UNH SEDS
Technical Manufacturing Report

September 2018

Charlie Nitschelm, Reilly Webb, Thomas Collins,

Grace Johnston, Charles Gould, Ross Thyne,

Ester Yee, Silas Johnson

Advisor: Doctor Todd Gross

Table of Contents

LIST OF FIGURES **3**
LIST OF TABLES **4**
LIST OF EQUATIONS **4**
INTRODUCTION **6**
ABSTRACT 6
OBJECTIVES OF UNH SEDS 6
UNIVERSITY STUDENT ROCKETRY COMPETITION 6
Goals *7*
Constraints *7*
OVERALL ROCKET CONFIGURATION AND CONCEPT OF OPERATIONS 7
SAFETY AND LAUNCH PROCEDURE CHECKLIST **9**
LAUNCH OPERATIONS AND SAFETY PRECAUTION CHECKLIST 9
TECHNICAL DESCRIPTION OF VEHICLE ANALYSIS **11**
STATIC TEST FIRE RIG 11
Test Results *11*
Test Comparisons *16*
FLIGHT TRAJECTORY 18
Aerodynamic Models *18*
Assumptions 18
Program Structure 19
Model Verification 25
NONLINEAR OPTIMIZATION 27
Constraints *28*
VEHICLE ANALYSIS OF COMPETITION ROCKET 29
Launch Optimization and Flight Path of Final Rocket *29*
BUILD PROCESS VALIDATION **31**
AETHER CLASS ITERATIONS LEADING TO FINAL COMPETITION ROCKET 31
Aether I *31*
Aether II *31*
Aether III *31*
Aether IV *32*
Aether V *32*
Aether VI *32*
AETHER VII – COMPETITION ROCKET 33
Booster Build Process *33*
Sustainer Build Process *35*
ELECTRONICS DOCUMENTATION **37**
ONBOARD RECOVERY ELECTRONICS 37
ELECTRONICS TEST PLAN AND PRELIMINARY RESULTS 38
ISSUES ENCOUNTERED DURING THE MANUFACTURING PROCESS **39**
MATERIAL CHOICE 39
ENGINE CHOICE 39

PARACHUTE CHOICE 40
LAUNCH FAILURE MITIGATION 41
IMPROVEMENT CYCLE FOR FAILURE MITIGATION 41
FAILURE MITIGATION SAFETY CHECKS 42
UPDATED LAUNCH DATES 44
AVAILABLE LAUNCH DATES UNTIL OCTOBER 12TH: 44
CURRENT COMPLETION STATUS 44
REFERENCES 46
APPENDIX 46
MATLAB SIMULATION CODE 46
Overall Simulation Function 46
Thrust 47
Mass 50
Drag of Sustainer 51
Drag of Booster and Sustainer 53
Acceleration of Sustainer 57
Acceleration of Booster 57
CALIBER FUNCTION CODE 57

List of Figures

Figure 1. Rocket Components 8
Figure 2. STFR test of booster engine 12
Figure 3. Load cell calibration 13
Figure 4. Booster Engine Response 14
Figure 5. Sustainer Engine Response 15
Figure 6. Engine Impulse vs Time 16
Figure 7. Experimental Data vs Cesaroni Supplied Data 17
Figure 8 Trapezoidal Fin Model 20
Figure 9. Aether IV Caliber 21
Figure 10. Aether IV Launch Simulation 25
Figure 11: Model verification using Aether IV rocket 26
Figure 12. Optimization of Aether VI Dimensions 28
Figure 13. Convergence of Aether VIII Dimensions 29
Figure 14. Flight Profile of Rocket Sustainer 30

Figure 15. Booster Body Tube: Bottom View 34

Figure 16. Booster Body Tube and Coupler 35

Figure 17. Electronic Bay Sled: TeleMega Side 36

Figure 18. Electronic Bay Sled: SEDS Altimeter Side 36

Figure 19. 3D Printed Nose Cone Side View 36

Figure 20. 3D Printed Nose Cone Top-Down View 37

Figure 21. Sustainer Body Tube and Coupler 37

Figure 22. Wiring Schematics of the Telemega GPS 38

Figure 23. Improvement Cycle 41

Figure 24. FEA of the Nosecone during Flight 43

Figure 25. FEA of the Centering Rings during Booster Thrust 43

Figure 26. Booster Body Tube and Coupler 44

Figure 27. Booster Body Tube and Coupler 45

List of Tables

Table 1. Surface Roughness 23

Table 2. Aether IV Apogee Predictions 26

List of Equations

Equation 1 19

Equation 2 19

Equation 3 20

Equation 4 21

Equation 5 22

Equation 6 22

Equation 7 22

Equation 8 23

Equation 9 23

Equation 10 23

Equation 11 24

Equation 12 24

Equation 13 ... 24
Equation 14 ... 24
Equation 15 ... 25
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 to give a detailed view of taking our final rocket design into manufacturing, the problems encountered, and a greater detailed view in ensuring a safe, reliable flight.

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 was 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.

The 2017-2018 school year and the beginning of the Fall semester of 2018 was dedicated to design a rocket to compete in the University Student Rocketry Competition in October 2018. Students from all classes and majors dedicated themselves to learn the insider view of high power rocketry to be able to place with some of the top rocketry team in the country.

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 fully 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 12th, 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.

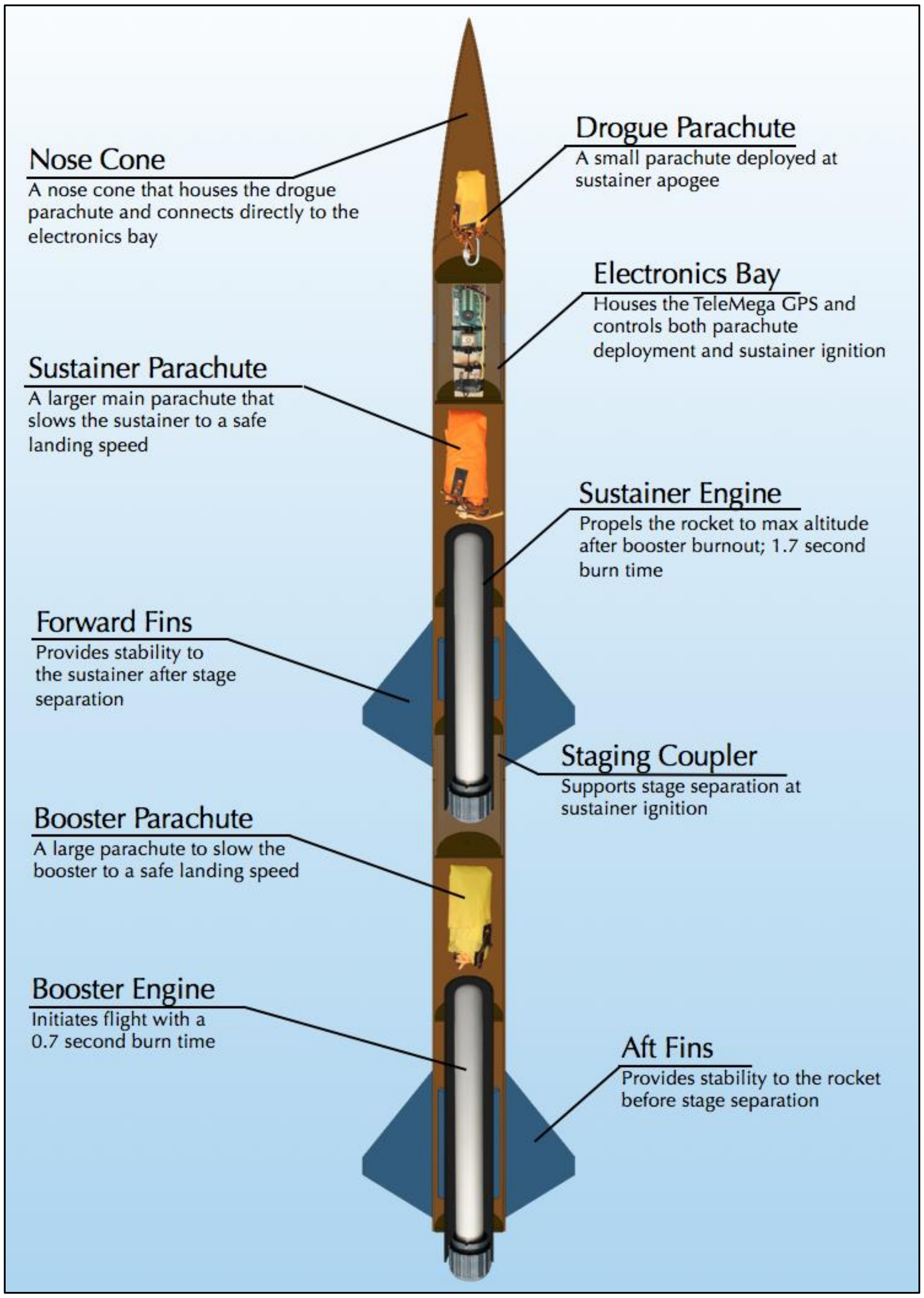

Nose Cone
A nose cone that houses the drogue parachute and connects directly to the electronics bay
Drogue Parachute
A small parachute deployed at sustainer apogee
Electronics Bay
Houses the TeleMega GPS and controls both parachute deployment and sustainer ignition
Sustainer Parachute
A larger main parachute that slows the sustainer to a safe landing speed
Sustainer Engine
Propels the rocket to max altitude after booster burnout; 1.7 second burn time
Forward Fins
Provides stability to the sustainer after stage separation
Staging Coupler
Supports stage separation at sustainer ignition
Booster Parachute
A large parachute to slow the booster to a safe landing speed
Booster Engine
Initiates flight with a 0.7 second burn time
Aft Fins
Provides stability to the rocket before stage separation

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 be 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 decent 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.

Safety and Launch Procedure Checklist

Launch Operations and Safety Precaution Checklist

1. Assemble Launch Pad in desired location given the rocket flight path, wind direction and field orientation
    - 1.1. If the wind velocity is significant on launch day, a calculated launch angle may be used to compensate for the drift the rocket in the limited size of the field
    - 1.2. Set up the launch rail stopper to raise the rocket roughly 6 inches above the launch pad base
    - 1.3. Move the rocket into position on the stopper, sliding the two buttons into the t-slots on the rail
2. Turn on the Telemega GPS to begin satellite connection and flight configuration
    - 2.1. Ensure battery is fully charged and beeps when switch is flipped on
3. Turn on Altimeter by connecting the terminals on the side
    - 3.1. Ensure correct beeps upon startup
4. Insert the booster engine assembly into the booster
    - 4.1. Ensure that there are no possible situations that can ignite the booster engine when configuring on the launch pad
5. Screw rocket aft retainer onto engine tube
6. Remove yellow safety cap to begin igniter installation

- 6.1. 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
- 6.2. All other members and observers must be 200 feet away from the rocket

7. Install igniter leads into engine by inserting through the nozzle and up the grain until it hits the top
8. Screw engine casing retainer onto the booster engine tube to constrain the engine axially
9. Attach igniter leads onto the launch system alligator leads.
    - 9.1. Check that the other igniter launch leads are not connected to the battery and the launch director has the launch key
10. Insert the sustainer engine assembly into sustainer engine tube.
    - 10.1. Ensure that there are no possible situations that can ignite the sustainer engine when configuring on the launch pad
11. Screw rocket after retainer onto engine tube
12. Remove yellow safety cap to being igniter installation
13. Install pre-installed igniter leads from e-bay into the engine.
    - 13.1. Insure proper lead connections between the e-bay and ignitor wire
14. Install igniter leads into engine by inserting through the nozzle and up the grain until it reaches the top
15. Screw engine casing retainer onto the sustainer engine tube to constrain the engine in axially
16. Re-attach sustainer, coupler, and booster tubes.
17. Perform final checks on the rocket
    - 17.1. Fin alignment between stages
    - 17.2. Proper retainment on all engines
    - 17.3. Correct pin installations if needed

18. Run from launch pad (It is UNH SEDS tradition to always run from the launch pad before countdown)
19. Verify TeleMega connection, continuity on all igniters and GPS tracking
20. Setup launch controller to the battery for launch
21. Verify surroundings and safety of all people
    - 21.1. Launch safety officer must give the okay for launch
22. Check continuity
23. Countdown --> Launch
24. If Launch is unsuccessful, disconnect leads from battery
    - 24.1. Disconnect alligator clips from ignitor leads

24.2. Remove ignitor leads from engine

24.3. WAIT 60 seconds

24.4. Problem solve the issue, and start from step 7

Technical Description of Vehicle Analysis

The culmination of our research in engine performance, flight simulation, rocket manufacturing and part optimization enabled the team to make informed decisions on the final design of our rocket. The manufacturing and assembly of multi-stage high powered rockets that were mainly self-taught during the 2017-2018 school year allowed us to dive deep into a comprehensive simulation of rocket flight and optimization that worked hand-in-hand with the constraints of manufacturing and assembly. The first three sections of the 'Technical Description of Vehicle Analysis' guide the reader through the work that needed to occur for us to accurately design our rocket with little to no problems during the manufacturing and assembly phase. Once the background work involved is understood, a detailed description of our end results are explained on how we finally selected our end dimensions and the benefits/downsides of each part.

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 2). 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.

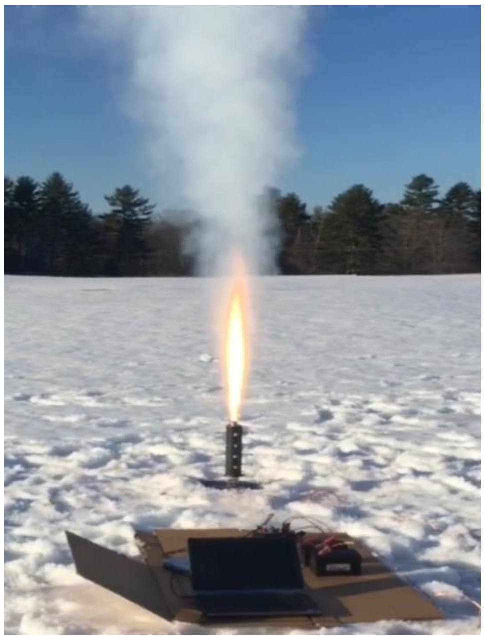


Figure 2. STFR test of booster engine

The test setup consisted of a custom STFR, two 12V batteries, a 250 lb. load cell, an AD620 Amplifier, two 100 kΩ 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 3).

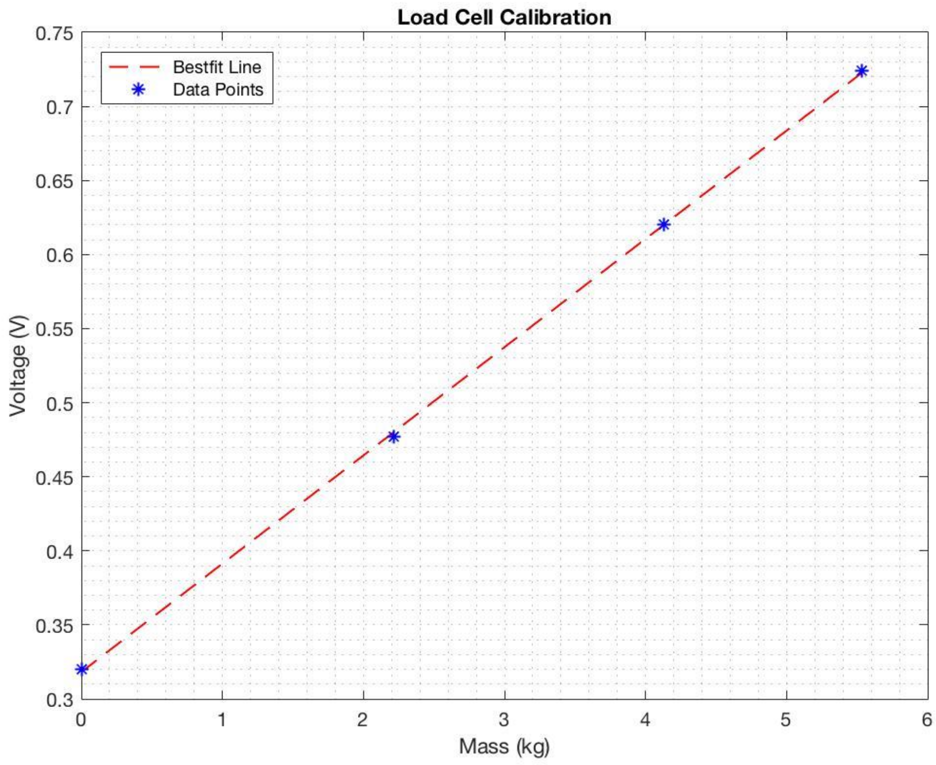


Figure 3. Load cell calibration

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 4.

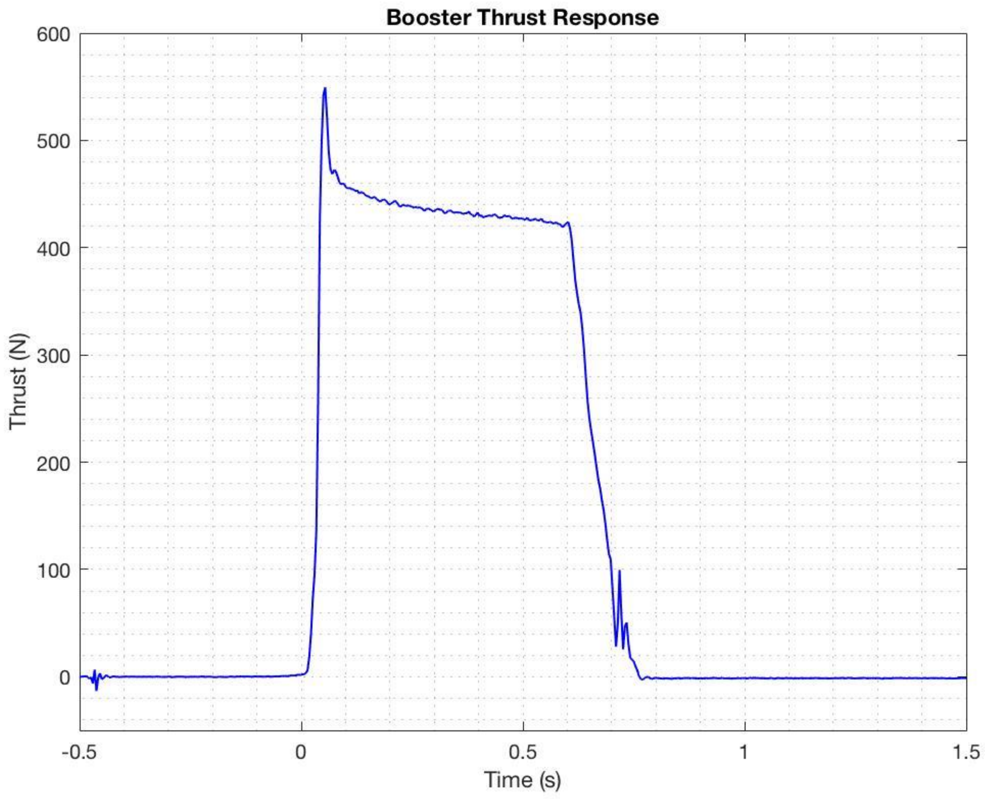


Figure 4. Booster Engine Response

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 5, was not quite as accurate.

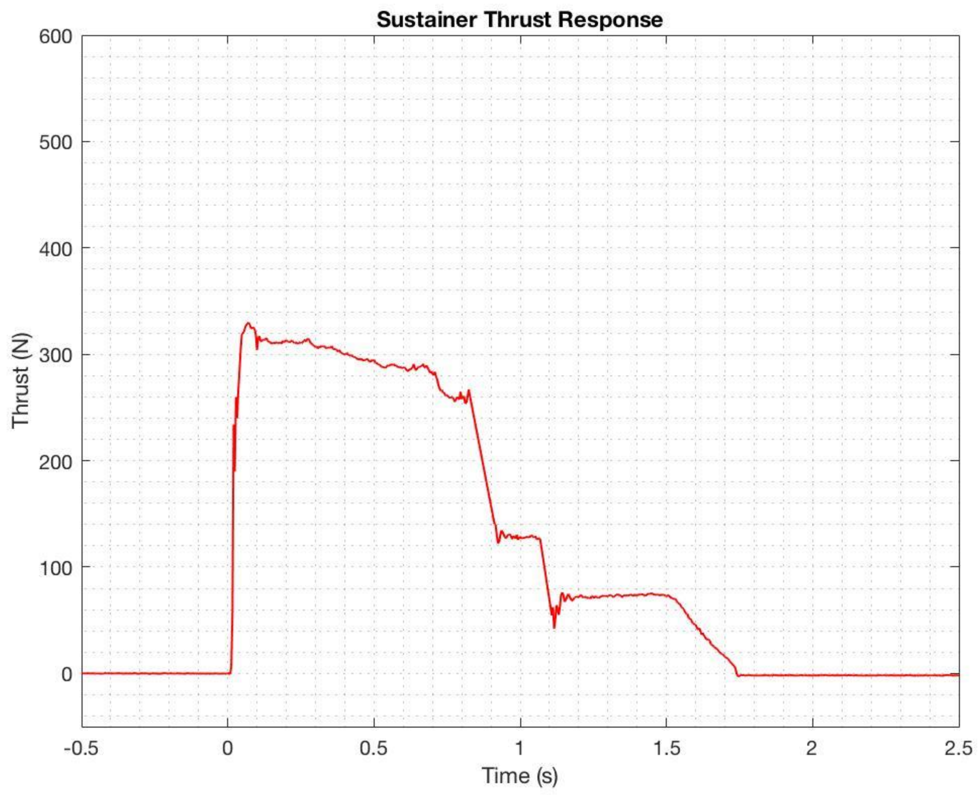


Figure 5. Sustainer Engine Response

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 6 shows the measured impulse of both engine types.

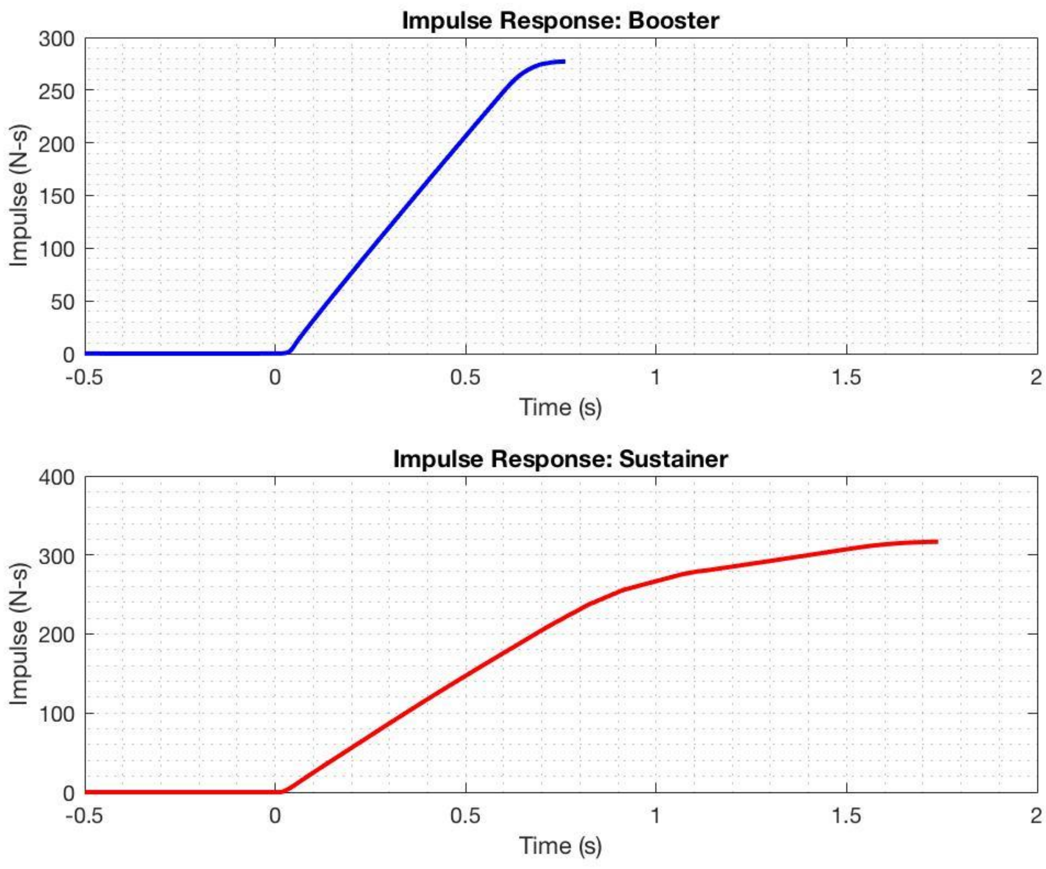


Figure 6. Engine Impulse vs Time

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 7.

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.

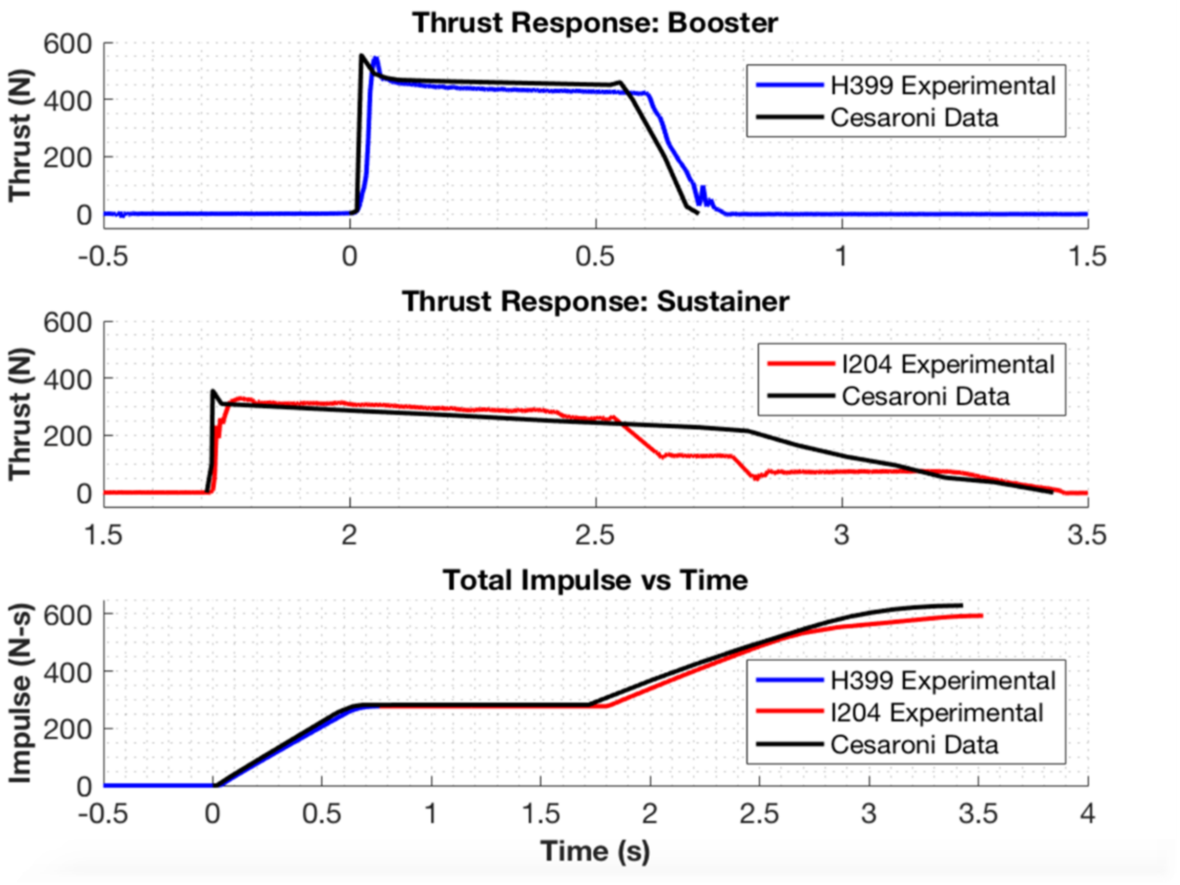


Figure 7. Experimental Data vs Cesaroni Supplied Data

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 7 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

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 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 logic 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

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

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. The 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 Ct is zero (see Figure 8). Square fins can also be represented as Xr goes to zero and Ct=Cr.

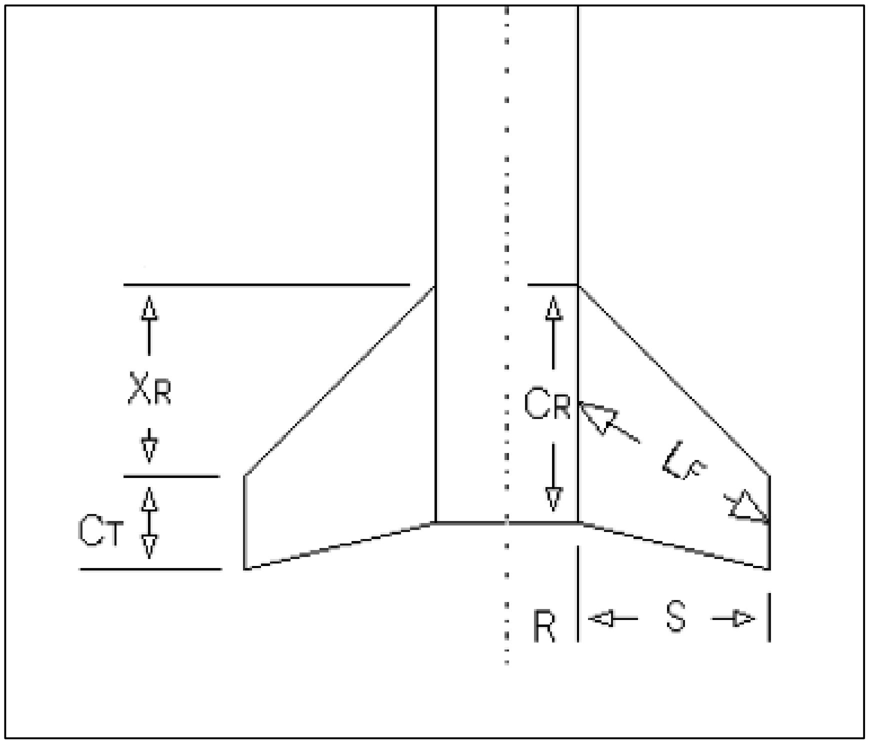


Figure 8 Trapezoidal Fin Model

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

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

Where all variables are defined in Figure 8 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 9).

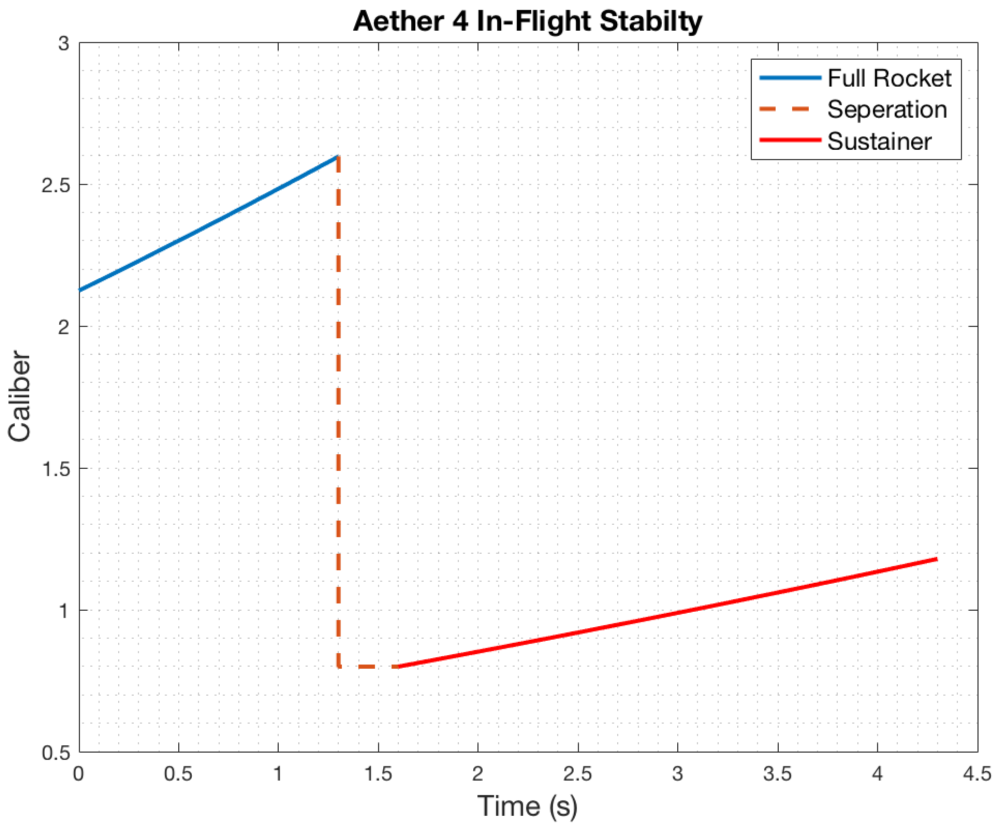


Figure 9. Aether IV Caliber

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 2nd 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

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

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 a 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 ρ_o is the air density at sea level (1.225 kg/m^3), 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).

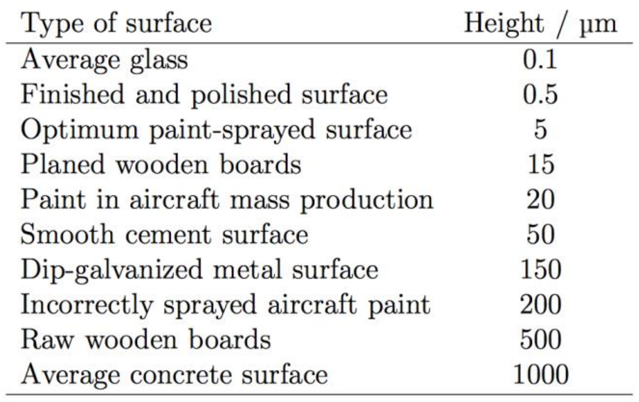
The following equations 8-12 for drag were taken from Barrowman, 1966 **[1]**.

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 / µm
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

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

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

$$if\ R_e < 10^4:\ \ C_{sf} = 1.48x10^{-2}$$

$$if\ 10^4 < R_e < R_{crit}:\ \ C_{sf} = \frac{1}{(1.5lnR - 5.6)^2}$$

$$if\ R_e > R_{crit}:\ \ C_{sf} = 0.032(\frac{R_s}{L})^2$$

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

$$for\ M < 0.9:\ \ C_D = [(1 - M^2)^{-0.417} - 1]cos^2(LEA)$$

$$for\ 0.9 < M < 1:\ \ C_D = [1 - 1.785(M - 0.9)]cos^2(LEA)$$

$$for\ M > 1:\ \ C_D = \left[1.214 - \frac{0.502}{M^2} + \frac{0.1095}{M^4}\right]cos^2(LEA)$$

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

Equation 12

$$for\ M < 1:\ \ C_D = 0.12 + 0.13M^2$$

$$for\ M > 1:\ \ C_D = \frac{0.25}{M}$$

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

$$v_i = v_{i-1} + \Delta t * a_i$$

$$h_i = h_{i-1} + \Delta t * h_i$$

A time step resolution of $\Delta t = 0.01$ was found to be acceptable, as the results converged to a consistent solution. Figure 10 demonstrates a completed simulation for the *Aether IV* design.

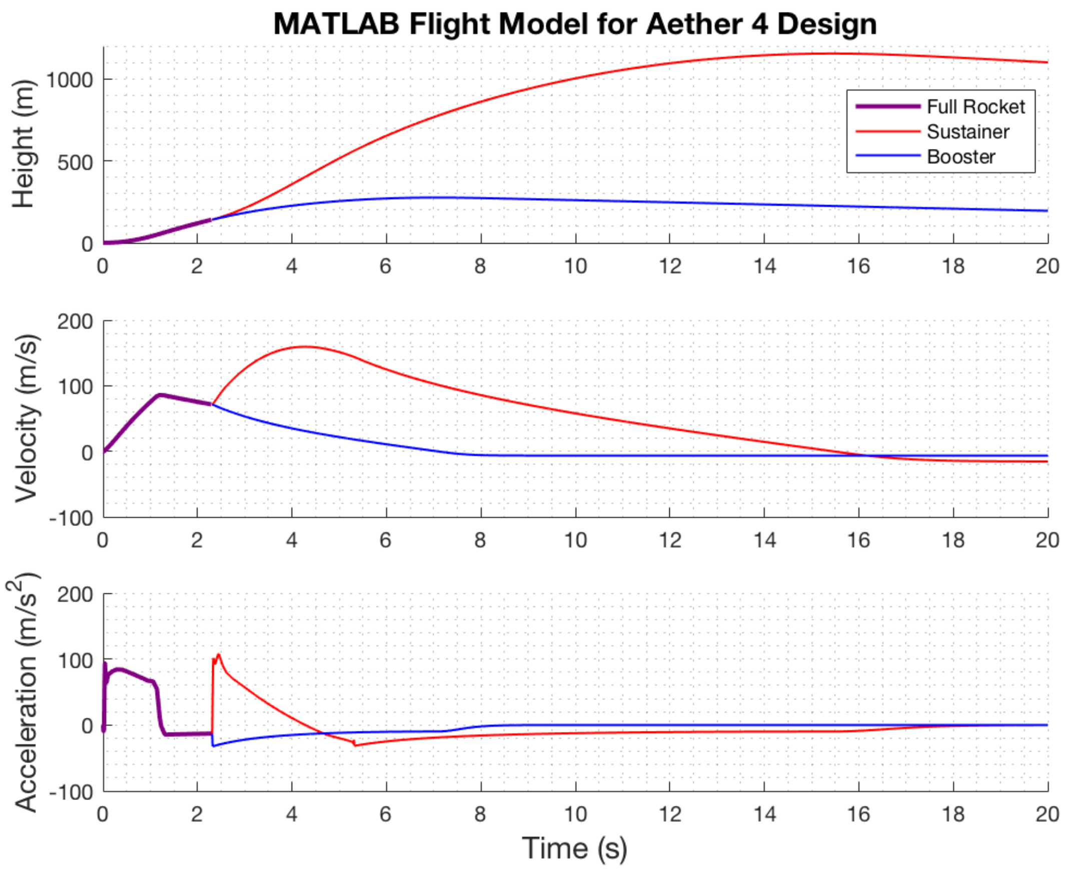


Figure 10. Aether IV Launch Simulation

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 11.

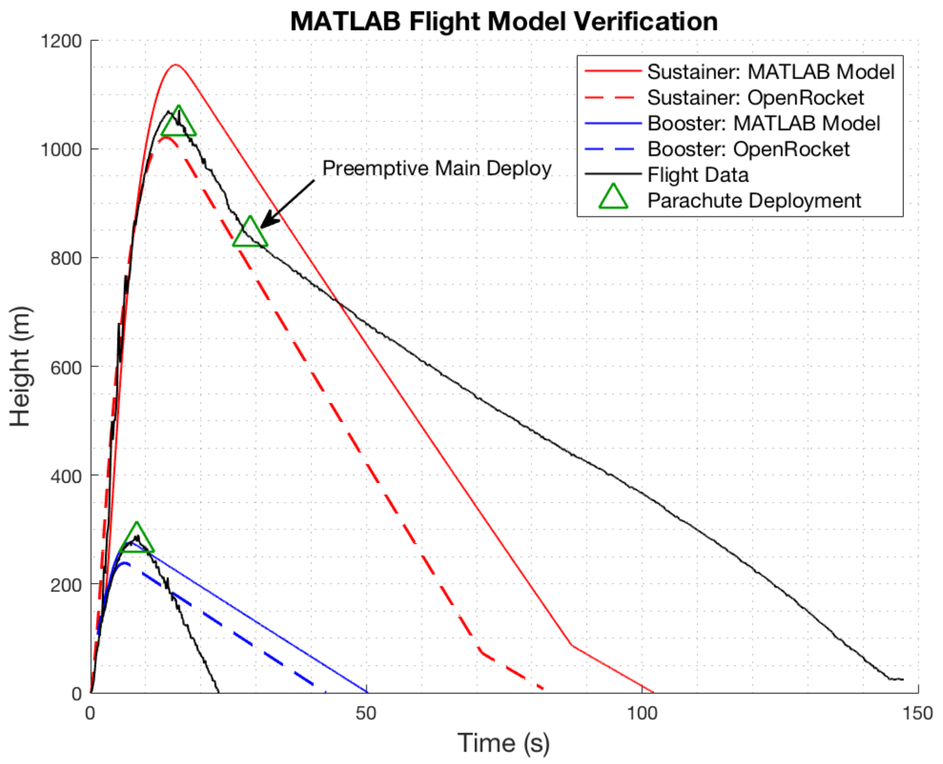


Figure 11: Model verification using Aether IV rocket

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 **Error! Reference source not found.** 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**
Sustainer Apogee	1071.1 m	1020.6 m	1154.1 m
Booster Apogee	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.

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 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 12.

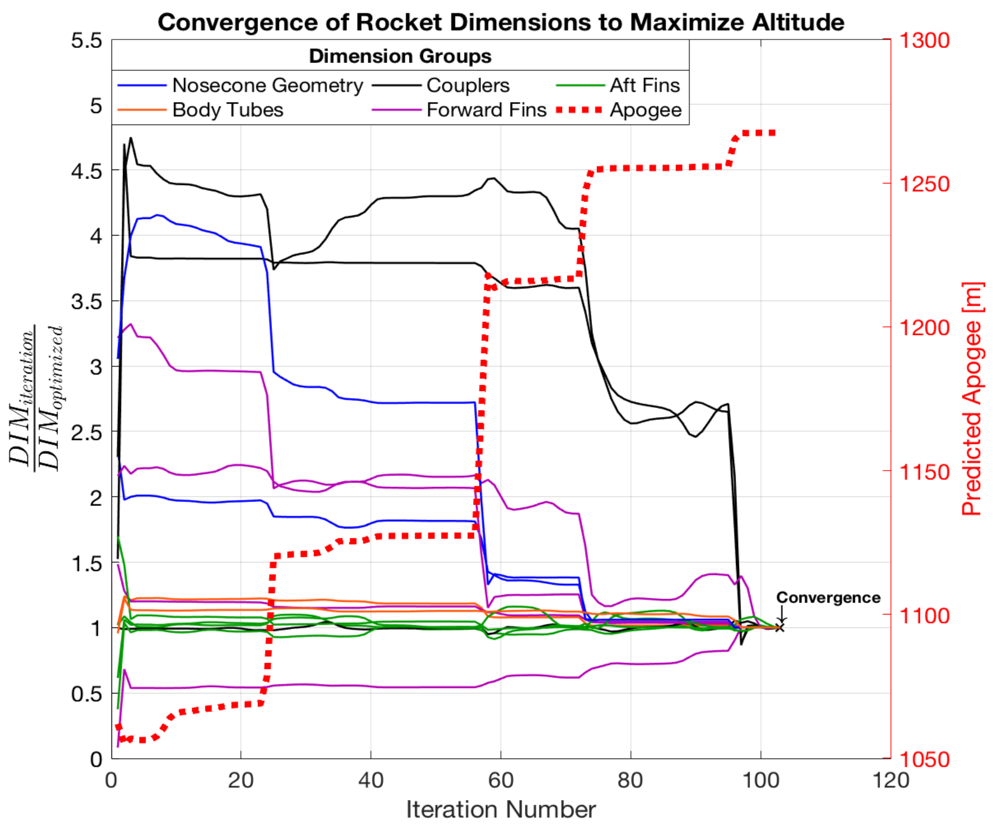


Figure 12. Optimization of Aether VI Dimensions

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.

Vehicle Analysis of Competition Rocket

With the overview of how we have been able to develop the expertise, data and software to confidently design the competition rocket with the constraints of manufacturing and assembly in mind, we are able to detail the reasons of our design, flight mechanics, and the compromises that needed to be made to mitigate potential failure with the decrease in maximum altitude.

Launch Optimization and Flight Path of Final Rocket

The convergence of our final competition rocket dimensions is illustrated below. With iterating different dimensions for each part, and calculating its maximum altitude, we are able to validate our initial guess for optimal dimensions to a dimension backed by our MATLAB software. All the drag theory, flight mechanics and engine thrust values drove the decisions on every dimension of our rocket, set with constraints based on real-world limitations.

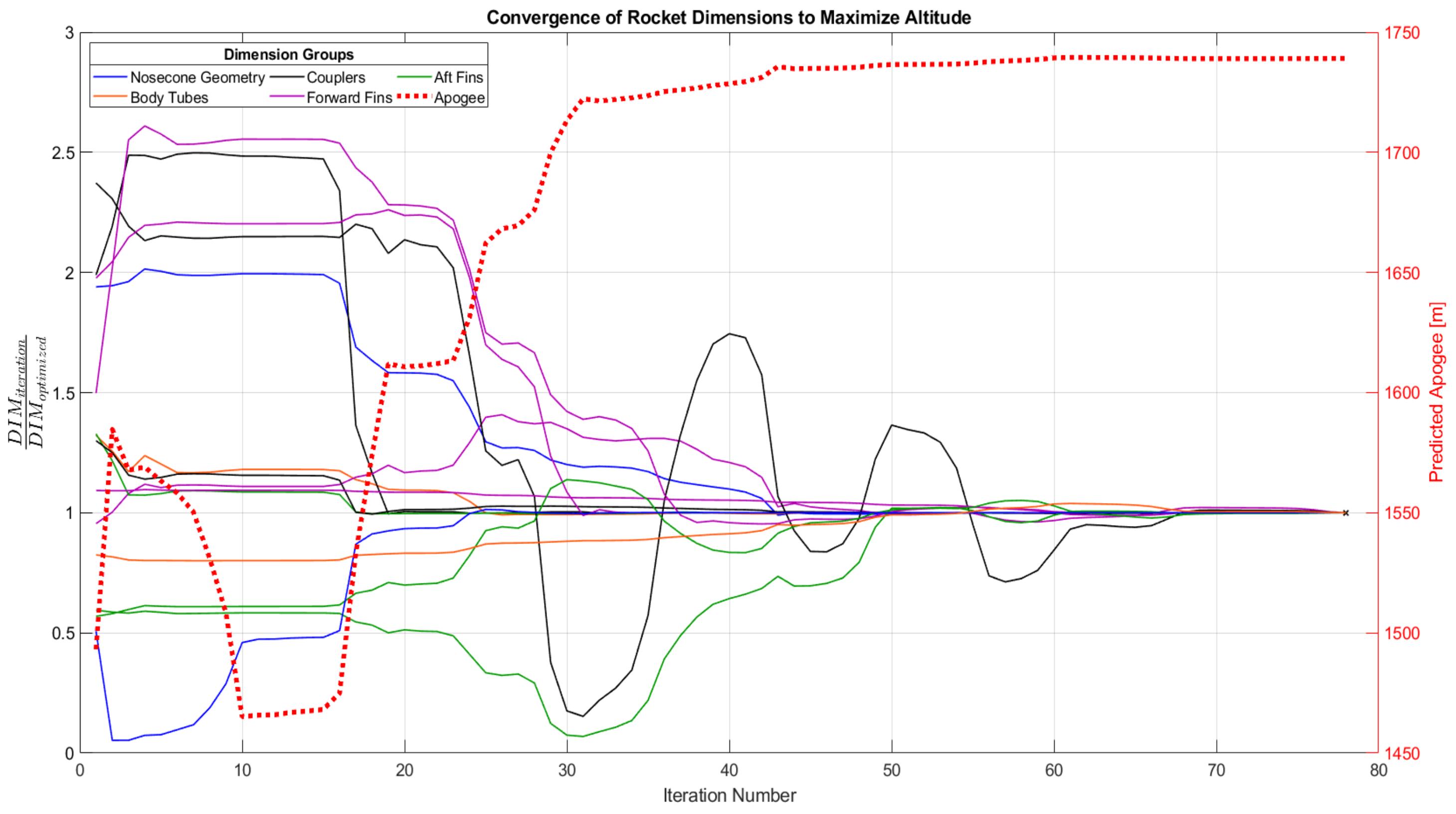


Figure 13. Convergence of Aether VII Dimensions

Based on past flight experience, the most prone-to-failure parts of our Aether class rockets are the fins, which works directly with our body and engine tubes, as well as the centering rings. With that, we decided to further strengthen our rocket as we approach faster speeds by moving to carbon fiber. With every material, there are benefits and downsides. The benefits of using a composite such as carbon fiber are clear: robust material properties, relatively cheap, and with a wide selection of material online. The downsides are apparent, and have affected us directly: it is difficult to manufacture the raw materials to size. We overcame the downsides to ensure that the sensitive parts of the rocket are the strongest they can be. Both the centering rings and fins interface directly with the carbon fiber body tubes. The centering rings were made of thick acrylic used heavier epoxy fillets. The fins were able to be 3D printed with fiberglass reinforcement after to mitigate the threat of fracture upon landing.

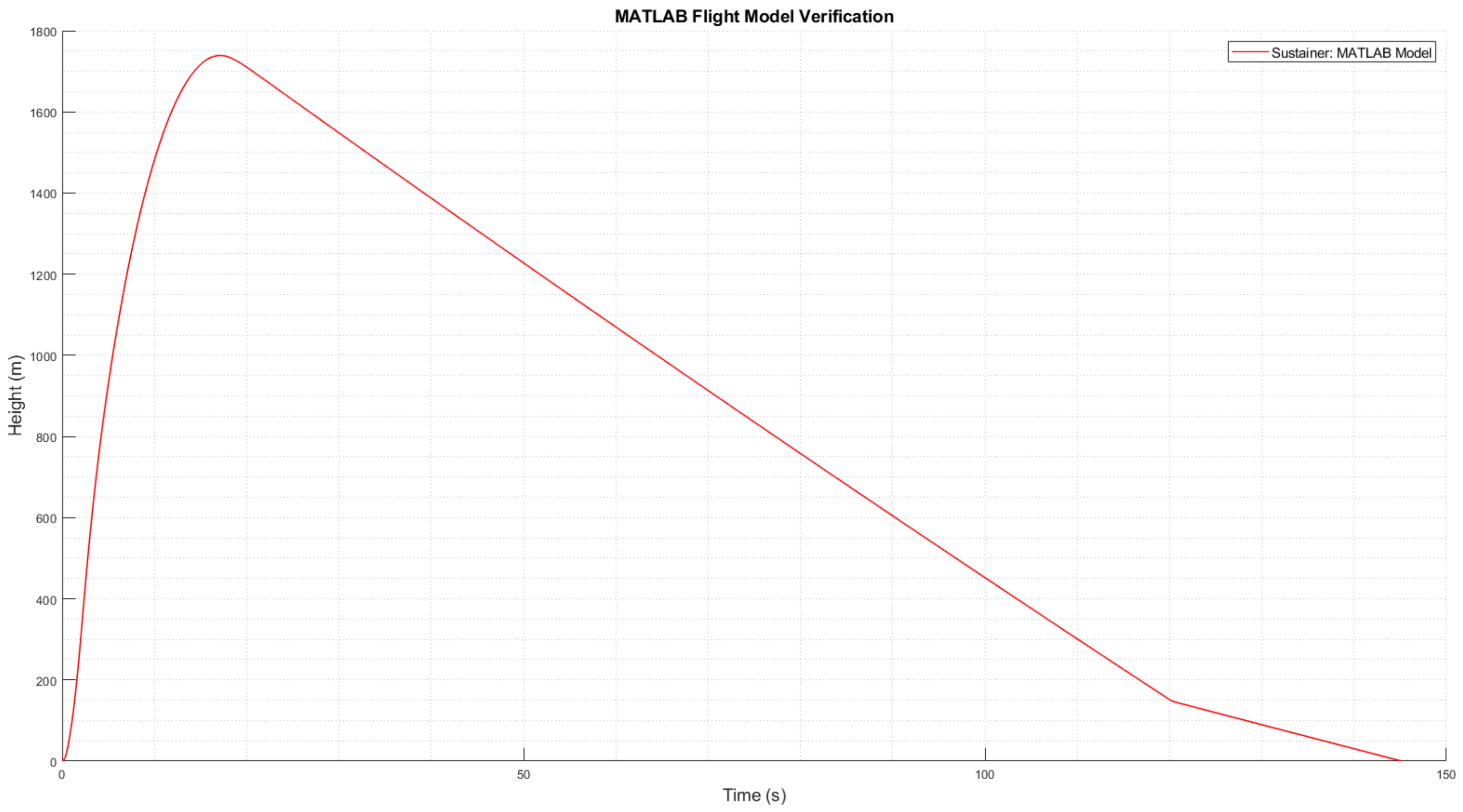


Figure 14. Flight Profile of Rocket Sustainer

Build Process Validation

Aether Class Iterations Leading to Final Competition Rocket

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 500 feet or 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.

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.

Aether VII – Competition Rocket

Launch Date: Either 9/22, 9/29, or 10/6

Aether VII is the culmination of UNH SEDS since its creation in the Fall of 2017. With a talented and multi-disciplinary group, we were able to see through our large goal of competing in this competition. With our many builds, countless lines of simulation and optimization code, and learning the ins and outs of high power rocketry concepts, we have been able to design, manufacture and assemble a rocket we all are extremely proud of. With our superior manufacturing and assembly expertise compared to just a few months ago, we have been able to manufacture and assemble our rocket correctly the first time with minimal mistakes. Without our previous editions of the Aether class, our final competition rocket would not be possible. They gave us an in depth knowledge on how every part of the rocket affects its overall outcome on the field (beginning with experimentation and imminent failure).

With the final dimensions selected for each part of the entire competition rocket, we could begin the building and integration of our rocket. The carbon fiber body tubes, engine tubes, and coupler tubes were purchased online due to our inexperience in rolling our own. This is a goal for the future composite rockets we make. The nosecone was attempted to be made from a 3D printed female mold and fiberglass (because fiberglass allows radio waves through, unlike carbon fiber). Being that it was everyone's first time dealing with the composites, we made a fiberglass nosecone that was strong and correct inner dimensions, but the outside dimensions were not acceptable for our expectations. Instead, we quickly were able to 3D print our nosecone to the exact dimensions and verify that plastic would be able to handle the amount of force we are expecting on our rocket due to drag. The centering rings and bulk plates used for our couplers were going to be made from a 1/8-inch carbon fiber sheet, but without the necessary equipment to manufacture it to size, we were forced to use a ½-inch acrylic sheet laser cut using out local makerspace. The difference in material and overall thickness proved to be a suitable replacement. With our limitation in cutting carbon fiber, we had to 3D print our fins and then reinforce them with fiberglass. The electronics bay sled was also 3D printed to perfectly fit into our nosecone and sustainer coupler to perfectly mount it. The containment of our engines in the positive and negative y direction is accomplished using two different aft retainers that were purchased online via a rocketry supplier company, Apogee Rockets. Our recovery systems are primarily ordered online to ensure that the standard quality of each part (parachutes, shock cords, etc.) are consistent.

Booster Build Process

The final integration of all our designed parts was nearly identical to our previous Aether builds during the 2017-2018 school year. Although the material of most our parts have changed, like the carbon fiber tubes and the acrylic centering rings and bulk plates, nearly identical tactics were implemented in order to assembly the rocket together. The booster stage was the first section of the rocket that was assembled due to the ease of complexity compared to the sustainer.

The main sections that need to be assembled in order are:

1. The engine tube and centering ring with shock cord slots assembly
2. The addition of the shock cord to the engine tube assembly for recovery attachment
3. Inserting the engine tube assembly to the body tube
4. The connection of the fins to the fin slots of the body tube
5. Inserting another centering ring for additional integrity
6. The addition of an aft retainer to the end of the engine tube
7. Adding the staging coupler
8. Fastening the recovery system to the shock cord

Five minute, two-part epoxy is used for the connection of the majority of the components on the competition rocket. Over every critical connection, generous fillets are created for greater strength. The centering rings are one of the most critical components in regards to strength. One centering ring was attached to the top of the booster engine tube and one was placed so the end of the ring would fit directly to the top of the fin tab, once inserted. The shock cord is then wrapped around the top centering ring and pulled through the entire body tube to be later attached to the bulk place on the staging coupler connecting the sustainer and booster together for the first few seconds of flight.

Once the shock cord position is set, and the centering rings have gone through a full cure (1 hour), we place epoxy inside the body tube to meet the top centering ring when the engine tube assembly is pushed up. Having the top centering ring encased by the body tube results in a natural epoxy fillet between the two parts if done correctly. We do the same for the bottom centering rings when the engine assembly is halfway to its final location. Once the engine tube assembly is pushed slowly to its final location, more epoxy is added to the aft end of the bottom centering rings for extra strength connecting the engine tube assembly to the booster body tube.

The booster fin tabs are then covered with epoxy and pushed through the fin slots in the booster body tube. Additional fillets are made inside as the tab presses against the engine tube. When the epoxy is beginning to set, our laser cut fin alignment tool is brought in to ensure correct angular fin spacing with respect to the body tube surface, achieving tight perpendicularity between the parts. A third centering ring is then brought in to the bottom of the rocket to seal the bottom completely and provide further support to the aft end of the fin tabs. Epoxy fillets are created for the third centering ring on the aft end. The aft retainer is then epoxied onto the end of the engine tube for booster engine retainment at launch. The final image up to this point can be seen below.

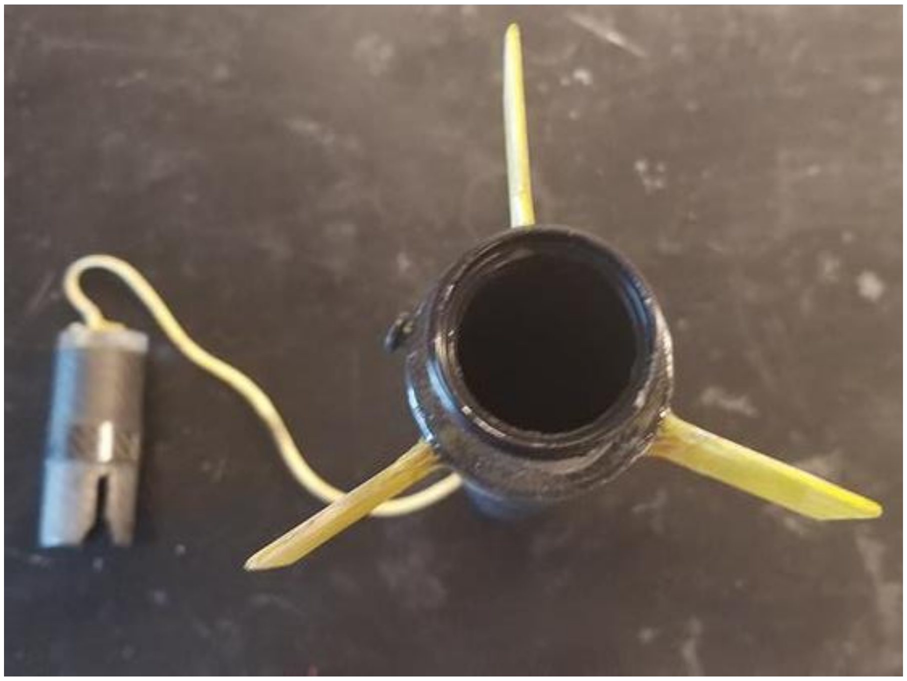


Figure 15. Booster Body Tube: Bottom View

The forward end of the Kevlar will then be connected to our staging coupler which will complete the booster rocket section. The staging coupler is assembled from our carbon fiber coupler body tube, a carbon fiber body tube piece (2cm long), and two different acrylic bulk plates. The carbon fiber body tube is slipped onto the middle of the coupler body tube and epoxied one with fillets on either side. We then grab the small and large bulk plates and epoxy them together so that the small bulk plate fits into the coupler body tube, and the larger one mates with the edge of the coupler body tube.

The bulk plates are then epoxied onto the aft side of the staging coupler with the forward end of the shock cord wrapped around two holes within the bulk plates. The parachute swivel hook and parachute lines can then be attached to the middle of the shock cord sticking out of the booster body tube. The parachute is then wrapped with the parachute lines and excess shock cord and positioned inside the forward end of the booster body tube. The staging coupler is placed inside the forward end of the booster body tube with a The parachute will then open after a set time from the built-in black powder ejection charge from the booster engine which will be set to a time that is predicted for booster apogee. The final booster assembly can be seen here:

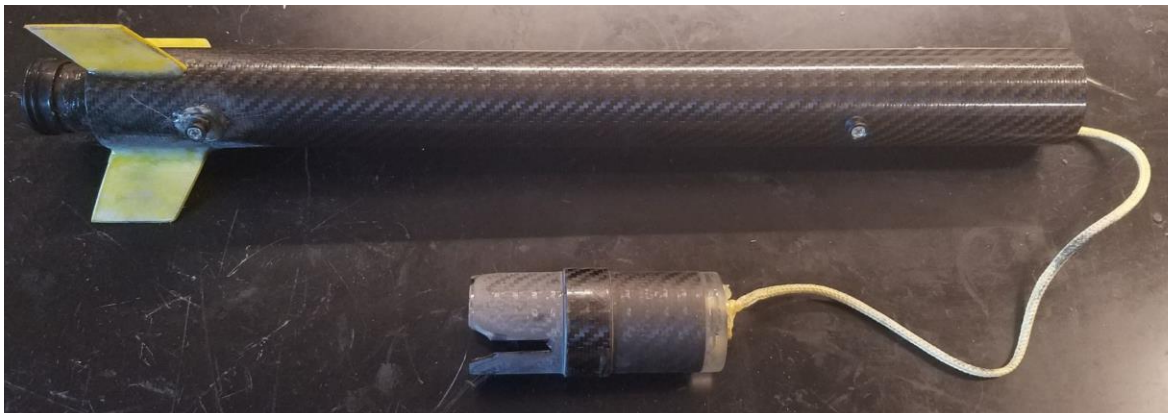


Figure 16. Booster Body Tube and Coupler

Sustainer Build Process

The sustainer has similar build process', but there are additional features to consider. To avoid repetition, the sustainer build process explained below will only elaborate on the differences from the booster build.

The main sections that need to be assembled for the sustainer are as follows:

1. The engine tube and centering ring assembly with shock cord and sustainer ignition wire slots
2. The addition of the shock cord to the engine tube assembly for recovery attachment
3. The insertion of the engine tube assembly to the body tube
4. The connection of the fins to the fin slots of the body tube
5. The addition of an aft retainer to the end of the engine tube
6. The creation of the forward coupler
7. The assembly of the e-bay inside the nose cone
8. The addition of the recovery system to the shock cord and an aft ejection charge on the forward coupler bulk plate

The sustainer engine tube and centering ring assembly is created nearly identically, but the centering rings have an extra slot for the sustainer ignition wire to travel through. The shock cord is then wrapped around the engine tube using both centering rings. Once the shock cord is attached to the engine tube and centering ring assembly, the attachment to the body tube works the same as the booster. The fins are attached with heavy fillets on every mated surface. The aft retainer can then be attached to the end of the sustainer engine tube. The staging coupler from the booster assembly must then be modified to be able to insert into the bottom of the sustainer body tube by cutting slits that avoid the fin tabs, and light sanding all around. The forward coupler can then be made by an identical process from the booster assembly. The electronics sled is then outfitted with the Telemega, battery, switch and the SEDS altimeter seen below.

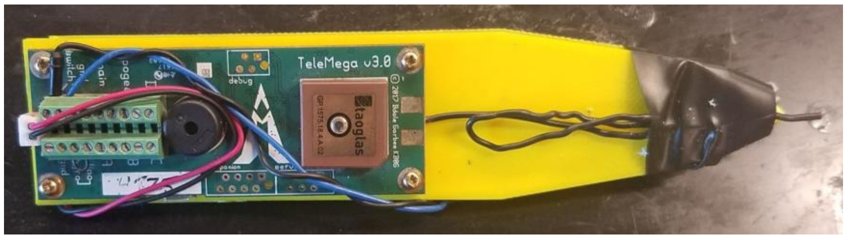


Figure 17. Electronic Bay Sled: TeleMega Side

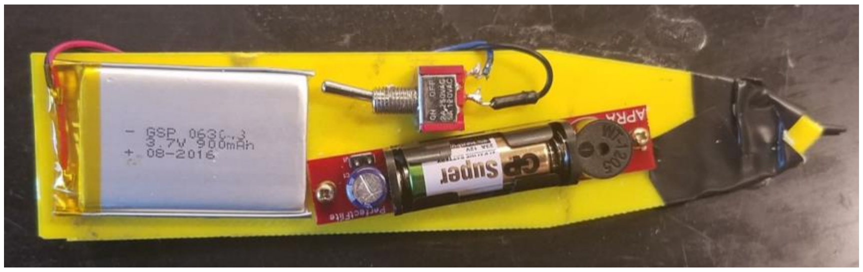


Figure 18. Electronic Bay Sled: SEDS Altimeter Side

With Styrofoam being epoxied into the tip of the nose cone creating a tight slit in the middle, the front of the electronics sled can be wedged in between making the it static in the nose cone. The nose cone can be seen below in side view and bottom view detailing the shape and Styrofoam, respectively.

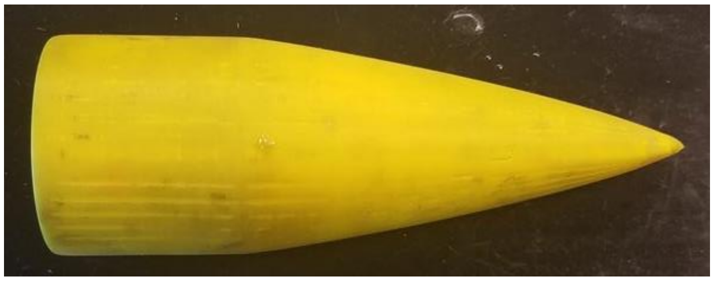


Figure 19. 3D Printed Nose Cone Side View

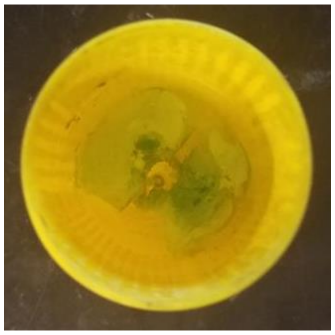


Figure 20. 3D Printed Nose Cone Top-Down View

The forward coupler can then be sanded to a tolerance fit to close the electronic bay in the nose cone. Vent holes are then added to the forward coupler to ensure correct pressure readings through flight. This new electronics bay offers a huge upgrade to instantaneous accessibility and testing. The front of the shock cord attached to the sustainer engine tube and centering ring assembly can then be attached to the aft bulk plates of the forward coupler. The small and large parachutes can then attach so that the small parachute would open first. The aft ejection charge is then equipped onto the forward coupler bulk plate to ensure safe recovery of the sustainer. The final sustainer body tube and coupler can be seen below:

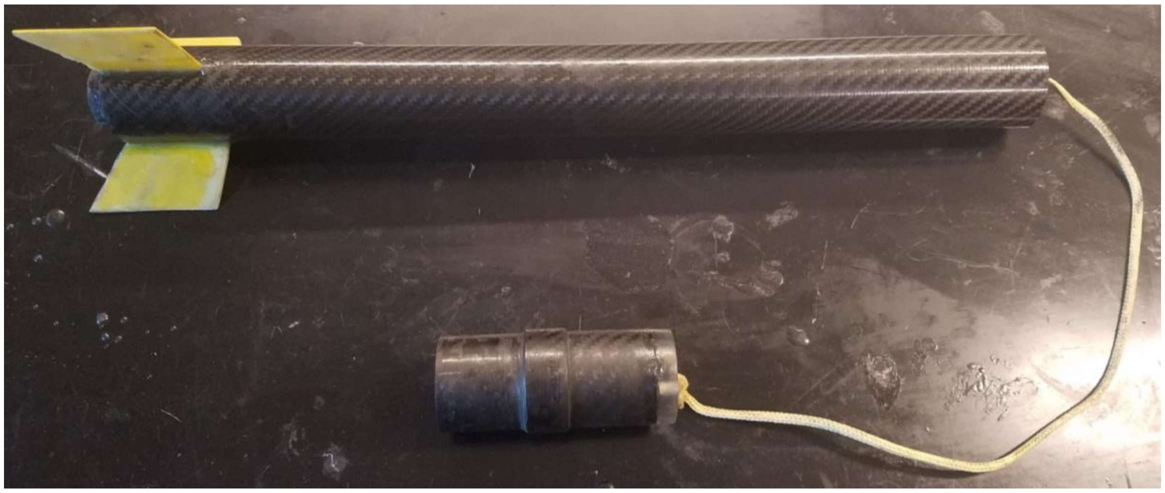


Figure 21. Sustainer Body Tube and Coupler

Electronics Documentation

Onboard Recovery Electronics

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. Our electronics bay is in the nosecone of our rocket on a plate held in with Styrofoam. On one side of the plate is the Telemega, and on the other is the, the Telemega battery, the switch to activate the Telemega and the SEDS-required beeper altimeter. The Telemega system is shown in Figure 22.

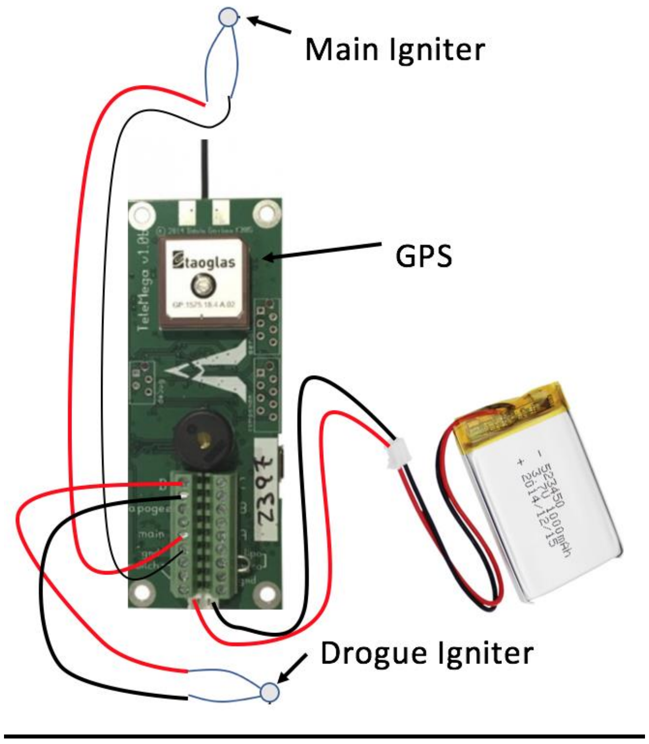


Figure 22. Wiring Schematics of the Telemega GPS

Electronics Test Plan and Preliminary Results

On previous rockets, we tested the RRC3 altimeter by hooking a vacuum up to the body tube to decrease the pressure read by the altimeter. When it reached a max pressure and began to stop (what happens just after apogee), the altimeter would wait one second and then ignite a black powder capsule to deploy the parachutes. With higher quality electronics, the Telemega is able to bypass the more complex and dirty pressure test to simply hold the rocket vertically and thrust it upward quickly to fool the altimeter into thinking it was launched. After a preprogrammed amount of time (2 seconds for our competition rocket after launch), the Telemega would send a signal to our sustainer igniter to light the engine. After apogee and most of its descent, when the Telemega senses that the rocket has passed an altitude of 1,000 feet with a negative velocity, it would again ignite another charge to pressurize our sustainer body tube to deploy the two parachutes.

Before launch, we synchronize the GPS (part of the Telemega) with nearby satellites. The GPS not only tracts where the rocket lands but it also records the rockets flight data so we can consider the actual flight path of the rocket and compare it to the simulations we programmed last year to optimize the final rocket.

From there we can address the differences between the simulation and the actual data to further our simulation accuracy after the competition.

So far, we have only been able to test the Telemega GPS connection, and the successful ignition of igniters we made. During the week of September 17th, we plan to test the opening of our small and main parachute in the sustainer body tube using very fine gun powder and an electrical spark made from the Telemega. Last year, we were able to master the software of the Telemega (with many mistakes) to ensure we understand it enough to ensure a safe and successful flight with full recovery.

Issues Encountered During the Manufacturing Process

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 began the final manufacturing and assembly of our competition rocket.

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 perfect fit for each part. The centering rings, bulk plates and nose cone will be manufactured in house. The centering rings and bulk plates could be cut from 1/8-inch carbon fiber sheets we have available, but with limitation in how we can manufacture it to size, we had to go with ½ inch thick acrylic laser cut using our local makerspace. With Solidworks, we were able to ensure that the acrylic is comparable to the carbon fiber at this thickness and configuration. It was also calculated that it only increased weight by an insignificant amount (<10%). During the first week of the Fall semester, we immediately began the printing of a female mold to fiber glass the nose cone. This was a new skill to all of us, so there was a large learning curve. We found it hard to keep the layers of fiberglass even throughout and the final result was not axially symmetrical. Having a non-symmetrical nose cone could have cause extreme issues in stable flight and we didn't want to risk the entire rocket on a skill we only started to develop. Ultimately, we used a 3D printed PLA nose that was sanded down for smoothness. Although an ABS nosecone would prove to be structurally stronger and less prone to deformation from outside pressures, we did not have the printer capable of printing in ABS available to us. We decided to also 3D print our fins exactly to size, and then added two layers of fiberglass for strength and the avoidance of fin flutter. As we continue building newer and better rockets, in house composite manufacturing will improve naturally.

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 requires stronger centering rings, and an overall rocket build capable of travelling at faster speeds. This lead us to go with carbon fiber, a stronger composite then the rocketry standard, blue tube, and thicker centering rings with more precise and experienced epoxy fillets. When reinforcing the sustainer centering rings with thicker layers of epoxy, we mistakenly forgot to insert the wire leads for sustainer ignition through the slots we made to easily solder the igniter section for the engine and the wire leads to the Telemega. This has required us to order a 1/8-inch extended drill bit (12 inches long) so we can reach into our body tube and make the necessary cuts to reopen our slots from the epoxy.

Parachute Choice

The parachute research and testing we have completed has given us a base of knowledge that was easily transferable to our competition rocket. A major change from our original design report, the negation of our drogue parachute, was chosen mainly to ensure the rocket gets down to sight level as soon as it can. We will be incorporating a small, drogue-like parachute with our main parachute, folded in a way to ensure the drogue opens first. With the extra strength of 1000 lb. Kevlar shock cord, professionally made parachutes and reinforced centering rings, we have calculated that the force from two separate parachutes opening at different times does not have the required strength to break any critical components to lead to failure.

Launch Failure Mitigation

Improvement Cycle for Failure Mitigation

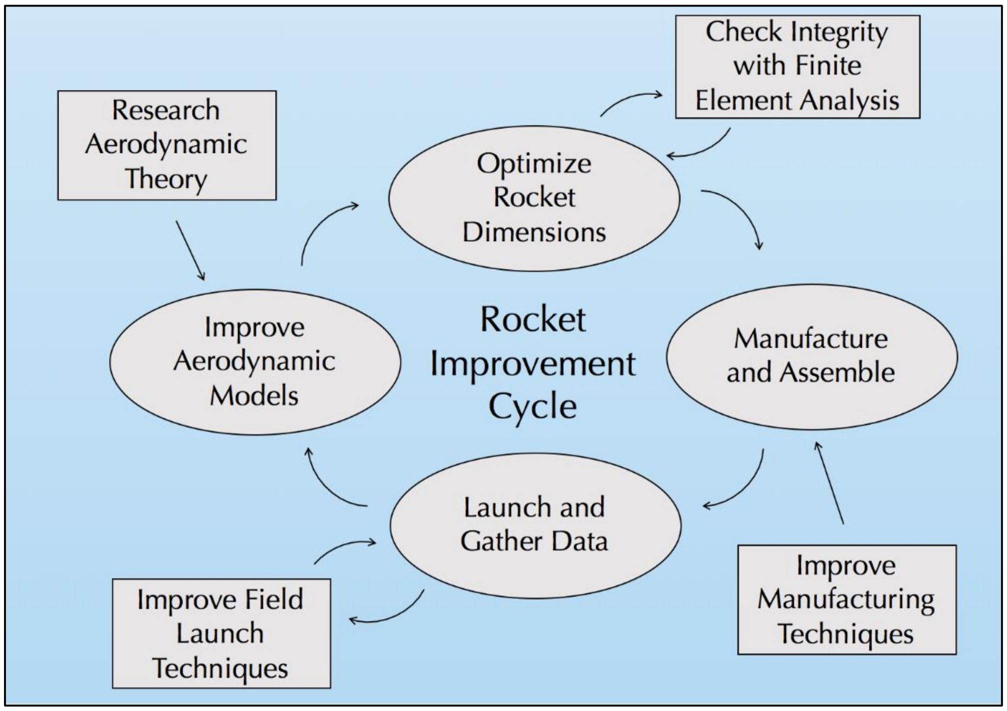

Check Integrity with Finite Element Analysis
Research Aerodynamic Theory
Optimize Rocket Dimensions
Improve Aerodynamic Models
Rocket Improvement Cycle
Manufacture and Assemble
Launch and Gather Data
Improve Field Launch Techniques
Improve Manufacturing Techniques

Figure 23. Improvement Cycle

Our team utilized an improvement cycle to ensure progression and improvement on each rocket iteration to ensure each flight becomes more successful and safer. 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 then checked with finite element analysis to ensure structural integrity with a minimum factor of safety of 3. This safety of factor was chosen purely to mitigate the error that we predict of a new rocket team. In the future, we would like to lower this factor of safety closer to 1.5. The rocket is then manufactured with the optimal dimensions, while working 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.

The SEDS competition rocket is our most precise and strongest rocket ever made. It is the 9th rocket we have made, and is the last of its Aether class, Aether VII. This improvement cycle, which has been modified and improved over the 2017-2018 school year, has helped us continue to improve with each build, preventing senseless mistakes and encourages us to improve.

Failure Mitigation Safety Checks

Our efforts to mitigate the risk of failure of any part of the rocket started with an excessive roll during flight due to the method we used to attach and align the rockets fins. In order to improve upon our design and further mitigate possible sources of error, the team created an alignment tool for the fins out of acrylic. The cross-sectional shape of the body tube and fins was laser cut into a piece of acrylic with a minimal offset to create an evenly spaced holster for manufacturing. During the manufacturing process the alignment tool was used to space out the cuts for the fins to ensure 120-degree separation. To further ensure that the fins were then structurally secure to the rocket, epoxy was laid in the inlet for the fin as well as between the fin and the outside of the body tube.

In an effort to save money on our builds, the team decided to try to create our own fuses to use to reignite the sustainer engine mid-flight through the use of the accelerometer system. Unfortunately, where we saved money, we lost quality and witnessed multiple failures due to faulty wiring and fuses. In an effort to reduce this source of error, we decided to use outsourced fuses in future builds, to ensure quality and repeatability of the launch. This has proven beneficial as ignition of the sustainer engine has not continued to be a problem. We also had to ensure that the igniters that were purchased were long enough to allow for full parachute deployment as the stage finishes fight. One flight had a failure in main parachute deployment due to the igniter being shorter than the shock cord attached to the parachute. This was fixed by extending our igniters manually.

We ran into the same issue with our parachutes, as our in house made versions were just not as high quality as something that could be purchased, resulting in a few failed parachute deployments. In order to mitigate risks, parachutes were outsourced and integrated into the rocket design as well. Premade parachutes not only aided in making the rocket less prone to failure, but also saved time in the manufacturing process, giving added benefit.

One thing that we were actually able to incorporate in the design as result of intuition and not failure was our upgraded centering rings. Earlier iterations of the rocket used wooden centering rings to align internal components and boosters, but we realized that as we increased the power of our booster and sustainer, we would need to dilute the shear forces that would be produced against the centering rings. Preemptively, we decided to try to use carbon fiber to create the rings. Unfortunately, cutting the carbon fiber proved difficult, so our next attempt saw the centering rings being made from laser cut acrylic. The results were stronger centering rings with greater repeatability of build due to laser-precision accuracy. The same thought process went into our creation of our 3D printed nose cone made with PLA plastic. Both parts were modeled in Solidworks and processed through a finite element analysis in which the goal was to create parts with at least a factor of safety of 3 with the expected applied forces during operation.

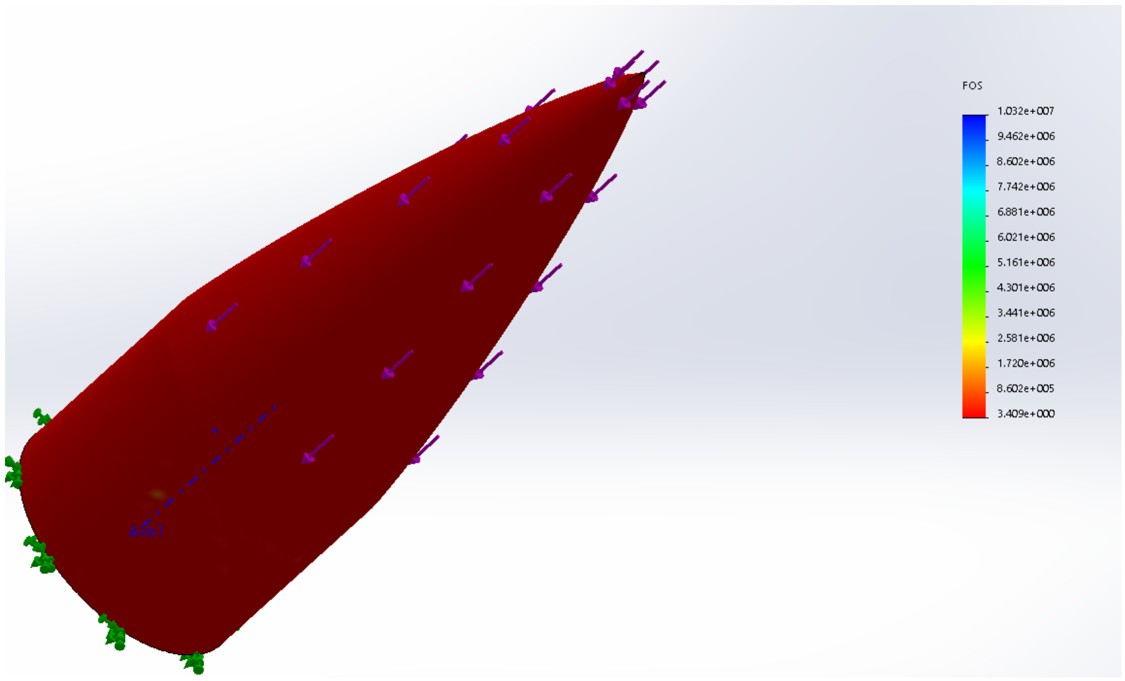


Figure 24. FEA of the Nosecone during Flight

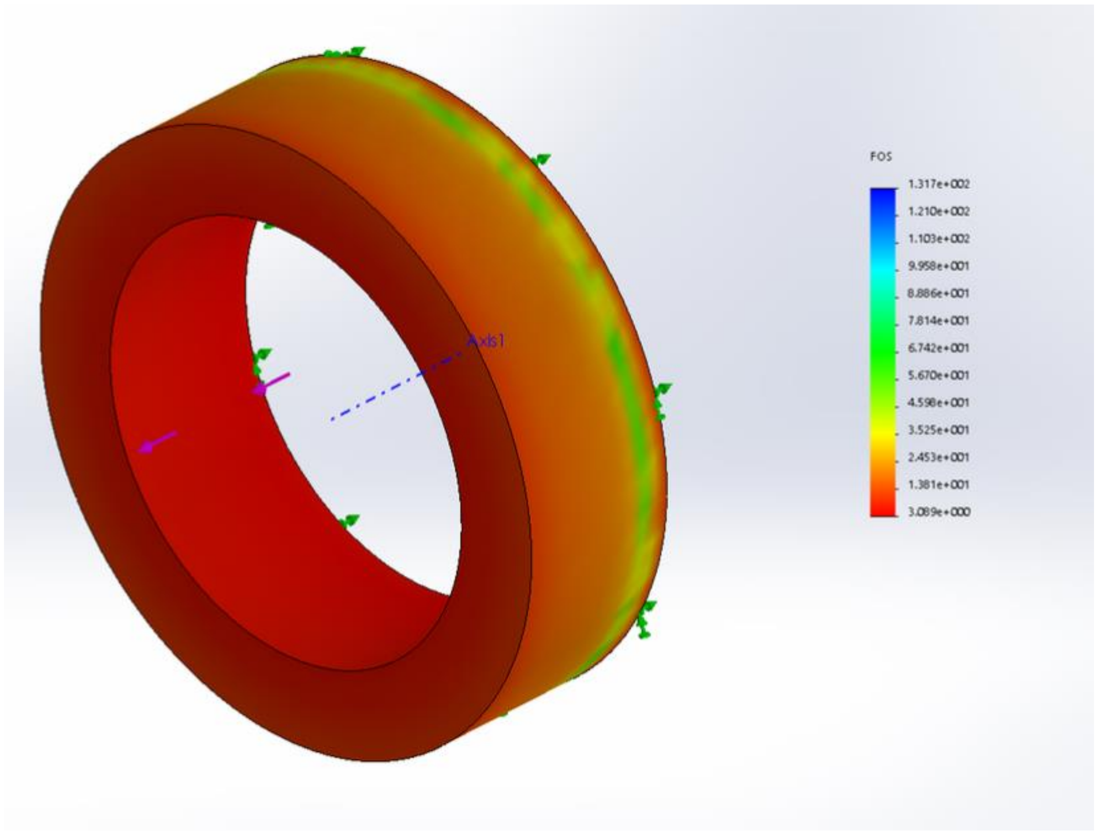


Figure 25. FEA of the Centering Rings during Booster Thrust

Updated Launch Dates

With the unpredictability of scheduled rocket launches, we have had to adapt to what we are given for time and place of launches. Our nominal launch field 20 minutes away with a flight ceiling recently changed their future launch days to be in Amesbury, Massachusetts, which only has a flight ceiling of 4,000 feet. From that, we have contacted all fields in a 5-hour radius and got into contact with a launch field in Cherry field, Maine, that is very willing to host us during their launch days every Saturday leading up to when the launch window closes. This field is 4 hours away, which would make our final launch even more invigorating. We were also able to get into contact with our fellow South Berwick rocketeers that told is that they might be able to change one of their launch days in Amesbury back to South Berwick to accommodate to us. Although it was stressful to constantly be on our toes on where we could launch before the launch window, we were able to, yet again, realize that the community of rocketeers in the Northeast are always willing to help each other out.

Available Launch Dates until October 12th:

- September 22nd in South Berwick, Maine or Cherryfield, Maine.
- September 29th in South Berwick, Maine or Cherryfield, Maine.
  - This launch date is if the September 22nd launch falls through.
- October 6th in South Berwick, Maine or Cherryfield, Maine.
  - This launch date is if the September 29th launch falls through.

Current Completion Status

All raw materials of the UNH SEDS competition rocket arrived throughout the summer. During the beginning of the Fall semester, the team thoroughly reviewed the process that we developed last year to build a high power rocket. Once we were all caught up in the entire process of manufacturing and assembly, we began manufacturing all the parts needed for the final assembly of the rocket. On September 10th, 2018, we began booster assembly and finished it the evening of September 11th. The current state of our rocket booster can be seen below:

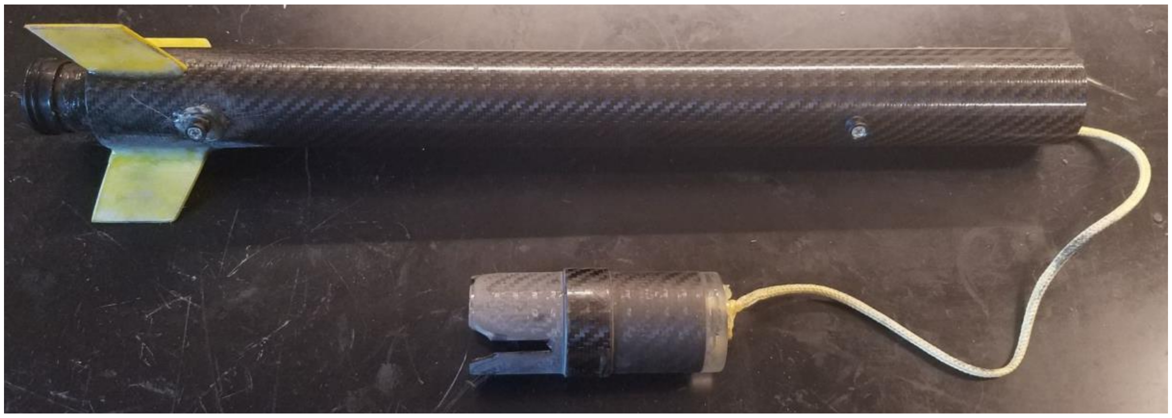


Figure 26. Booster Body Tube and Coupler

After the booster was completed with the installation of recovery we began the assembly of the sustainer stage and mainly finished on September 13th. As for ongoing assembly, we still are in the process of getting the staging coupler to the right fit for sustainer ignition and coupler blow off. The metal aft retainer to restrain the sustainer engine in the –z direction (+z is towards nosecone) has to also be installed with high temperature epoxy. The last two features that need to be added to the competition rocket are the addition of electrical holes through the sustainer centering rings to electrically ignite the sustainer engine during flight, and the installation of our black powder case that will blow the main parachute out at our desired altitude of 1,000 feet. The current state of our sustainer assembly can be seen here:

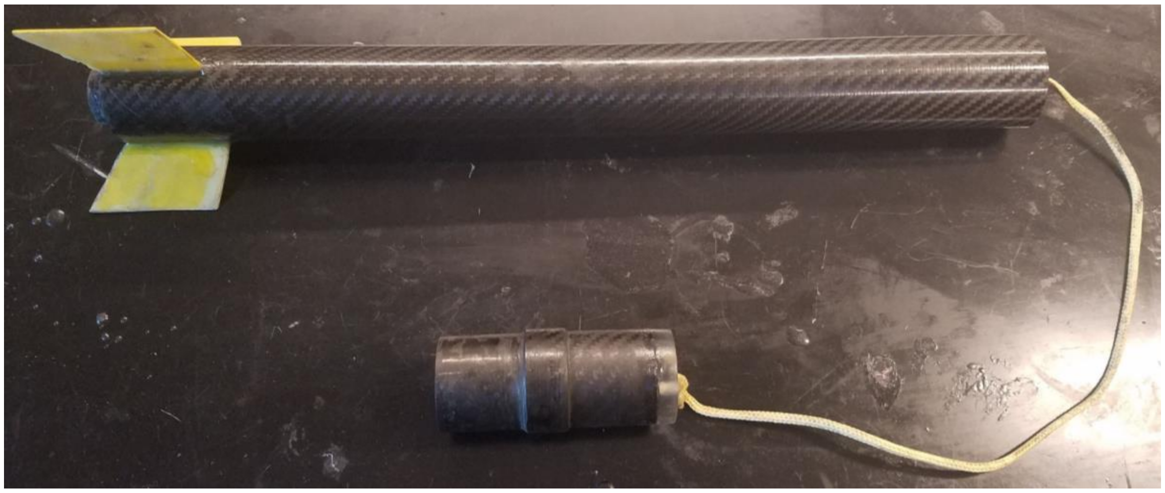


Figure 27. Booster Body Tube and Coupler

The nosecone and electronics bay is finished with the intent to continue testing next week leading up to our scheduled launch dates. Although these first few weeks have been exciting seeing all our work rise up to a physical accomplishment, more would could have been done during our off hours (summer) to mitigate the amount of work that we had to put into to get the rocket and report ready by our deadlines.

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

[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
tseperation = 2.3;   % booster seperation

%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 Seperation)
% Initial Values and Vectors for Booster Seperation

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

for t = tseperation+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
```

Thrust

```
function [thrust] = GetThrust ( t )

% ~ AETHER4 ~

% burntimeboost = 1.3;  % s
% burntimesust = 3;     % s
% startTimeboost = 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
```

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
```

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;
    crosssection_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 coefficent
    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*(crosssection_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
```

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 seperation

% ~ 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
Maftfins            = .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 + Maftfins + 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
xdrogueparachute      = 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 coefficent, 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
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)
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
coefficent, 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
```