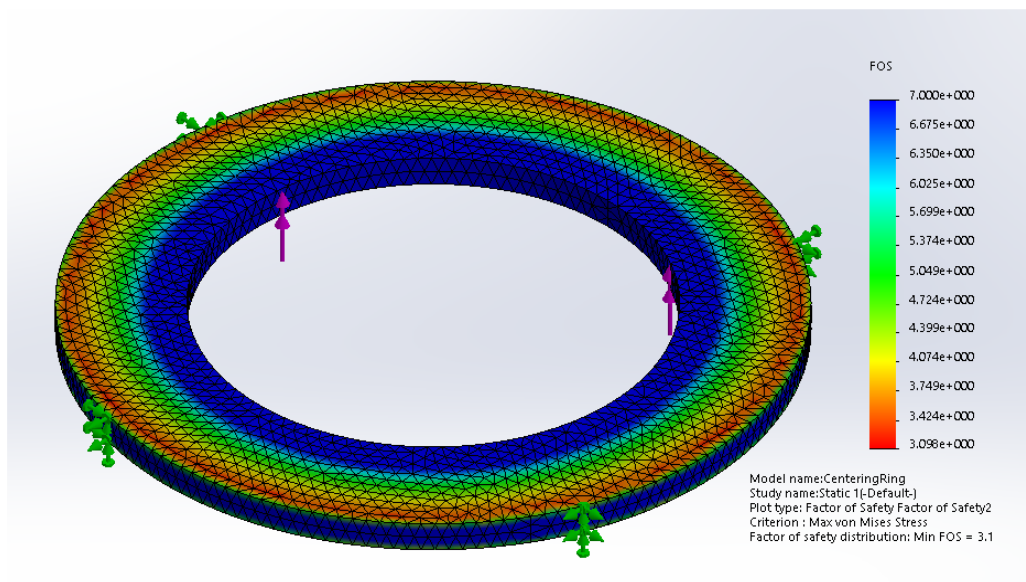


Figure 17. Centering Ring FEA

If the factor of safety falls below 3 for any of the parts, the optimization program must be reconfigured and reran to get a new set of dimensions. Typically, this means adjusting the upper and lower bounds of

certain dimension to ensure that there will be more material on the part on the next set of optimized dimensions.

## Rocket Iterations

### Model Rocket Class Iterations

#### Model Rocket v1

Launched on 2/17/18

Rocket v1 was a single stage cardboard rocket with wood fins and a plastic conical nose cone. The body tube inner diameter is 38 mm and utilized a G54 engine. The objective for this rocket was to successfully manufacture, launch and recover, with little concerns of performance. After the build it was discovered the nose cone was the incorrect size; resulting in the body tube outer diameter being larger than the nose cone outer diameter. This is not optimal as it will induce much greater drag. During ignition, the engine fell out of the body tube due to no engine retainer. This resulted in a failure to launch. The two bolts used to hold the engine in place deformed the body tube, therefore it was concluded the centering rings must be used moving forward.

#### Model Rocket v2

Launched on 2/24/18

Rocket v2 was constructed with the intent to fix the problems from v1. The rocket had the same overall structure as v1, with an implementation of a proper sized nose cone, correct engine detainment and centering rings. The goals for v2 were the same as v1 with an emphasis on launch. The launch was successful, and the flight followed a straight and stable trajectory. Recovery was unsuccessful due to the lack of visibility through the clouds. The rocket was lost and it was unclear if the main parachute deployed at apogee.

### Aether Class Iterations

#### Aether I

Launched on 3/19/18

The *Aether* class iterations utilize dual parachute deployment through the use of an electronics bay rather than an engine ejection charge. The goal for *Aether I* is to successfully launch a single stage rocket with parachute dual deployment. *Aether I* consists of a 54 mm diameter cardboard body tube with wood fins, an ogive nose cone, and the same 29 mm engines utilized in for v1 and v2. Launch was successful with a straight and stable trajectory. The altimeter read and apogee 550 meters when open rocket predicted an apogee of 330 meters. This discrepancy is most likely due to user error of open rocket simulations. At apogee the drogue deployed, and the aft body tube separated from the coupler and body tube, resulting in free fall. The drogue was tangled during decent and the main parachute failed to deploy at 150 meters like it was supposed to. The aft body tube had a hole located at the bottom between the centering rings. The

reason for the failure is suspected to be because the shear pins that are supposed to keep the body tube, coupler and electronics bay aligned were not removed, resulting in improper separation.

## Aether II

Launched on 3/26/18

*Aether II* was designed and manufactured with the intent to fix the problems from *Aether I*. It is a remake of *Aether I* with all the same characteristics and dimensions. The goal of *Aether II* for successful dual deployment and recovery. Specifically, shear pins must be removed so proper separation can occur, the drogue must unfold correctly and main parachute ejection charge should occur at 150 meters. Launch was successful, except trajectory was unstable resulting in rocket wobble. This can be fixed with better fin alignment. Drogue and main parachute activated correctly, however it was stuck in a tree. This is tough to control as the launch site is not as large as desired. To fix this we could deploy main parachute lower and potentially a smaller drogue, producing less drift.

## Aether III

Launched on 4/9/18

*Aether III* was the first attempt at a multistage rocket. The rocket was 54 mm in diameter, utilized wood fins, an ogive nose cone, a G54 booster engine and a G125 sustainer engine. The fins were aligned using a fin alignment tool to prevent the roll of the rocket. A lower parachute deployment height was selected to prevent drift. The goal is to successfully launch a two-stage rocket with proper dual deploy. For launch, both the booster and sustainer fired correctly, the booster parachute deployed and was recovered successfully. The sustainer parachute did not deploy due to incorrect packing and insufficient black powder charge. The electronics bay and nose cone were lost in the forest, resulting in no experimental data. Due to this loss, a Telemega GPS was acquired to prevent the loss of rocket and electronics bay.

## Aether IV

Launched on 4/13/18

*Aether IV* was designed and manufactured to improve on the mistakes made from *Aether III*. *Aether IV* is the same design as *Aether III* except implementing a blue tube body, fiberglass fins and fiber glass centering rings. Fiber glass is a stronger more aerodynamic material than wood. Black powder was calculated, measured, and tested to ensure the charge was powerful enough to deploy the parachutes. During launch, the booster fired with the trajectory changing angle mid-flight. The booster was recovered successfully, however the sustainer never ignited. The sustainer did not ignite because an improper ejection setting was selected using the new GPS system. The drogue parachute could not be fully deployed because the igniter was shorter than the Kevlar chord, preventing full separation. After flight, it was concluded that components such as centering rings and fins need to be milled; not sanded by hand. The more precisely the rings are the machined the easier it is to make the engine concentric with the body tube.

## Aether V

Launched on 4/22/18

*Aether V* utilized the salvaged components from *Aether IV* to attempt a relaunch with proper sustainer ignition. The Booster stage was remade because it was damaged, this time using a fin mold to evenly apply epoxy. The fins and centering rings were cut using a mill for better precision. The sustainer ignition wire was lengthened so the drogue can deploy properly. Testing was conducted to ensure the sustainer and booster ejection charges would ignite and parachutes would deploy. Better parachute folding techniques were employed to prevent tangling. Booster launch was successful, however the sustainer failed to fire. It was concluded that the ignition failure was due to an igniter short. To fix this we started buying igniters rather than making our own.

## Current Iteration (Detailed)

### Aether VI

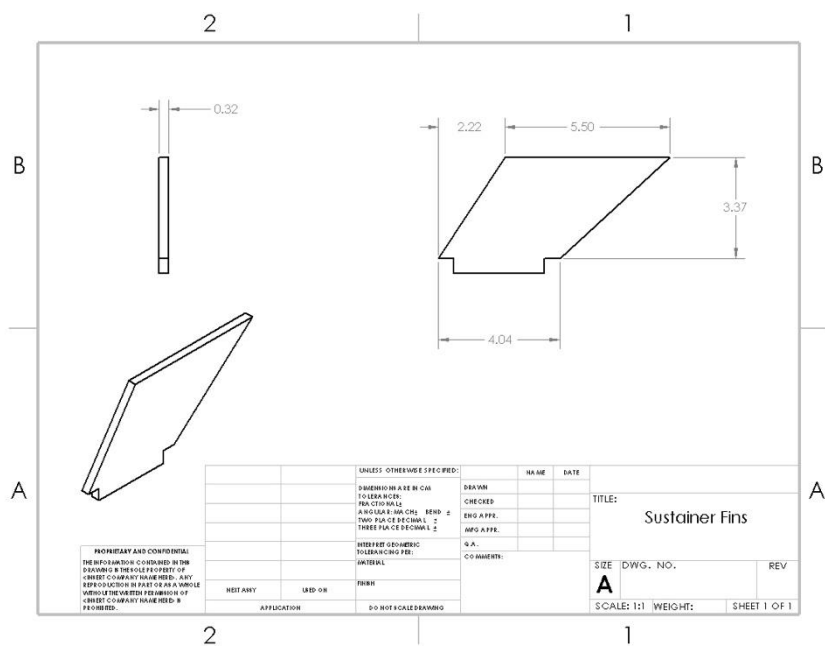
Launch Date: 5/9/18

*Aether VI* will be very similar to *Aether V*, and we are focused on achieving consistent ignition and parachute deployment. Since our homemade ignitors can no longer be trusted, high-grade firework ignitors were purchased. Also, two parachutes were purchased that were made specifically for rocketry. The material is much thinner and does not hold its folded shape as easily as nylon does. The booster from *Aether V* will be reused, and a new sustainer body tube was made. As we approach the manufacturing of our competition rocket, we will consider the lessons learned from the Aether series as well as understand the tolerances needed for all the correct fits of each part. We have begun to master the art of correct tolerances to provide the perfect amount of fit to withstand flight, but be able to maneuver in flight when needed including all staging and recovery events.

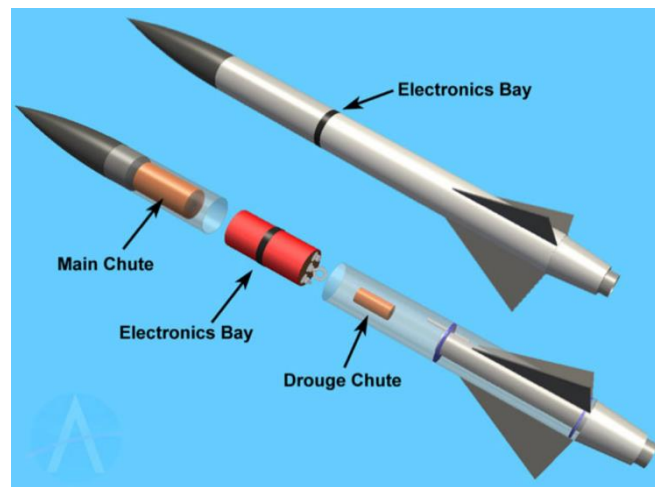
The following drawings are a sample of the most current optimized dimensions for various components. The fin dimensions have been used in *Aether VI* construction.

For our competition rocket the team will be switching to a Von Karman nose cone geometry in anticipation of supersonic velocities. The nose cone shown in the Figure 20 SolidWorks drawing is the competition rocket nose cone. For now, *Aether VI* fin dimensions will be used for the competition rocket as well (shown in Figure 18 and Figure 19); although as the team fine-tunes its MATLAB code and *fmincon* results more optimal dimensions may be utilized in the final iteration.









*Figure 21. Dual deployment basic design*

Parachute deployment was the most difficult part of the project for the team. It was a challenge for much of the year to establish what electronics to use and how to effectively use them for successful recovery. Originally, using an accelerometer with a mini servo and Arduino was proposed to trigger each deployment. Ultimately, we found it was in the team's best interest to use the TeleMega altimeter. The device can operate for dual-deployment as well as provide GPS tracking so that the rocket can be retrieved in the case of extensive drift during recovery. An electronics bay was created for the altimeter to keep it safe and constrained in the rocket. The system is shown in Figure 22.

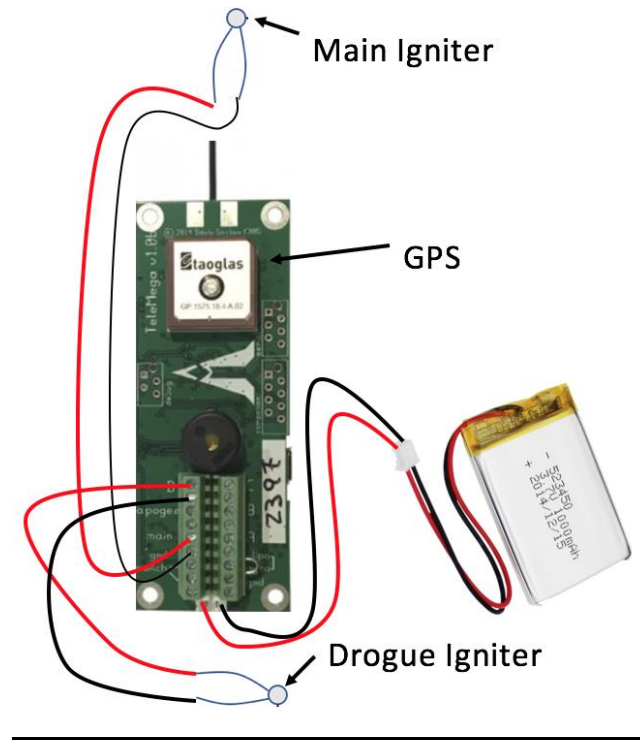


Figure 22. Electronics schematics of the TeleMega GPS

### Parachute Design

Getting the Rocket to apogee is only half the challenge. Recovery of the rocket is just as vital. To have a successful recovery of the competition rocket will utilized a dual-deployment recovery system. Dual-deployment includes a small parachute with a high decent velocity from apogee and a larger parachute with a low decent velocity at a lower predetermined altitude. From the weight of our rocket we could calculate the required diameter for our main and drogue parachutes. Considering the variables listed below, we determined that acceptable decent rates for our main and drogue would be 5 m/s and 30 m/s, respectfully. This determination was made using a chart in *Parachute Recovery Systems Design Manual* by Knacke [4]; which can be seen in Figure 23.

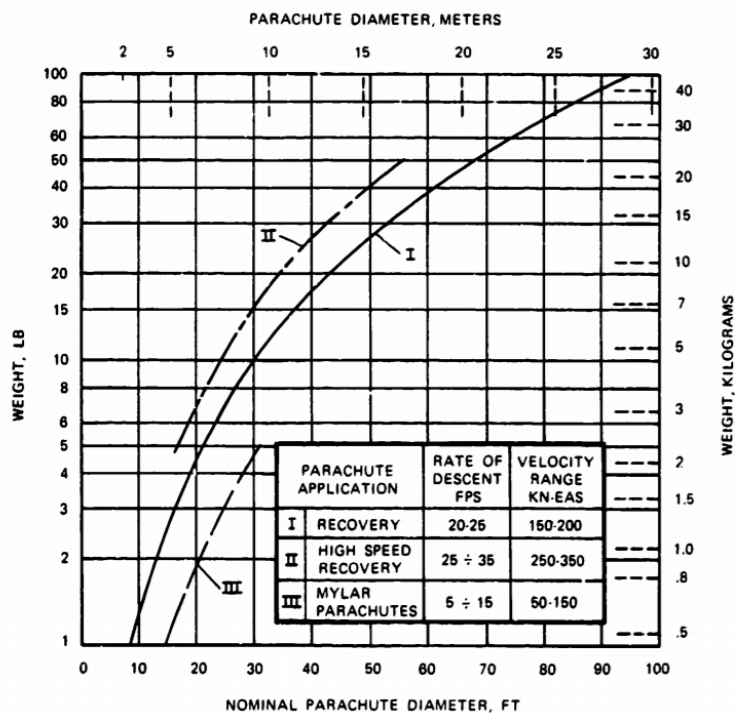


Figure 23. Parachute Descent Speeds























Optimization of the parachute was done both experimentally and analytically. The analytical portion can be seen below with equations and figures referenced from Knacke. Experimentally we were able to adjust and optimize our parachute over the course of our initial four rockets and the *Aether* series. Due to the success of the parachutes in our *Aether* series, we will be continuing this method towards our carbon fiber series.

Our initial parachutes were handcrafted and consisted of a heavy plastic-nylon material and kevlar string. These were cut to a hexagonal shape with duct tape for reinforcement on the connection between the parachute and strings. The parachutes had two significant issues during their deployment. The first was the material was very stiff and once wrapped would take up a larger than desired portion of the body tube. Secondly, the parachutes would often conform to their wrapped shape and would have difficulty opening once ejected.

The second series of parachutes focused heavily on the material used in production. We purchased a 30'' Nylon Parachute then modified it through the addition of a slit hole and reinforcement to the connection lines. Slit hole calculations were incorporated in our initial calculations of parachute diameter. The total canopy surface area was modeled as the difference between the complete parachute at diameter and the slit hole diameter. A secondary advantage to purchasing our parachute was their predetermined drag coefficient. From here, calculations and experimental determinations were simplified. Table 3 shows important characteristics of various parachute geometries.

Table 3. Solid Textile Parachutes

TABLE 5-1. Solid Textile Parachutes.

| TYPE                      | CONSTRUCTED SHAPE                                                                   |                                                                                     | $\frac{D_c}{D_o}$ | INFLATED SHAPE<br>$\frac{D_i}{D_o}$ | DRAG COEF<br>$C_{D_o}$<br>RANGE | OPENING FORCE COEF<br>$C_X$<br>(INF MASS) | AVERAGE ANGLE OF OSCILLATION, DEGREES | GENERAL APPLICATION                  |
|---------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------|-------------------------------------|---------------------------------|-------------------------------------------|---------------------------------------|--------------------------------------|
|                           | PLAN                                                                                | PROFILE                                                                             |                   |                                     |                                 |                                           |                                       |                                      |
| FLAT CIRCULAR             |    |    | 1.00              | 0.67 TO 0.70                        | 0.75 TO 0.80                    | ~1.7                                      | ±10 TO ±40                            | DESCENT, OBSOLETE                    |
| CONICAL                   |    |    | 0.93 TO 0.95      | 0.70                                | 0.75 TO 0.90                    | ~1.8                                      | ±10 TO ±30                            | DESCENT, M < 0.5                     |
| BICONICAL                 |    |    | 0.90 TO 0.95      | 0.70                                | 0.75 TO 0.92                    | ~1.8                                      | ±10 TO ±30                            | DESCENT, M < 0.5                     |
| TRICONICAL POLYCONICAL    |    |    | 0.90 TO 0.95      | 0.70                                | 0.80 TO 0.96                    | ~1.8                                      | ±10 TO ±20                            | DESCENT, M < 0.5                     |
| EXTENDED SKIRT 10% FLAT   |    |    | 0.86              | 0.66 TO 0.70                        | 0.78 TO 0.87                    | ~1.4                                      | ±10 TO ±15                            | DESCENT, M < 0.5                     |
| EXTENDED SKIRT 14.3% FULL |    |    | 0.81 TO 0.85      | 0.66 TO 0.70                        | 0.75 TO 0.90                    | ~1.4                                      | ±10 TO ±15                            | DESCENT, M < 0.5                     |
| HEMISPHERICAL             |  |  | 0.71              | 0.66                                | 0.62 TO 0.77                    | ~1.6                                      | ±10 TO ±15                            | DESCENT, M < 0.5, OBSOLETE           |
| GUIDE SURFACE (RIBBED)    |  |  | 0.63              | 0.62                                | 0.28 TO 0.42                    | ~1.2                                      | 0 TO -2                               | STABILIZATION, DROGUE, 0.1 < M < 1.5 |
| GUIDE SURFACE (RIBLESS)   |  |  | 0.66              | 0.63                                | 0.30 TO 0.34                    | ~1.4                                      | 0 TO -3                               | PILOT, DROGUE, 0.1 < M < 1.5         |
| ANNULAR                   |  |  | 1.04              | 0.94                                | 0.85 TO 0.95                    | ~1.4                                      | < -6                                  | DESCENT, M < 0.5                     |
| CROSS                     |  |  | 1.15 TO 1.19      | 0.65 TO 0.72                        | 0.60 TO 0.85                    | 1.1 TO 1.2                                | 0 TO -3                               | DESCENT, DECELERATION                |

The desired result of the parachute calculations is the equilibrium velocity, or rate of descent ( $V_e$ ). The inputs needed to determine this value are the weight of the payload ( $W_t$ ), the parachute drag coefficient ( $C_{D_o}$ ), the canopy surface area ( $S_o$ ), and the density of air at the given altitude ( $\rho$ ).

Equation 16

$$V_e = \sqrt{\frac{2 \cdot W_t}{C_{D_o} \cdot S_o \cdot \rho}} \text{ ft/s}$$

Equilibrium velocity can then be used to give us an optimal parachute diameter that will allow the rocket to fall at the desired rate of descent; where  $D_o$  is the nominal parachute's diameter.

Equation 17

$$Do = \sqrt{\frac{2.547 * Wt}{CDo * Ve^2 * \rho}} ft$$

## Competition Rocket and Final Assembly Procedure

The bulk of our work leading up to this report was to become acclimated to the science of rocketry. As we near our launch date, we will begin the final design of our final competition rocket. The overall rocket assembly and flight events that have been utilized will be nearly the same for our competition rocket. The body material, engines, and final parachute selection will differ.

### Material Choice

Carbon fiber offers a lightweight and strong selection for material, despite its material cost. All major rocket components can be purchased with proper tolerances to ensure the correct fits of each part. The centering rings, bulk plates and nose cone will be manufactured in house. The centering rings and bulk plates can be cut from carbon fiber sheets that will be milled to precise dimensions. The nose cone will be created with proper molds and carbon fiber manufacturing processes. This will be researched over the summer to ensure a smooth assembly of all the competition rocket parts.

### Engine Choice

The engines that have been used to increase our body of knowledge in rocketry were G class. The biggest transition that will occur from the flight of our competition rocket will be the addition of total impulse to the rocket. We have done work on handling the bigger high power engines with our Static Test Fire Rig that showed us first-hand the kick these engines provide. The integration of these larger competition engines do not require much change to our competition except to allow extra room for the increase in length that engine has.

### Parachute Choice

The parachute research and testing we have completed has given us a base of knowledge that is easily transferable to our competition rocket. The only difference in input will be the mass that will be needed to slow down for descent. We will continue to use semi-premade parachute material to manufacture the perfect parachute.

### Final Assembly Procedure

Through our rocket improvement cycle, we have continuously created new tactics to create a more precise and better performing rocket. The transition to carbon fiber will introduce different handling condition that will be need to be carefully thought out, but the overall process is the same. Below is a detailed list that we continue to modify to increase our body of knowledge in rocketry building techniques.

1. Obtain materials:
  - 1.1. Three precision cut fins
  - 1.2. Two standard rail buttons that are compatible with the launch rail
  - 1.3. Carbon fiber body tubes & custom carbon fiber nose cone
  - 1.4. Carbon engine tubes for the booster & sustainer engines
  - 1.5. Parachute
    - 1.5.1. Kevlar

- 1.5.2. String
- 1.5.3. Nylon sheet
- 1.5.4. Duct tape
- 1.5.5. Carabineer
- 1.6. Engine aft retainers to constrain booster and sustainer engines
- 1.7. 5-minute epoxy
- 1.8. Thrust ring & engine tube centering rings
- 1.9. Electronic bay materials
  - 1.9.1. Tight fit carbon fiber coupler
  - 1.9.2. Bulk plates
  - 1.9.3. Threaded rods
  - 1.9.4. Electronics sled
  - 1.9.5. TeleMega GPS
  - 1.9.6. Lithium Polymer battery
  - 1.9.7. Zip ties
  - 1.9.8. Bolt, washers and nuts
  - 1.9.9. Switch
- 2. Epoxy top and bottom centering rings onto outside of engine tube such that it is above and below the fin tabs when installed
- 3. Drill holes through centering rings such that Kevlar may be fitted through.
  - 3.1. [Sustainer] Measure Kevlar such that sustainer main parachute is halfway between booster engine tube and booster shoulder. Measure Kevlar such that sustainer drogue has min 1ft between nose cone and shoulder.
  - 3.2. [Booster] Measure Kevlar such that there is a min 2ft kevlar between main booster parachute and booster body tube.
  - 3.3. Knot Kevlar just below each centering ring such that force acts upon rings when pulling on the Kevlar.
- 4. Attach entire engine assembly
  - 4.1. Insert engine tube such that the engine tube sticks out of the body tube so there is enough length for the casing retainer to be epoxied on later (approx 1.5+in)
  - 4.2. Epoxy bottom of fin tab and mate to engine tube surface through fin slots
  - 4.3. Epoxy casing retainer onto engine tube
  - 4.4. Epoxy bottom centering ring to the body tube to fully constrain engine assembly
- 5. Attach rail buttons to body tube
  - 5.1. Aligned axially, so that the fins are furthest from the rail
  - 5.2. Bottom rail button placed 1" from bottom of body tube, Top rail button halfway up booster body tube
- 6. Add epoxy fillets to where the fin meets the body tube
- 7. Repeat steps 2 – 5 for the booster assembly
- 8. Manufacture parachutes
  - 8.1. Reinforce kevlar slack lines
  - 8.2. Cut slip hold
  - 8.3. Connect parachute lines with swivel and parachute protector.
- 9. Fold parachute into triangle, roll from slip-hole to outer edge



10. Wrap the parachute with the strings
11. Wrap excess wadding and insert parachute assembly into applicable body tube or nose cone
12. Repeat for each parachute assembly
13. Connect all kevlar lines to the electronics bay and assemble complete rocket

### Launch Dates

There are two launch fields in New England that meet the maximum altitude ceiling of the competition fiber rocket, CMASS and MMMS. The CMASS is located Amesbury, Massachusetts and offers a flight ceiling of 10,000 feet. The MMMS is located in South Berwick, Maine and also offers a flight ceiling of 10,000 feet. The launch dates that will support this competition before the launch window closes are:

- September 1<sup>st</sup> at MMMS in South Berwick, Maine
- September 15<sup>th</sup> at CMASS in Amesbury, Massachusetts
- September 29<sup>th</sup> at MMMS in South Berwick, Maine
- October 6<sup>th</sup> at CMASS in Amesbury, Massachusetts
- October 13<sup>th</sup> at CMASS in Amesbury, Massachusetts

## BILL OF MATERIALS and Cost Breakdown

### Overall Cash Flow

|                                    | Beginning | Nov-17   | Dec-17 | Jan-18 | Feb-18   | Mar-18   | Apr-18   | May-18 | Jun-18 | Jul-18 | Aug-18 | Sep-18 | Oct-18 | Total    |
|------------------------------------|-----------|----------|--------|--------|----------|----------|----------|--------|--------|--------|--------|--------|--------|----------|
| Cash on hand (beginning of month)  | 500.00    | 500.00   | 481.80 | 481.80 | 24.11    | 3,238.29 | 2,865.10 | 496.69 | 496.69 | 496.69 | 496.69 | 496.69 | 496.69 |          |
| <b>CASH RECEIPTS</b>               |           |          |        |        |          |          |          |        |        |        |        |        |        |          |
| Grants                             |           |          |        |        | 3,863.00 |          |          |        |        |        |        |        |        | 3,863.00 |
| Sponsors                           |           |          |        |        |          |          |          |        |        |        |        |        |        | 0.00     |
| Donations                          |           |          |        |        |          |          |          |        |        |        |        |        |        | 0.00     |
| Fund Raising                       |           |          |        |        |          |          |          |        |        |        |        |        |        | 0.00     |
| Other contributions                |           |          |        |        |          |          | 38.06    |        |        |        |        |        |        | 0.00     |
| Pending Transactions               |           | 700.00   |        |        |          |          |          |        |        |        |        |        |        | 700.00   |
| <b>TOTAL CASH RECEIPTS</b>         |           | 700.00   | 0.00   | 0.00   | 3,863.00 | 0.00     | 38.06    | 0.00   | 0.00   | 0.00   | 0.00   | 0.00   | 0.00   | 4,563.00 |
| <b>Total cash available</b>        | 500.00    | 1,200.00 | 481.80 | 481.80 | 3,887.11 | 3,238.29 | 2,903.16 | 496.69 | 496.69 | 496.69 | 496.69 | 496.69 | 496.69 |          |
| <b>CASH PAID OUT</b>               |           |          |        |        |          |          |          |        |        |        |        |        |        |          |
| Outreach                           |           |          |        |        |          |          |          |        |        |        |        |        |        | 0.00     |
| Networking and Trips               |           |          |        |        | 150.00   |          |          |        |        |        |        |        |        | 150.00   |
| Electrical Components              |           |          |        | 22.45  | 100.00   | 24.99    | 265.33   |        |        |        |        |        |        | 412.77   |
| Static Test Fire Rig               |           |          |        | 48.46  | 32.17    |          |          |        |        |        |        |        |        | 80.63    |
| Launch Pad                         |           |          |        |        | 23.99    |          |          |        |        |        |        |        |        | 23.99    |
| Aerodynamics                       |           |          |        | 9.15   |          | 75.13    | 120.60   |        |        |        |        |        |        | 204.88   |
| Rocket Engine                      |           |          |        | 362.71 | 292.66   | 273.07   | 2,011.65 |        |        |        |        |        |        | 0.00     |
| Reimbursements                     |           | 718.20   |        | 14.92  |          |          |          |        |        |        |        |        |        | 733.12   |
| Other Expenses                     |           |          |        |        | 50.00    |          | 8.89     |        |        |        |        |        |        | 58.89    |
| Pending Transactions               |           | 0.00     |        |        |          |          |          |        |        |        |        |        |        | 0.00     |
| <b>SUBTOTAL</b>                    |           | 718.20   | 0.00   | 457.69 | 648.82   | 373.19   | 2,406.47 | 0.00   | 0.00   | 0.00   | 0.00   | 0.00   | 0.00   | 4,604.37 |
| Owners' withdrawal                 |           |          |        |        |          |          |          |        |        |        |        |        |        | 0.00     |
| <b>TOTAL CASH PAID OUT</b>         |           | 718.20   | 0.00   | 457.69 | 648.82   | 373.19   | 2,406.47 | 0.00   | 0.00   | 0.00   | 0.00   | 0.00   | 0.00   | 4,604.37 |
| <b>Cash on hand (end of month)</b> | 500.00    | 481.80   | 481.80 | 24.11  | 3,238.29 | 2,865.10 | 496.69   | 496.69 | 496.69 | 496.69 | 496.69 | 496.69 | 496.69 |          |

### Detailed Cash Outflow

| Item                                   | Quantity | Cost      | Sub-Total   | Shipping  | Order Total | Lead Time (Days) | Source |
|----------------------------------------|----------|-----------|-------------|-----------|-------------|------------------|--------|
| Cessaroni - P29-3G Red Lighting (G125) | 1        | \$ 22.76  | \$ 22.76    |           |             | 7                | Apogee |
| Arrow Antenna 440-3                    | 1        | \$ 45.00  | \$ 45.00    | \$ 54.06  | \$ 76.82    | 7                | Apogee |
| MFJ-7736 Adapter                       | 1        | \$ 4.95   | \$ 4.95     |           |             |                  |        |
| Gorilla Tape - Black                   | 1        | \$ 8.89   | \$ 8.89     | \$ 111.18 | \$ 161.13   | 2                | Amazon |
| Nylon Fabric                           | 2        | \$ 8.95   | \$ 17.90    | \$ -      |             | 2                | Amazon |
| Braided Black Kevlar                   | 1        | \$ 39.99  | \$ 39.99    | \$ -      |             | 2                | Amazon |
| Fabri-Tac Permanent Adhesive           | 1        | \$ 11.34  | \$ 11.34    | \$ -      |             |                  |        |
| 54 mm Blue Tube                        | 3        | \$ 23.95  | \$ 71.85    |           |             | 7                | Apogee |
| 54 mm Blue Tube Coupler                | 1        | \$ 24.85  | \$ 24.85    |           |             |                  |        |
| PNC-2.14"                              | 2        | \$ 14.80  | \$ 29.60    |           |             |                  |        |
| Kevlar Cord (100ft)                    | 40       | \$ 0.97   | \$ 38.80    |           |             |                  |        |
| G10 FiberGlass Sheet 1/8"              | 2        | \$ 27.00  | \$ 54.00    |           |             |                  |        |
| Madcow 6inch Parachute Protector       | 5        | \$ 5.95   | \$ 29.75    |           |             |                  |        |
| 54mm Blue Ebay                         | 1        | \$ 33.95  | \$ 33.95    |           |             |                  |        |
| 29mm Blue Tube                         | 1        | \$ 12.49  | \$ 12.49    |           |             |                  |        |
| Tracking Powder                        | 1        | \$ 6.25   | \$ 6.25     |           |             |                  |        |
| TeleMega                               | 1        | \$ 428.00 | \$ 428.00   |           |             |                  |        |
| Starter Pack with TeleDongle           | 1        | \$ 107.00 | \$ 107.00   |           |             |                  |        |
| 58 inch Orange Shock Cord Protector    | 1        | \$ 15.99  | \$ 15.99    |           |             |                  |        |
| Eyebolts with washers and nuts         | 2        | \$ 9.46   | \$ 18.92    |           |             |                  |        |
| 220# Ball Bearing                      | 4        | \$ 16.80  | \$ 67.20    |           |             |                  |        |
| 1/4 inch Quick Link                    | 4        | \$ 15.76  | \$ 63.04    |           |             |                  |        |
|                                        |          |           | \$ 1,001.69 | \$ 286.89 |             |                  |        |
| P29 3G                                 | 3        | \$ 24.00  | \$ 72.00    |           |             | 2                | AMW    |
| P29 Closure                            | 3        | \$ 18.00  | \$ 54.00    |           |             |                  |        |
| Epoxy                                  | 1        | \$ 14.99  | \$ 14.99    |           |             |                  |        |
| Accelerometer                          | 1        | \$ 4.47   | \$ 4.47     | \$ 15.00  |             | 2                | Amazon |
|                                        |          |           | \$ -        | \$ -      |             |                  |        |

|                                         |   |                        |                       |                    |               |
|-----------------------------------------|---|------------------------|-----------------------|--------------------|---------------|
| 54mm LOC Body Tube                      | 5 | \$ 9.13                | \$ 45.65              | 7                  | Apogee        |
| 2.14" LOC Coupler                       | 4 | \$ 2.26                | \$ 9.04               |                    |               |
| 29mm Centering Rings                    | 5 | \$ 4.99                | \$ 24.95              |                    |               |
| Sunward Parachute Protector             | 2 | \$ 6.15                | \$ 12.30              |                    |               |
| 29mm Aft Closure                        | 1 | \$ 21.40               | \$ 21.40              |                    |               |
| 220# Ball Bearing                       | 4 | \$ 4.20                | \$ 16.80              |                    |               |
| Removable Plastic Rivets                | 1 | \$ 3.71                | \$ 3.71               |                    |               |
| Cessaroni - P29-3G Red Lighting (G125)  | 2 | \$ 22.76               | \$ 45.52              |                    |               |
| Cessaroni - P29-3G Red Lighting (G54)   | 2 | \$ 22.76               | \$ 45.52              |                    |               |
| 54mm Blue Ebay                          | 1 | \$ 33.95               | \$ 33.95              |                    |               |
| E-match Starter Kit                     | 1 | \$ 70.00               | \$ 70.00              |                    |               |
|                                         |   |                        | \$ -                  | \$ 100.50          |               |
| Arduino Nano                            | 1 | \$ 22.00               | \$ 22.00              | 2                  | Amazon        |
|                                         |   |                        | \$ -                  | \$ 3.69            |               |
| Cessaroni 29mm 6 Grain Case             | 1 | \$ 31.04               | \$ 31.04              | 7                  | Apogee        |
| Cessaroni - P29-3G White Thunder (H399) |   |                        | \$ -                  |                    |               |
| Cessaroni - P29-3G Red Lighting (I204)  |   |                        | \$ -                  |                    |               |
|                                         |   |                        | \$ -                  | \$ 64.08           |               |
| Gorilla Epoxy                           | 3 | \$ 4.96                | \$ 14.88              | 2                  | Amazon        |
| JB Weld                                 | 2 | \$ 4.40                | \$ 8.80               |                    |               |
| Clear Quik Cure Epoxy                   | 1 | \$ 9.42                | \$ 9.42               |                    |               |
|                                         |   |                        | \$ -                  |                    |               |
| Nylon Shear Pins                        | 1 | \$ 3.10                | \$ 3.10               | 7                  | Apogee        |
| Centering Rings 29mm                    | 3 | \$ 4.99                | \$ 14.97              |                    |               |
| Madcow 6inch Parachute Protector        | 2 | \$ 5.95                | \$ 11.90              |                    |               |
| 2.14" LOC Coupler                       | 2 | \$ 2.26                | \$ 4.52               |                    |               |
| 54mm Blue Tube                          | 1 | \$ 23.95               | \$ 23.95              |                    |               |
|                                         |   |                        | \$ 58.44              | \$ 15.60           |               |
| Parachute Recovery Systems              | 1 | \$ 49.95               | \$ 3.99               | 2                  | Amazon        |
| Assorted Bolts                          | 1 | \$ 20.00               | \$ 20.00              | 2                  | McMASTER-CARR |
| Assorted Washers                        | 1 | \$ 13.00               | \$ 13.00              |                    |               |
| Assorted Nuts                           | 1 | \$ 12.18               | \$ 12.18              |                    |               |
|                                         |   |                        | \$ -                  | \$ 13.08           |               |
| First Fire Jr. Starters                 | 2 | \$ 12.83               | \$ 25.66              | 2                  | Amazon        |
|                                         |   |                        | \$ -                  | \$ 8.56            |               |
| 54mm LOC Body Tube                      | 1 | \$ 9.13                | \$ 9.13               | 7                  | Apogee        |
| Terminal Blocks                         | 2 | \$ 3.41                | \$ 6.82               |                    |               |
|                                         |   |                        | \$ -                  | \$ 15.60           |               |
| Remove Before Flight Flag               | 1 | \$ 7.50                | \$ 7.50               | 2                  | Amazon        |
|                                         |   |                        | \$ -                  |                    |               |
| Switch                                  | 1 | \$ 6.50                | \$ 6.50               | 2                  | Amazon        |
|                                         |   |                        | \$ -                  | \$ -               |               |
| 2.14" LOC Stiffy                        | 1 | \$ 4.68                | \$ 4.68               | 7                  | Apogee        |
| RRc3 Altimeter                          | 1 | \$ 71.95               | \$ 71.95              |                    |               |
| USB Interface Module                    | 1 | \$ 32.95               | \$ 32.95              |                    |               |
|                                         |   |                        | \$ -                  | \$ 4.82            |               |
| Ester Solar Starter                     | 2 | \$ 6.07                | \$ 12.14              | 2                  | Amazon        |
|                                         |   |                        | \$ -                  | \$ -               |               |
| 54mm LOC Body Tube                      | 1 | \$ 9.13                | \$ 9.13               | 7                  | Apogee        |
| PNC 2.14"                               | 1 | \$ 14.80               | \$ 14.80              |                    |               |
| Centering Rings 29mm                    | 1 | \$ 4.99                | \$ 4.99               |                    |               |
| Epoxy Clay                              | 1 | \$ 12.55               | \$ 12.55              |                    |               |
| Cessaroni 29mm 3 Grain Case             | 1 | \$ 24.83               | \$ 24.83              |                    |               |
| Cessaroni 29mm Tapered Aft Closure      | 1 | \$ 22.76               | \$ 22.76              |                    |               |
| 1.4" Quick Link                         | 2 | \$ 3.94                | \$ 7.88               |                    |               |
| Madcow 6inch Parachute Protector        | 1 | \$ 5.95                | \$ 5.95               |                    |               |
|                                         |   |                        | \$ -                  | \$ 34.58           |               |
| Aluminum Round Tube                     | 1 | \$ 40.86               | \$ 40.86              | 2                  | McMASTER-CARR |
|                                         |   |                        | \$ -                  | \$ 6.32            |               |
| 1500mAh Lipo Battery                    | 1 | \$ 23.99               | \$ 23.99              | 2                  | Amazon        |
| Red Black Car Battery                   | 1 | \$ 6.35                | \$ 6.35               |                    |               |
|                                         |   |                        | \$ -                  | \$ -               |               |
| Swivel Leveling Mount                   | 3 | \$ 8.08                | \$ 24.24              | 2                  | McMASTER-CARR |
|                                         |   |                        | \$ -                  | \$ 6.06            |               |
| Low Carbon Steel Sheet                  | 1 | \$ 11.28               | \$ 11.28              | 2                  | McMASTER-CARR |
|                                         |   |                        | \$ -                  | \$ 6.01            |               |
| Galvanized Steel Corner Bracket         | 8 | \$ 4.46                | \$ 35.68              | 2                  | McMASTER-CARR |
|                                         |   |                        | \$ -                  | \$ 6.26            |               |
|                                         |   | <b>Total Good Cost</b> | <b>Total Shipping</b> | <b>Total Cost</b>  |               |
|                                         |   | <b>\$ 3,303.32</b>     | <b>\$ 752.29</b>      | <b>\$ 4,055.61</b> |               |

## References

[1] Borrowman, J. S. (1966). The Theoretical Prediction of the Center of Pressure. NARAM -8. NASA.

[2] *Definition of Scale Height*. (n.d.). Retrieved from Astronomy Education at the University of Nebraska: [http://astro.unl.edu/naap/scaleheight/sh\\_bg1.html](http://astro.unl.edu/naap/scaleheight/sh_bg1.html)

[3] Burr, A and Cheatham, J: *Mechanical Design and Analysis, 2nd edition*, section 5.2. Prentice-Hall, 1995

- [4] Knacke, T.W., *Parachute Recovery Systems Design Manual*. Santa Barbara, 1992
- [5] Westerfield, Mike. *Make: High-Power Rockets*. Maker Media, 2018.

## APPENDIX

### MATLAB Simulation Code

The following code details the functions used to accurately model our *Aether* class rockets. The code is easily applied to any two-stage rocket with a change in several inputs.

#### Overall Simulation Function

```
clear all;
close all;

% ~ AETHER6 Launch Simulation ~

L = 'linewidth';
D = 'displayname';
a = 0;
v = 0;           % initial velocity
h = 0.5;         % initial height
d = 0;           % initial drag
sf = 0;          % initial skin friction drag
pd = 0;          % initial pressure drag
bd = 0;          % initial base drag
tstart = 0;      % start time
dt = 0.01;       % time step
tstop = 160;     % endtime
tseparation = 2.3; % booster separation

%Initial Vectors
height = []; % Sustainer height
velocity = []; % Sustainer velocity
acceleration = []; % Sustainer acceleration
drag = []; % Sustainer drag
sfd = []; % Skin Friction Drag
pressured = []; % Pressure Drag
based = []; % Base Drag
x = []; % Used to keep track of time

for t = tstart:dt:tstop

    height(end + 1) = h;
    velocity(end + 1) = v;
    acceleration(end + 1) = a;
    drag(end + 1) = d;
    sfd(end + 1) = sf;
    pressured(end + 1) = pd;
    based(end + 1) = bd;
    x(end + 1) = t;
```

```

[a, d] = GetAcceleration(t,v,h); % get current acceleration
v = v + dt*a ; % update velocity
h = h + dt*v ; % update height

if h < 0
    break
end
end

% Booster (Starts Tracking at Separation)
% Initial Values and Vectors for Booster Separation

h = height(230); %(tseparation/dt);
v = velocity(230); %(tseparation/dt);
a = acceleration(230); %(tseparation/dt);
height2      = []; %height((tseparation-dt)/dt);
velocity2     = []; %velocity((tseparation-dt)/dt);
acceleration2 = []; %acceleration((tseparation-dt)/dt);
drag2        = []; %drag((tseparation-dt)/dt)
sfd2         = []; % Skin Friction Drag
pressured2    = []; % Pressure Drag
based2       = []; % Base Drag
x2 = []; %x((tseparation-dt)/dt);

for t = tseparation+dt:dt:tstop
    height2(end + 1)      = h;
    velocity2(end + 1)    = v;
    acceleration2(end + 1) = a;
    drag2(end + 1)        = d;
    sfd2(end + 1)         = sf;
    pressured2(end + 1)   = pd;
    based2(end + 1)       = bd;
    x2(end + 1)           = t;
    [a, d] = GetAcceleration2(v,h); % get current acceleration
    v = v + dt * a ; % update velocity
    h = h + dt * v ; % update height

    if h < 0
        break
    end
end
end

```

## Thrust

```

function [thrust] = GetThrust ( t )

% ~ AETHER4 ~

% burntimeboost = 1.3; % s
% burntimesust = 3; % s
% startimeboost = 2.3; % s (1.3 + 1 second delay)

% G125 - Booster Engine data (taken from thrustcurve.org)
xb_data = [0 0.01 0.025 0.03 0.037 0.044 0.055 0.1 0.19 0.27 0.4 0.94 1.05 1.13 1.19 1.22 1.3]; % s
yb_data = [0 5 155 169 160 127 118 140 148 152 151 126 125 108 40 20 0]; % N

```

```

% G54 - Sustainer Engine data (taken from thrustcurve.org)
xs_data = [0 0.018 0.031 0.059 0.135 0.22 0.299 0.432 0.959 1.757 2.418 2.851 2.98 3.0] + 2.3; % s
ys_data = [0 107.3 113.6 103.5 121.7 104.6 95.5 88.3 69.7 43.5 20.76 9.48 5.57 0]; % N

% uses data points to find the slope of the line between points and
% estimate the thrust as time iterates through the function

% ~ BOOSTER FIRES ~
if t > xb_data(1) && t <= xb_data(2)
    b1s = yb_data(2) - ((yb_data(2) - yb_data(1))/(xb_data(2) - xb_data(1)))*xb_data(2);
    thrust = ((yb_data(2) - yb_data(1))/(xb_data(2) - xb_data(1)))*t + b1s;

elseif t > xb_data(2) && t <= xb_data(3)
    b2s = yb_data(3) - ((yb_data(3) - yb_data(2))/(xb_data(3) - xb_data(2)))*xb_data(3);
    thrust = ((yb_data(3) - yb_data(2))/(xb_data(3) - xb_data(2)))*t + b2s;

elseif t > xb_data(3) && t <= xb_data(4)
    b3s = yb_data(4) - ((yb_data(4) - yb_data(3))/(xb_data(4) - xb_data(3)))*xb_data(4);
    thrust = ((yb_data(4) - yb_data(3))/(xb_data(4) - xb_data(3)))*t + b3s;

elseif t > xb_data(4) && t <= xb_data(5)
    b4s = yb_data(5) - ((yb_data(5) - yb_data(4))/(xb_data(5) - xb_data(4)))*xb_data(5);
    thrust = ((yb_data(5) - yb_data(4))/(xb_data(5) - xb_data(4)))*t + b4s;

elseif t > xb_data(5) && t <= xb_data(6)
    b5s = yb_data(6) - ((yb_data(6) - yb_data(5))/(xb_data(6) - xb_data(5)))*xb_data(6);
    thrust = ((yb_data(6) - yb_data(5))/(xb_data(6) - xb_data(5)))*t + b5s;

elseif t > xb_data(6) && t <= xb_data(7)
    b6s = yb_data(7) - ((yb_data(7) - yb_data(6))/(xb_data(7) - xb_data(6)))*xb_data(7);
    thrust = ((yb_data(7) - yb_data(6))/(xb_data(7) - xb_data(6)))*t + b6s;

elseif t > xb_data(7) && t <= xb_data(8)
    b7s = yb_data(8) - ((yb_data(8) - yb_data(7))/(xb_data(8) - xb_data(7)))*xb_data(8);
    thrust = ((yb_data(8) - yb_data(7))/(xb_data(8) - xb_data(7)))*t + b7s;

elseif t > xb_data(8) && t <= xb_data(9)
    b8s = yb_data(9) - ((yb_data(9) - yb_data(8))/(xb_data(9) - xb_data(8)))*xb_data(9);
    thrust = ((yb_data(9) - yb_data(8))/(xb_data(9) - xb_data(8)))*t + b8s;

elseif t > xb_data(9) && t <= xb_data(10)
    b9s = yb_data(10) - ((yb_data(10) - yb_data(9))/(xb_data(10) - xb_data(9)))*xb_data(10);
    thrust = ((yb_data(10) - yb_data(9))/(xb_data(10) - xb_data(9)))*t + b9s;

elseif t > xb_data(10) && t <= xb_data(11)
    b10s = yb_data(11) - ((yb_data(11) - yb_data(10))/(xb_data(11) - xb_data(10)))*xb_data(11);
    thrust = ((yb_data(11) - yb_data(10))/(xb_data(11) - xb_data(10)))*t + b10s;

elseif t > xb_data(11) && t <= xb_data(12)
    b11s = yb_data(12) - ((yb_data(12) - yb_data(11))/(xb_data(12) - xb_data(11)))*xb_data(12);
    thrust = ((yb_data(12) - yb_data(11))/(xb_data(12) - xb_data(11)))*t + b11s;

elseif t > xb_data(12) && t <= xb_data(13)
    b12s = yb_data(13) - ((yb_data(13) - yb_data(12))/(xb_data(13) - xb_data(12)))*xb_data(13);
    thrust = ((yb_data(13) - yb_data(12))/(xb_data(13) - xb_data(12)))*t + b12s;

elseif t > xb_data(13) && t <= xb_data(14)
    b13s = yb_data(14) - ((yb_data(14) - yb_data(13))/(xb_data(14) - xb_data(13)))*xb_data(14);
    thrust = ((yb_data(14) - yb_data(13))/(xb_data(14) - xb_data(13)))*t + b13s;

```

```

elseif t > xb_data(14) && t <= xb_data(15)
    b14s = yb_data(15) - ((yb_data(15) - yb_data(14))/(xb_data(15) - xb_data(14)))*xb_data(15);
    thrust = ((yb_data(15) - yb_data(14))/(xb_data(15) - xb_data(14)))*t + b14s;

elseif t > xb_data(15) && t <= xb_data(16)
    b15s = yb_data(16) - ((yb_data(16) - yb_data(15))/(xb_data(16) - xb_data(15)))*xb_data(16);
    thrust = ((yb_data(16) - yb_data(15))/(xb_data(16) - xb_data(15)))*t + b15s;

elseif t > xb_data(16) && t <= xb_data(17)
    b16s = yb_data(17) - ((yb_data(17) - yb_data(16))/(xb_data(17) - xb_data(16)))*xb_data(17);
    thrust = ((yb_data(17) - yb_data(16))/(xb_data(17) - xb_data(16)))*t + b16s;

% ~ SUSTAINER FIRES ~
elseif t > xs_data(1) && t <= xs_data(2)
    b1b = ys_data(2) - ((ys_data(2) - ys_data(1))/(xs_data(2) - xs_data(1)))*xs_data(2);
    thrust = ((ys_data(2) - ys_data(1))/(xs_data(2) - xs_data(1)))*t + b1b;

elseif t > xs_data(2) && t <= xs_data(3)
    b2b = ys_data(3) - ((ys_data(3) - ys_data(2))/(xs_data(3) - xs_data(2)))*xs_data(3);
    thrust = ((ys_data(3) - ys_data(2))/(xs_data(3) - xs_data(2)))*t + b2b;

elseif t > xs_data(3) && t <= xs_data(4)
    b3b = ys_data(4) - ((ys_data(4) - ys_data(3))/(xs_data(4) - xs_data(3)))*xs_data(4);
    thrust = ((ys_data(4) - ys_data(3))/(xs_data(4) - xs_data(3)))*t + b3b;

elseif t > xs_data(4) && t <= xs_data(5)
    b4b = ys_data(5) - ((ys_data(5) - ys_data(4))/(xs_data(5) - xs_data(4)))*xs_data(5);
    thrust = ((ys_data(5) - ys_data(4))/(xs_data(5) - xs_data(4)))*t + b4b;
elseif t > xs_data(5) && t <= xs_data(6)
    b5b = ys_data(6) - ((ys_data(6) - ys_data(5))/(xs_data(6) - xs_data(5)))*xs_data(6);
    thrust = ((ys_data(6) - ys_data(5))/(xs_data(6) - xs_data(5)))*t + b5b;

elseif t > xs_data(6) && t <= xs_data(7)
    b6b = ys_data(7) - ((ys_data(7) - ys_data(6))/(xs_data(7) - xs_data(6)))*xs_data(7);
    thrust = ((ys_data(7) - ys_data(6))/(xs_data(7) - xs_data(6)))*t + b6b;

elseif t > xs_data(7) && t <= xs_data(8)
    b7b = ys_data(8) - ((ys_data(8) - ys_data(7))/(xs_data(8) - xs_data(7)))*xs_data(8);
    thrust = ((ys_data(8) - ys_data(7))/(xs_data(8) - xs_data(7)))*t + b7b;

elseif t > xs_data(8) && t <= xs_data(9)
    b8b = ys_data(9) - ((ys_data(9) - ys_data(8))/(xs_data(9) - xs_data(8)))*xs_data(9);
    thrust = ((ys_data(9) - ys_data(8))/(xs_data(9) - xs_data(8)))*t + b8b;

elseif t > xs_data(9) && t <= xs_data(10)
    b9b = ys_data(10) - ((ys_data(10) - ys_data(9))/(xs_data(10) - xs_data(9)))*xs_data(10);
    thrust = ((ys_data(10) - ys_data(9))/(xs_data(10) - xs_data(9)))*t + b9b;

elseif t > xs_data(10) && t <= xs_data(11)
    b10b = ys_data(11) - ((ys_data(11) - ys_data(10))/(xs_data(11) - xs_data(10)))*xs_data(11);
    thrust = ((ys_data(11) - ys_data(10))/(xs_data(11) - xs_data(10)))*t + b10b;

elseif t > xs_data(11) && t <= xs_data(12)
    b11b = ys_data(12) - ((ys_data(12) - ys_data(11))/(xs_data(12) - xs_data(11)))*xs_data(12);
    thrust = ((ys_data(12) - ys_data(11))/(xs_data(12) - xs_data(11)))*t + b11b;

elseif t > xs_data(12) && t <= xs_data(13)
    b12b = ys_data(13) - ((ys_data(13) - ys_data(12))/(xs_data(13) - xs_data(12)))*xs_data(13);
    thrust = ((ys_data(13) - ys_data(12))/(xs_data(13) - xs_data(12)))*t + b12b;

```

```

elseif t > xs_data(13) && t <= xs_data(14)
    b13b = ys_data(14) - ((ys_data(14) - ys_data(13))/(xs_data(14) - xs_data(13)))*xs_data(14);
    thrust = ((ys_data(14) - ys_data(13))/(xs_data(14) - xs_data(13)))*t + b13b;
else
    thrust = 0;
end

end

end

```

## Mass

```

function [mass] = GetMass ( t )

% ~ AETHER4 ~

Mnosecone           = .111;
Mshoulder           = .130;
Mebay               = .155;
Mbattery            = .000;
Msustbodytube       = .105;
Mforwardfins        = .123;
Mstagingcoupler     = .058;
Mboosterbodytube    = .145;
Maftfins            = .079;
Msustcasingtuberetainer = .101; % Mass of the Engine Casing, Engine Tube and Retainer
Mboostcasingtuberetainer = .101; % Mass of the Engine Casing, Engine Tube and Retainer
Mboostinit          = .198; % Initial mass of booster
Msustinit           = .194; % Initial mass of sustainer
Mboostprop          = .082; % Mass of propellant
Msustprop           = .086; % Mass of propellant

Mdrogueparachute    = .024;
Mmainparachute      = .071;
Mboosterparachute   = .036;

initialMass = Mboostinit + Msustinit + Mnosecone + Mshoulder + Mebay + Mbattery + Msustbodytube + Mforwardfins
+ Msustcasingtuberetainer...
+ Mstagingcoupler + Mboosterbodytube + Mboostcasingtuberetainer + Maftfins +...
+ Mdrogueparachute + Mmainparachute + Mboosterparachute;
initialSustMass = Msustinit + Mnosecone + Mshoulder + Mebay + Msustbodytube + Msustcasingtuberetainer...
+ Mforwardfins + Mdrogueparachute + Mmainparachute;

burnTimeBoost = 1.3; % sec
burnTimeSust = 3; % sec
startTimeboost = 2.3; % sec
startTimecoast = startTimeboost + burnTimeSust;

if (t>=0 && t<burnTimeBoost)
    mass = initialMass - Mboostprop *(t/burnTimeBoost);
elseif (t>=burnTimeBoost && t<startTimeboost)
    mass = initialMass - Mboostprop;
elseif (t>=startTimeboost && t< startTimeboost + burnTimeSust)
    mass = initialSustMass - Msustprop * (t/(burnTimeSust + startTimeboost));
elseif (t>startTimecoast)

```



```

        mass = initialSustMass - Msustprop;
    else
        mass = initialMass;
    end

end

end

```

### *Drag of Sustainer*

```

function [drag] = GetDrag2 (v,h)
% calculates total drag forces for booster after seperation

% ~ AETHER6 ~

pi = 3.14;
mu = 1.79e-5;
% NB ! When v is + ve (up) drag should be + ve (down)
rho = 1.217*exp(-h/8500);
kv = mu/rho;
l = 0.45; % length of booster (m)

if v < 0
    D = .50; % parachute diameter (m)
    k = 1.0; % parachute drag coefficient
    drag = -k * 0.5 * rho * v^2 * pi/4 * D^2;
else
    % ~ SKIN FRICTION DRAG ~
    D = .0574; % (m) diameter of body tube
    r = D/2;
    Re = (v*l)/kv;
    Rs = 60e-6; % roughness "Incorrectly sprayed aircraft paint"
    Rcrit = 51*(Rs/l)^-1.039;

    if Re < 1.0e4
        CDsf = 1.48e-2;
    elseif Re > 1.0e4 && Re < Rcrit
        CDsf = 1/(1.50*log(Re) - 5.6)^2;
    elseif Re > Rcrit
        CDsf = 0.032*(Rs/l)^0.2;
    end

    % BOOSTER FINS
    fb = l/D; % fineness ratio
    t_b = 0.003; % fin thickness
    c_b = 0.056; % aerodynamic cord length
    finbase_b = 0.056;
    finheight_b = 0.126;
    finarea_sfb = 0.5*finbase_b*finheight_b;
    fin_totalarea_sfb = finarea_sfb*6;
    bodytube_area = 2*pi*r*l;
    crossection_area = pi*r^2;
    ref_area_sf = fin_totalarea_sfb + bodytube_area;

    % Compressibility Effects

```

```

a = 343; %m/s
M = v/a;
if M < 0.9
    Cfc = Cdsf*(1 - 0.1*M^2);
elseif M >= 0.9
    Cfc = Cdsf/((1 + 0.15*M^2)^0.58);
end

Cdsf = Cfc*((1 + 1/(2*fb))*(bodytube_area) + (1 + (2*t_b)/c_b)*fin_totalarea_sfb)/ref_area_sf;

Fdrag_sf = Cdsf*(1/2)*rho*v^2*(ref_area_sf); % force of skin friction drag

% ~ PRESSURE DRAG ~
% No nose cone; pressure drag on cross-section approximated as a flat
% plate
CDpd_top = 1; % flate plate coefficient
Fpd_top = CDpd_top*(1/2)*rho*v^2*(crosssection_area);

% Fin Pressure Drag
% BOOSTER FINS
LEAb = 61.2; % leading edge angle
if M < 0.9
    CDpd_finsb = ((1 - M^2)^(-0.417) - 1)*cosd(LEAb)^2;
elseif M > 0.9 && M < 1
    CDpd_finsb = (1 - 1.785*(M - 0.9))*cosd(LEAb)^2;
elseif M > 1.0
    CDpd_finsb = (1.214 - 0.502/(M^2) + 0.1095/(M^4))*cosd(LEAb)^2;
end

Fpd_finsb = CDpd_finsb*(1/2)*rho*v^2*(finbase_b*t_b);

Fpd_fins_totb = Fpd_finsb*3;

Fdrag_pd = Fpd_top + Fpd_fins_totb; % force of pressure drag

% ~ BASE DRAG ~

if M < 1
    Cdbd = 0.12 + 0.13*M^2 ;
elseif M > 1
    Cdbd = 0.25/M;
end

Fdrag_bd = Cdbd*(1/2)*rho*v^2*(crosssection_area); % force of base drag

% ~ TOTAL DRAG ~
drag = Fdrag_sf + Fdrag_pd + Fdrag_bd; % total force of drag

end

end

```

### *Drag of Booster and Sustainer*

```

function [drag] = GetDrag (v,h,t)
% Calculates total drag for the sustainer and booster; carries to

```

```

% sustainer then to recovery.

% Both fin sets are accounted for.

% NB ! When v is + ve (up) drag should be + ve (down)

% ~ AETHER6 ~

pi = 3.14;
mu = 1.79e-5;
rho = 1.217*exp(-h/8500);
kv = mu/rho;

if v >= 0

    % FULL ROCKET
    if t <= 2.3
        l = 1.25; % length of full rocket (m)

        % ~ SKIN FRICTION DRAG ~
        D = .0574; % (m) Diameter of Nosecone
        r = D/2;
        Re = (v*l)/kv;
        Rs = 60e-6; % roughness
        Rcrit = 51*(Rs/l)^-1.039;

        if Re < 1.0e4
            CDSf = 1.48e-2;
        elseif Re > 1.0e4 && Re < Rcrit
            CDSf = 1/(1.50*log(Re) - 5.6)^2;
        elseif Re > Rcrit
            CDSf = 0.032*(Rs/l)^0.2;
        end

        fb = l/D; % fineness ratio

        % BOOSTER FINS
        t_b = 0.003; % booster fin thickness
        c_b = 0.056; % booster fin aerodynamic cord length
        finbase_booster = 0.056;
        finheight_booster = 0.126;
        finarea_sfb = 0.5*finbase_booster*finheight_booster;
        fin_totalarea_sfb = finarea_sfb*6; % booster fin area
        % SUSTAINER FINS
        t_s = 0.003; % sust fin thickness
        c_s = 0.056; % sust fin aerodynamic cord length
        finbase_sust = 0.056;
        finheight_sust = 0.126;
        finarea_sfs = 0.5*finbase_sust*finheight_sust;
        fin_totalarea_sfs = finarea_sfs*6; % sust fin area
        % BODY TUBE
        bodytube_area = 2*pi*r*l; % surface area
        crosssection_area = pi*r^2; % cross-sectional area
        % EFFECTIVE AREA
        ref_area_sf = fin_totalarea_sfb + fin_totalarea_sfs + bodytube_area;

        % Compressibility Effects
        a = 343; %m/s
        M = v/a;
        if M < 0.9

```

```

        Cfc = Cdsf*(1 - 0.1*M^2);
    elseif M >= 0.9
        Cfc = Cdsf/((1 + 0.15*M^2)^0.58);
    end

    Cdsf = Cfc*((1 + 1/(2*fb))*(bodytube_area) + (1 + (2*t_b)/c_b)*fin_totalarea_sfb + (1 +
(2*t_s)/c_s)*fin_totalarea_sfs)/ref_area_sf;

    Fdrag_sf = Cdsf*(1/2)*rho*v^2*(ref_area_sf); % force of skin friction drag

% ~ PRESSURE DRAG ~
% Nose Cone Pressure Drag
% if M is less than 0.9...
% CDpd_nc = 0.05; % ogive nose cone
CDpd_nc = 0;
Fpd_nosecone = CDpd_nc*(1/2)*rho*v^2*(crosssection_area);

% Fin Pressure Drag

% BOOSTER FINS
LEAb = 61.2; % leading edge angle
if M < 0.9
    CDpd_finsb = ((1 - M^2)^(-0.417) - 1)*cosd(LEAb)^2;
elseif M > 0.9 && M < 1
    CDpd_finsb = (1 - 1.785*(M - 0.9))*cosd(LEAb)^2;
elseif M > 1.0
    CDpd_finsb = (1.214 - 0.502/(M^2) + 0.1095/(M^4))*cosd(LEAb)^2;
end

Fpd_finsb = CDpd_finsb*(1/2)*rho*v^2*(finbase_booster*t_b);

Fpd_fins_totalb = Fpd_finsb*3;

% SUSTAINER FINS
LEAs = 61.2; % leading edge angle
if M < 0.9
    CDpd_finss = ((1 - M^2)^(-0.417) - 1)*cosd(LEAs)^2;
elseif M > 0.9 && M < 1
    CDpd_finss = (1 - 1.785*(M - 0.9))*cosd(LEAs)^2;
elseif M > 1.0
    CDpd_finss = (1.214 - 0.502/(M^2) + 0.1095/(M^4))*cosd(LEAs)^2;
end

Fpd_finss = CDpd_finss*(1/2)*rho*v^2*(finbase_sust*t_s);

Fpd_fins_totals = Fpd_finss*3;

Fdrag_pd = Fpd_nosecone + Fpd_fins_totalb + Fpd_fins_totals; % force of pressure drag

% ~ BASE DRAG ~

if M < 1
    CDbd = 0.12 + 0.13*M^2 ;
elseif M > 1
    CDbd = 0.25/M;
end

Fdrag_bd = CDbd*(1/2)*rho*v^2*(crosssection_area); % force of base drag

% ~ TOTAL DRAG ~
drag = Fdrag_sf + Fdrag_pd + Fdrag_bd; % total force of drag

```

```

% SUSTAINER STAGE ONLY
elseif t > 2.3
    l = 0.80; % length of sustainer body tube (m)

    % ~ SKIN FRICTION DRAG ~
    D = .0574; % (m) Diameter of Nosecone
    r = D/2;
    Re = (v*l)/kv;
    Rs = 60e-6; % roughness
    Rcrit = 51*(Rs/l)^-1.039;

    if Re < 1.0e4
        CDSf = 1.48e-2;
    elseif Re > 1.0e4 && Re < Rcrit
        CDSf = 1/(1.50*log(Re) - 5.6)^2;
    elseif Re > Rcrit
        CDSf = 0.032*(Rs/l)^0.2;
    end

    fb = l/D; % fineness ratio

    % SUSTAINER FINS
    t_s = 0.003; % sust fin thickness
    c_s = 0.056; % sust fin aerodynamic cord length
    finbase_sust = 0.056;
    finheight_sust = 0.126;
    finarea_sfs = 0.5*finbase_sust*finheight_sust;
    fin_totalarea_sfs = finarea_sfs*6; % sust fin area
    % BODY TUBE
    bodytube_area = 2*pi*r*l; % surface area
    crosssection_area = pi*r^2; % cross-sectional area
    % EFFECTIVE AREA
    ref_area_sf = fin_totalarea_sfs + bodytube_area;

    % Compressibility Effects
    a = 343; %m/s
    M = v/a;
    if M < 0.9
        Cfc = CDSf*(1 - 0.1*M^2);
    elseif M >= 0.9
        Cfc = CDSf/((1 + 0.15*M^2)^0.58);
    end

    Cdsf = Cfc*((1 + 1/(2*fb))*(bodytube_area) + (1 + (2*t_s)/c_s)*fin_totalarea_sfs)/ref_area_sf;

    Fdrag_sf = Cdsf*(1/2)*rho*v^2*(ref_area_sf); % force of skin friction drag

    % ~ PRESSURE DRAG ~
    % Nose Cone Pressure Drag
    % if M is less than 0.9...
    % CDpd_nc = 0.05; % ogive nose cone
    CDpd_nc = 0;
    Fpd_nosecone = CDpd_nc*(1/2)*rho*v^2*(crosssection_area);

    % Fin Pressure Drag

    % SUSTAINER FINS

```

```

LEAS = 61.2; % leading edge angle
if M < 0.9
    CDpd_finss = ((1 - M^2)^(-0.417) - 1)*cosd(LEAS)^2;
elseif M > 0.9 && M < 1
    CDpd_finss = (1 - 1.785*(M - 0.9))*cosd(LEAS)^2;
elseif M > 1.0
    CDpd_finss = (1.214 - 0.502/(M^2) + 0.1095/(M^4))*cosd(LEAS)^2;
end

Fpd_finss = CDpd_finss*(1/2)*rho*v^2*(finbase_sust*t_s);

Fpd_fins_totals = Fpd_finss*3;

Fdrag_pd = Fpd_nosecone + Fpd_fins_totals; % force of pressure drag

% ~ BASE DRAG ~

if M < 1
    CDbd = 0.12 + 0.13*M^2 ;
elseif M > 1
    CDbd = 0.25/M;
end

Fdrag_bd = CDbd*(1/2)*rho*v^2*(crossection_area); % force of base drag

% ~ TOTAL DRAG ~
drag = Fdrag_sf + Fdrag_pd + Fdrag_bd; % total force of drag

end

% ~ RECOVERY ~
elseif v < 0
    D = 0.30; % diameter of parachute (m)
    k = 1.0;
    drag = -k * 0.5 * rho * v^2 * pi/4 * D^2;

    if h < 90
        D = 0.75;
        k = 1.0;
        drag = -k * 0.5 * rho * v^2 * pi/4 * D^2;
    end

end

end

end

```

### *Acceleration of Sustainer*

```

% ~ AETHER4 ~

function [a, d] = GetAcceleration (t , v, h )
g = 9.81;
f = GetThrust ( t );
m = GetMass ( t );
[d] = GetDrag ( v,h,t );

```

```
a = ( f - m*g - d ) / m ; % Newton Second Law
end
```

### Acceleration of Booster

```
% Achieve acceleration of the booster after separation

% ~ AETHER4 ~

function [a, d] = GetAcceleration2 ( v,h )

% Mass of Booster parts
Mstagingcoupler = .058; % Mass of Staging Coupler
Mboosterbodytube = .145; % Body tube and two centering rings
Mafftins = .079; % Mass of the 3 aft fins
Mboostinit = .198; % Initial mass of booster
Mboostprop = .082; % Mass of booster propellant
Mcasingtuberetainer = .101; % Mass of the Engine Casing, Engine Tube and Retainer

g = 9.81; % Gravity
through flight (constant < 10,000 feet)
f = 0; % No
upwards thrust
m = Mstagingcoupler + Mboosterbodytube + Mafftins + Mboostinit - Mboostprop + Mcasingtuberetainer; % Mass of
falling booster
[d] = GetDrag2 ( v,h ); % Drag
of booster through flight
a = ( f - m*g - d ) / m ; % Newton's Second Law %
Acceleration equation for a falling body with drag
end
```

### Caliber Function Code

```
% Calculated the CG, CP and plots the caliber of a 2 stage rocket

% Mass in kilograms (kg)
% Distance in meters (m)
% Time in seconds (s)

% Time Resolution
res = 1000;

% Booster and Sustainer Time Values
Tboost = 1.3;
Tsust = 3.0;

% Lengths: In Meters (m)
% Main Components:
Lnosecone = .240;
Lshoulder = .310;
```

```

Lebay          = .200;
Lsustbodytube  = .330;
Lforwardfins   = .130;
Lstagingcoupler = .135;
Lboosterbodytube = .448;
Laftfins       = .115;
Lboost         = .187;
Lsust          = .187;

% Internal electronics (m)
Lbattery       = 0.0265;
%Laltimeter    = 0.0996;
%LSEDSaltimeter = 0.06985;
%Lswitch       = 0.01;

% Internal pieces (m)
Ldrogueparachute = 0.04;
Lmainparachute   = 0.08;
Lboosterparachute = 0.06;

% Coupler BodyTube Pieces
LcouplerBodyTubeEbay = .025; %(m)
LcouplerBodyTubeStage = .025;
% Motor Hang
sustmotorhang = .0458;
boostmotorhang = .0359;

% Distances: Centroid Calculations from the Tip of Nosecone to the middle of the component

% Main Components:
Dnosecone      = Lnosecone*.666;
Dshoulder      = Lnosecone + Lshoulder/2;
Debay          = Dshoulder + Lshoulder/2 + LcouplerBodyTubeEbay/2;
Dsustbodytube  = Debay + LcouplerBodyTubeEbay/2 + Lsustbodytube/2;
Dforwardfins   = Dsustbodytube + Lsustbodytube/2 - Lforwardfins/2 - .015;
Dstagingcoupler = Dsustbodytube + Lsustbodytube/2 + LcouplerBodyTubeStage/2;
Dsust          = Dstagingcoupler - LcouplerBodyTubeStage/2 - Lsust/2 + sustmotorhang;
Dboosterbodytube = Dstagingcoupler + LcouplerBodyTubeStage/2 + Lboosterbodytube/2;
Daftfins       = Dstagingcoupler + LcouplerBodyTubeStage/2 + Lboosterbodytube - Laftfins/2 - .005;
Dboost         = Dstagingcoupler + LcouplerBodyTubeStage/2 + Lboosterbodytube - Lboost/2 + boostmotorhang;

% Internal Components: Marks the beginning of the Component
xbattery       = 0.02;    % From forward of E-Bay Bulk Plate
%xaltimeter    = 0.03;    % From forward of E-Bay Bulk Plate
%xswitch       = 0.02;    % From forward of E-Bay Bulk Plate
%xSEDSaltimeter = 0.22;    % From forward of Nosecone

xmainparachute = 0.12;     % From forward Shoulder Forward
xdroguenparachute = 0.10; % From aft of E-Bay Bulk Plate
xboosterparachute = 0.12 - .085; % From aft of Coupler Bulk Plate

```



```

%xcenteringring1    = 0.25;    % From aft of Sustainer Body Tube (-)
%xcenteringring2    = 0.08;    % From aft of Sustainer Body Tube (-)
%xcenteringring3    = 0.01;    % From aft of Booster Body Tube (-)
%xcenteringring4    = 0.12;    % From aft of Booster Body Tube (-)

% Internal Electronics:
Dbattery            = Debay - Lebay/2 + xbattery + Lbattery/2;    % Offset from E-Bay Forward Bulk
Plate
%Daltimeter         = Lnosecone + xaltimeter + Laltimeter/2;    % Offset from E-Bay Forward Bulk Plate
%Dswitch            = Lnosecone + xswitch + Lswitch/2;    % Offset from E-Bay Forward Bulk Plate
%DSEDSaltimeter     = xSEDSaltimeter + LSEDSaltimeter/2;    % Offset from Nose Cone Forward

% Recovery Components:

Ddrogueparachute    = Debay + Lebay/2 + xdrogueparachute + Ldrogueparachute/2 - .07;
% Offset from Nosecone Forward
Dmainparachute      = Lnosecone + xmainparachute + Lmainparachute/2;
% Offset from E-bay Aft
Dboosterparachute   = Dstagingcoupler + LcouplerBodyTubeStage/2 + Lstagingcoupler/2 +
xboosterparachute + Lboosterparachute/2;    % Offset from Staging Coupler Aft

% Engine Retainment:
%Dcenteringring1    = Dsustbodytube + Lsustbodytube/2 - xcenteringring1;    % Offset from
Sustainer Body Tube Aft
%Dcenteringring2    = Dsustbodytube + Lsustbodytube/2 - xcenteringring2;    % Offset from
Sustainer Body Tube Aft
%Dcenteringring3    = Dboosterbodytube + Lsustbodytube/2 - xcenteringring3;    % Offset from
Booster Body Tube Aft
%Dcenteringring4    = Dboosterbodytube + Lsustbodytube/2 - xcenteringring4;    % Offset from
Booster Body Tube Aft

% Masses (kg)
% Mass of Components
Mnosecone          = .113;
Mshoulder           = .125;
Mebay               = .401; % with a payload of 4 bolts at 180 grams total

Msustbodytube       = .080;
Mforwardfins        = .030;
Mstagingcoupler     = .083;
Mboosterbodytube    = .114;
Maftfins            = .030;
Mboostinit          = .194 + .077;    % Initial mass of booster
Msustinit           = .198 + .081;    % Initial mass of sustainer
Mboostprop          = .082;    % Mass of total propellant
Msustprop           = .087;    % Mass of total propellant

% Internal electronics (kg)
Mbattery            = 0.045;
Maltimeter          = 0.000;
MSEDSaltimeter      = 0.000;

```

```

Mswitch      = 0.000;

% Internal pieces (kg)
Mdrogueparachute = .025;
Mmainparachute   = .078;
Mboosterparachute = .043;

Mcenteringring = 0;

% Create a vector of values of the Center of Gravity through the entire flight

% Create array CG that follows the whole
% rocket from launch till booster seperation

tb      = linspace(0,Tboost,res);          % 1000 different times of flight from 0 to .7
seconds
Mboost   = Mboostinit-(Mboostprop/Tboost)*tb; % Mass of the booster during flight
CGBoost  = Mboost.*Dboost;                  % Center of Gravity of booster during flight

Mtot     = Mboost + Msustinit + Mnosecone + Mshoulder + Mebay + Msustbodytube + Mforwardfins +
Mstagingcoupler + Mboosterbodytube...      % Total mass of sustainer and
      + Maftfins + Mbattery + Maltimeter + MSEDSaltimeter + Mswitch + Mdrogueparachute +
Mboosterparachute + Mmainparachute + 4*Mcenteringring; % booster through flight

CGboostsust = (CGBoost + Dnosecone*Mnosecone + Dshoulder*Mshoulder + Debay*Mebay +
Dsustbodytube*Msustbodytube + Dforwardfins*Mforwardfins... % Center of gravity of entire
rocket
      + Dstagingcoupler*Mstagingcoupler + Dsust*Msustinit + Dboosterbodytube*Mboosterbodytube +
Daftfins*Maftfins + Mbattery*Dbattery... % through flight before seperation
      + Mdrogueparachute*Ddrogueparachute + Mboosterparachute*Dboosterparachute...
      + Mmainparachute*Dmainparachute)./Mtot;

% Create vector CGsust that follows the sustainer after booster
% seperation until sustainer burnout when mass stops changing

ts      = linspace(0,Tsust,res);          % 1000 different times of flight from 0 to 1.4
seconds
Msust   = Msustinit - (Msustprop/Tsust)*ts; % Mass of sustainer during flight
CGSust  = Msust.*Dsust;                  % Center of Gravity of sustainer during flight

Mtot2   = Msust + Mnosecone + Mshoulder + Mebay + Msustbodytube + Mforwardfins + Mbattery +
Maltimeter... % Total mass of sustainer through flight
      + MSEDSaltimeter + Mswitch + Mdrogueparachute + Mmainparachute + 2*Mcenteringring;
% after booster seperation

CG2      = (CGSust + Dnosecone*Mnosecone + Dshoulder*Mshoulder + Debay*Mebay + Mbattery*Dbattery +
Dsustbodytube*Msustbodytube + Dforwardfins*Mforwardfins... % Center of gravity of sustainer
through flight
      + Mdrogueparachute*Ddrogueparachute + Mmainparachute*Dmainparachute)./Mtot2;

% CP (m) Calculate Center of Pressure of the entire rocket and after booster seperation

% Fin parameters

```

```

Cr = .130; % Length of root chord
Ct = .015; % Length of tip chord
Ss = .050; % Length of semi-span
Lf = .055; % Length of mid-chord line
Xr = .100; % Length of fin root lead to fin tip lead
Xb = Dforwardfins - Lforwardfins/2; % Length of nosecone tip to beginning of root chord

Rbodytube = .0574/2; % Radius of Bodytube

cnn = 2; % Co-efficient for the type of nose cone - conical
xn = .466 * Lnosecone; % Location of the center of pressure for a conical nose cone
cnt = 0; % cnt would change if rocket diameter changed
xt = 0; % Not applicable
nf = 3; % Number of Fins

% Fin Calculations
x1=1.0+(Rbodytube/(Ss+Rbodytube)); % The following
variables allow cnf, a coefficient, to be calculated
x2=4.0*nf*(Ss*Ss/(Rbodytube*Rbodytube^4));
x3a=2.0*Lf;
x3b=Cr+Ct;
x3=x3a/x3b;
x4=1.0+sqrt(1.0+x3^2);
cnf=x1*x2/x4; % A coefficient
needed to find center of pressure
xf = (1.0/6.0)*(Cr+Ct-(Cr*Ct/(Cr+Ct)))+(Xr/3.0)*((Cr+2.0*Ct)/(Cr+Ct))+Xb; % cnf and xf are
coefficients that take the parameters of the fins. and factors in the % distance from the
tip of the nose cone to the root chord of the fin
% Calculation for CP (sustainer)
cnr=cnn+cnt+cnf;
cp = ((cnn*xn+cnt*xt+cnf*xf)/cnr); % Center of Pressure equation for a rocket

% Booster & Sustainer

% Fin Parameters
Cr2 = .115; % Length of root chord
Ct2 = .060; % Length of tip chord
Ss2 = .048; % Length of semi-span
Lf2 = .050; % Length of mid-chord line
Xr2 = .045; % Length of fin root lead to fin tip lead

Xb2 = Daftfins - Laftfins/2 - .02; % Length of nosecone tip to beginning of aft fins
root chord

% Equations for CP (full rocket)
x12=1.0+(Rbodytube/(Ss2+Rbodytube)); % The following variables allow cnf, a
coefficient, to be calculated
x22=4.0*nf*(Ss2*Ss2/(Rbodytube*Rbodytube^4));
x3a2=2.0*Lf2;
x3b2=Cr2+Ct2;
x32=x3a2/x3b2;

```

```

x42=1.0+sqrt(1.0+x32*x32);

cnf2=x12*x22/x42; % A
coefficient needed to find center of pressure
xf2 = (1.0/6.0)*(Cr2+Ct2-(Cr2*Ct2/(Cr2+Ct2)))+(Xr2/3.0)*((Cr2+2.0*Ct2)/(Cr2+Ct2))+Xb2; % cnf
and xf are coefficeints that take the parameters of the fins. and factors in the %
distance from the tip of the nose cone to the root chord of the fin

% Calculation for CP (sustainer)
cnr2=cnn+cnt+cnf+cnf2;
cp2 = ((cnn*xn+cnt*xt+cnf*xf+cnf2*xf2)/cnr2); % Center of Pressure equation for a rocket

%
% Plotting Caliber

%BoosterOpenRocket = 'CalOpenRocketBooster.csv';
%SustainerOpenRocket = 'CalOpenRocketSustainer.csv';

L = 'linewidth';
D = 'displayname';

figure;
cal1 = (cp2-CGboostsust)/(2*Rbodytube);
plot(tb,cal1,D,'Full Rocket',L,2)
cal2 = (cp-CG2)/(2*Rbodytube);
hold on
coast = .3; % Coast period between seperation
ts = linspace(Tboost+coast,Tsust+Tboost,res); % 1000 different times of flight from 0 to
1.4 seconds
xsep=[1.3,1.3,1.6];
ysep=[2.597,0.7994,0.7994];
plot(xsep,ysep,'--',D,'Seperation',L,2)
plot(ts,cal2,'r',D,'Sustainer',L,2) % Plot caliber through the flight
xlabel('Time (s)','fontsize',14)
ylabel('Caliber','fontsize',14)
title('Aether 4 In-Flight Stabilty','fontsize',14)
leg=legend('show')
set(leg,'fontsize',12)
axis([0 4.5 0.5 3])
grid minor

```