UNH SEDS
Final Launch Report

October 2018

Authors

Charlie Nitschelm, Reilly Webb, Thomas Collins,

Grace Johnston, Silas Johnson, Charles Gould, Ross Thyne,

Matt Dodge, Ester Yee, Carly Benik, Samantha Brady,

Andrew Hosman, Chase Eldredge, Collin Stroshine,

John Langer, Ryan Blatti, Tyler Landry, Lucas Simmonds

Advisor

Dr. Todd Gross

Table of Contents

List of Figures

List of Tables

Introduction

Abstract

Members of UNH SEDS are designing, manufacturing, and launching a high powered multi-stage rocket for the SEDS University Student Rocketry Competition (USRC). 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 the final launch, the failures 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. 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. A chapter of SEDS was founded at UNH in the Fall of 2017 with the USRC as the primary goal of the organization for its first year. 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, which we will be attending! 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 the UNH 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. There were modification to its configuration on the field to ensure complete safety of the spectators, including the addition of a chute release in the main parachute bay within the sustainer tube.

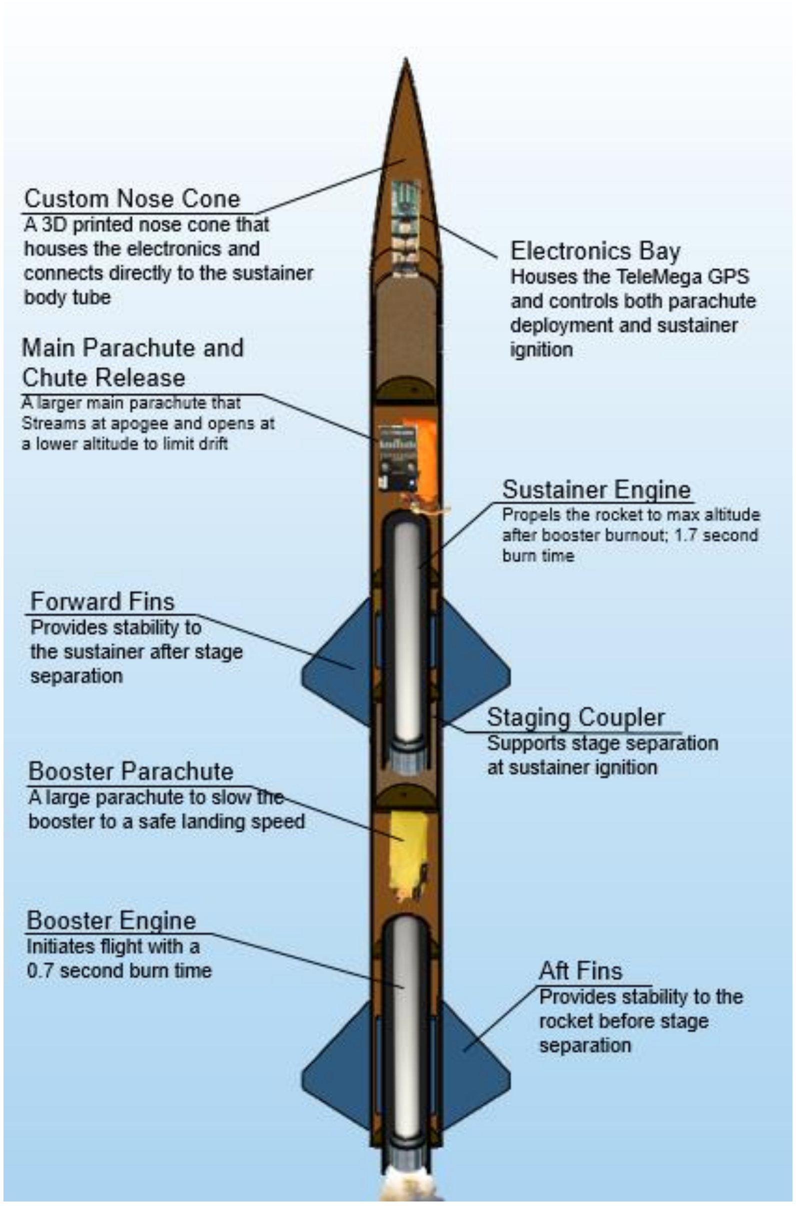

Custom Nose Cone
A 3D printed nose cone that houses the electronics and connects directly to the sustainer body tube
Electronics Bay
Houses the TeleMega GPS and controls both parachute deployment and sustainer ignition
Main Parachute and Chute Release
A larger main parachute that Streams at apogee and opens at a lower altitude to limit drift
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. In the future, UNH SEDS will attempt to design, manufacture and test their own hybrid rocket engines to compete in the Spaceport America Cup.

The TeleMega GPS/Altimeter system used will be referred to as the flight computer. The flight computer, which was moved to the nosecone, sends electrical signals at various preprogrammed events. Current is sent to two individual ignitors; one that initiates the firing of the sustainer engine and one that lights ejection charges for deploying the main parachute wrapped in the chute release.

The booster engine is the first engine to fire, and is 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 which was simulated to be 2310 meters. The flight computer will then send a signal to an ejection charge located in the aft end of the nosecone that opens the parachute bay releasing the main parachute wrapped in the Chute Release, acting as a streamer. The sustainer then free falls until it reaches a certain height to prevent large amount of sustainer drift (300m). At this predetermined altitude, chute release send a signal and opens the main parachute that slows the sustainer to a safe descent speed. This prevents significant damage from ground impact, allowing for full reusability.

Competition Flight Overview

When the team arrived at the launch field, sky conditions were nominal. Winds at the ground were at a low for the day (4 m/s), but winds near our simulated apogee were nearly 10m/s. The NAR South Berwick field usually has large crosswinds, which is the primary reason we designed our competition rocket to utilize a single parachute deployment at 300 meters, allowing the rocket to free fall from apogee. We used a launch angle of 10 degrees to help compensate for the wind, as recovery is very important for the competition.

Once the rocket was prepped and confirmed for launch, the launch safety officer was concerned with allowing our sustainer free fall from such a high altitude (2,310 meters simulated) due to the risk of parachute deployment not working. A chute release was donated to us at the field so we can separate at apogee with the main parachute acting as a streamer, and opening up at 300 meters via the chute release. We were hesitant to incorporate the chute release in the final moments because of our inexperience the mechanism, but it was the only way to safely launch on that day. We respected the concerns of the safety officer and agreed to use the chute release for the flight.

We went through our safety and launch procedure checklist to prepare our rocket for its competition launch. The countdown began, boiling the entire organizations work over the last year into just 10 seconds. The booster ignited with double the thrust our team has ever flown, launching the rocket off the launch pad. A less optimal booster engine needed to be used (H159) instead of the desired H399 because it wasn’t commercially available. With this change, take off velocity was lower than expected, resulting in instability at the beginning of our launch. This instability altered our flight trajectory, changing the tilt angle of the rocket significantly before sustainer ignition. Then, 2.9 seconds after booster cutoff, the sustainer ignited and carried our sustainer to its apogee of 2,215 meters. The delay time between booster engine cutoff and sustainer engine ignition was optimized to be 2.0 seconds for maximum height. The sustainer engine took an extra 0.9 seconds to start, which led to less than optimal sustainer ignition.

Recovery has never been the teams' strong suit, but this launch was different. The booster was recovered with complete reusability as seen in the footage linked near the end of this report. The sustainer, which its recovery methods were altered last minute, had a different fate. The apogee event was nominal, releasing the main parachute out of the body tube still wrapped in the donated chute release. This resulted in a descent speed much slower then what would have been seen if the rocket were to free fall, as initially designed. This decrease in descent speed and increase in flight time resulted in a large drift away from the launch pad that could not be predicted. We traveled to the coordinates of where the TeleMega last recorded the location of the rocket only to find that it had landed in the middle of a small lake. The electronics were lost and certainly waterlogged but the TeleMega still transmitted all of our flight data to our launch computer on the field. The SEDS altimeter went down with the sustainer, not allowing us to verify the beeping apogee of the rocket. Our field flight computer received all the data of the flight that is needed to analyze the launch and more.

Overall, our launch was successful. It reached an apogee that had surpassed previous competition winners and our flight data was recovered. Although the sustainer was not recovered from the lake (although we have some volunteer scuba divers attempting to find it next week), we are confident that if landed on land, the reported descent speed would be nominal to have been fully recoverable. Unfortunately, the sustainer was not designed to be water proof. We made the conscious choice to change our flight mechanics on the field in worry of parachute deployment failure, but we are confident that if the original flight mechanics were used for launch, a much better outcome would have been seen for the sustainer recovery.

Safety and Launch Procedure Checklist

Launch Operations and Safety Precaution Checklist

1. Assemble the launch pad in a desired location given the rocket flight path, wind direction and field orientation
   a. The wind was strong, so an angle of attack of 10 degrees from the zenith was implemented to compensate for environment conditions.
   b. Set up the launch rail stopper to raise the rocket roughly 6 inches above the launch pad base
   c. Slide rocket into position on the launch rail
2. Turn on the Telemega GPS to begin satellite connection and flight configuration
   a. Ensure battery is fully charged and correct beeps sound when switch is flipped on
3. Turn on SEDS Altimeter by connecting the terminals on the side
   a. Ensure correct beeps upon startup
4. Insert the booster engine assembly into the booster
   a. 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
    a. Ensured that the range safety officer and the launch director are the only ones at the launch controller to limit any risk of preemptive ignition
    b. All other members and observers must be 100 feet away from the rocket
7. Install igniter leads into engine by inserting through the nozzle and up the grain until it reaches the top
8. Screw engine casing retainer onto the booster engine tube to constrain the engine in both the positive and negative y directions
9. Perform final checks on the rocket
    a. Fin alignment between stages
    b. Proper retainment on all engines
10. Run from launch pad (It is UNH SEDS tradition to always run from the launch pad before countdown)
11. Verify TeleMega connection, continuity on all igniters and GPS tracking
    a. Connection and continuity of both igniter leads on the TeleMega were nominal.
    b. GPS tracking had 4 satellites in solution in under 3 minutes.
12. Setup launch controller to the battery for launch
13. Verify surroundings and safety of all people
    a. Launch safety officer gave the okay for launch
14. Check continuity
15. Countdown
16. Launch

Results Analysis

The following analysis of our launch results compares our real, experimental data taken at launch with our homemade, in-depth MATLAB simulation of a multi-stage rocket. OpenRocket, which is a widely accepted rocket launch simulation software, is also compared to lay a framework of how our simulations compare with what is commonly accepted accurate in the high-power rocketry community.

Height vs. Time Analysis

Figure 2 below details the actual flight profile of height vs. time compared to the homemade MATLAB and OpenRocket simulations. The raw launch data and both MATLAB and OpenRocket simulation data had identical time to apogee of 23 seconds. The recorded apogee of the raw TeleMega flight data provided a maximum height of 2215 meters or 7,267 feet. The MATLAB and OpenRocket simulation results recorded slightly higher maximum heights of 2,310 meters (7,578 feet) and 2,542 meters (8,340 feet). These data resulted in a maximum height difference of 95 meters (311 feet) from the raw TeleMega flight data to our homemade MATLAB flight simulation and 327 meters (1,073 feet) from the raw TeleMega flight data to OpenRocket's flight simulation. Our MATLAB flight simulations have consistently provided a closer prediction to the actual flight path compared to OpenRocket, which is an achievement we take pride in. The difference we experienced between the raw TeleMega flight data our homemade MATLAB simulation data could be the result of the initial instability of the booster during the first few seconds of launch. Our simulation assumes that the rocket remains stable over the entire ascent period.

Our TeleMega was located in the nosecone, so once separation of our rocket occurred at 4.9 seconds, the booster was no longer gather flight data, leaving no actual data of our booster section leading to booster apogee, descent and landing. The flight profile of the raw TeleMega flight data to the MATLAB flight simulation data and OpenRocket's flight simulation data are fundamentally different. The actual flight of the rocket utilized the donated chute release for an apogee event allowing the main parachute to act as a streamer during the majority of descent. The simulations kept our original flight mechanics allowing the sustainer to free fall until the sustainer hits 300 meters, ejecting the main parachute out of the sustainer body tube, slowing the rocket down to a very safe descent speed. This difference in the flight mechanics of the sustainer describes the difference the design choice made on the total flight time. Descent time and sustainer drift are directly proportional, meaning smaller descent times yield less horizontal drift from the launch pad. From the raw Telemega flight data we are able to conclude that the parachute ejection system using homemade black powder charge worked, meaning that if we maintained our original flight mechanics, we would have been able to recover our precious, expensive sustainer. Although this flight could have been completely nominal if the rocket were to be flown as it was designed (free fall at apogee until it hits 300 meters), we still believe the launch safety officer made the correct choice to add a chute release to our recovery system for the safety of everyone at the field. The actual descent speed of the rocket was recorded to be 20 m/s, following our models of what the wrapped parachute acting as a streamer would provide. The chute release, which has never been used before by the team (partially because we could not afford it), seemed to have not opened the parachute at 300 meters, as it continued the same streamer descent speed until the notorious lake landing.

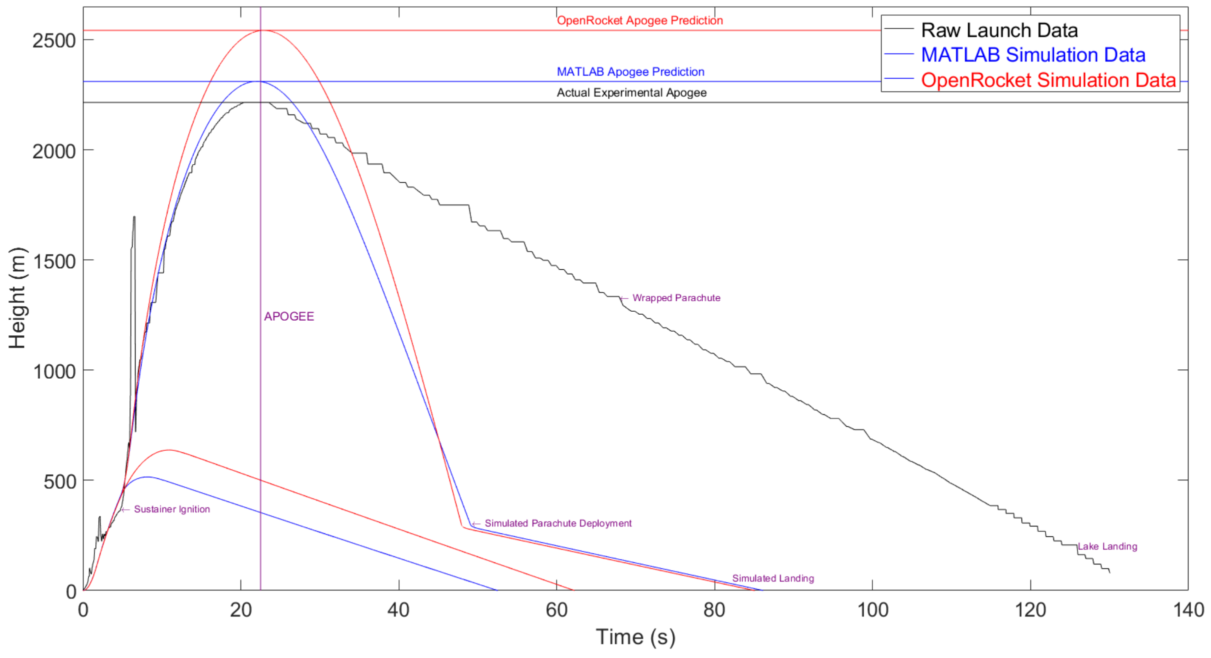


Figure 2 - Height versus time of our actual rocket flight recorded with our TeleMega plotted with our homemade MATLAB flight simulation program and OpenRocket's flight simulation of the predicted flight path

Figure 3 below enlarges figure 2 to include only the first 10 seconds of flight. The first 10 seconds clearly shows the beginning of flight, including launch, booster cutoff 2.0 seconds later and sustainer ignition 4.9 seconds into the flight. The TeleMega is housed in the nosecone with air vent holes through the coupler to allow the nosecone to have the same pressure as the outside environment. By using pressure to calculate the rockets height, it is common that at events that drastically change the acceleration of the rocket from above 0 (when thrust is applied) to below 0 m/s^2 (booster and sustainer cutoff), pressure drastically changes in the nosecone for a short amount of time until the pressure inside the nose cone returns to the environment pressure. You can see both of these occurrences in figure 2 from the booster and sustainer cutoff and the booster cutoff pressure effect in figure 3. This phenomenon is seen for all altimeters that use pressure for height recordings and does not affect the rest of the data outside of its event.

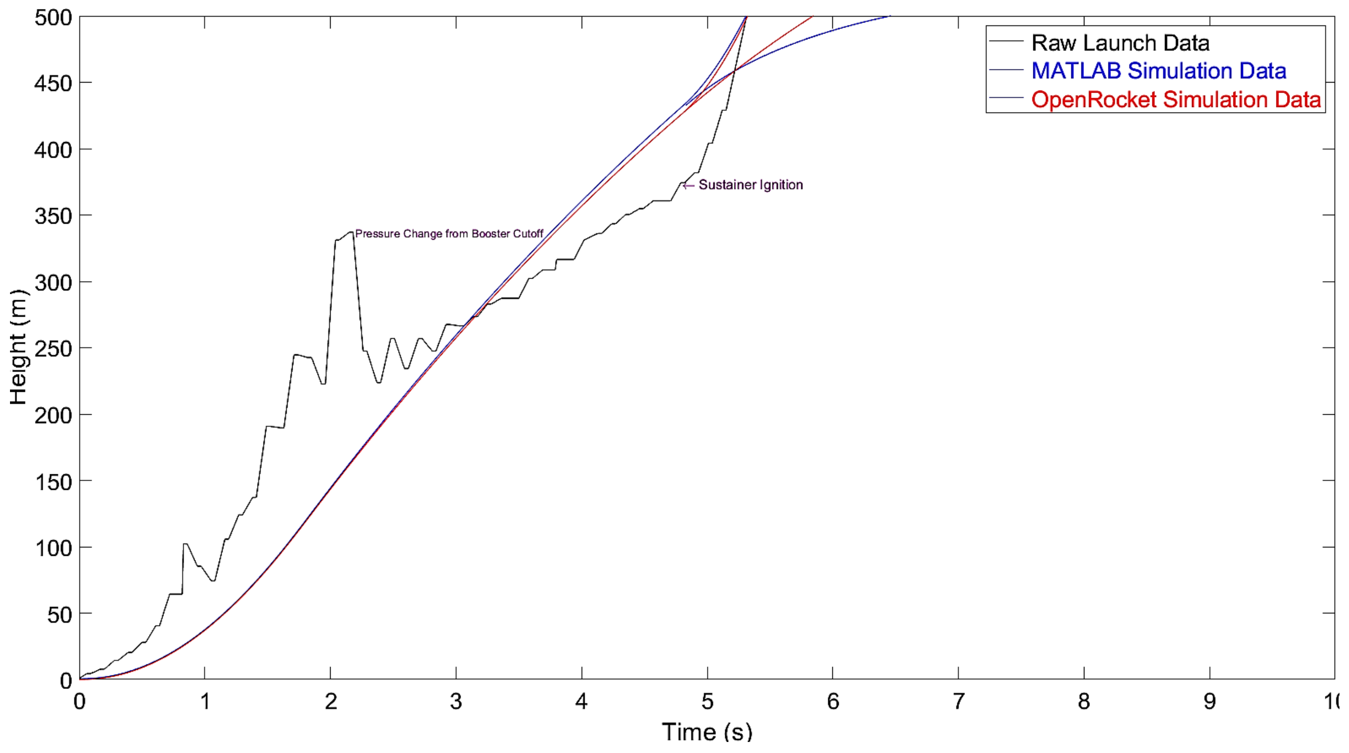


Figure 3 - A detailed view of the height versus time plot of our actual rocket flight with our TeleMega plotted with our homemade MATLAB flight simulation program and OpenRocket's flight simulation of the predicted flight path

Velocity vs. Time Analysis

We can then explore the relationships between the raw TeleMega flight data to our in-house MATLAB flight simulation data to dive deeper into the differences between the overall flight path results. Figure 4 below shows the relationship between the heights versus time data with the velocity versus time data. We quickly see that our simulation of the flight nearly precisely matches the raw TeleMega flight data of when apogee occurred. An interesting consequence of our simulation to the raw TeleMega flight data was that the actual flight experienced a higher max velocity (maxV). As our simulations predicted higher max heights then what actually happened, the simulation should result in higher velocities for most of the flight to result in a higher apogee that we have seen. This discrepancy was investigated and led the team to the velocity data of the raw TeleMega flight data and common data people who have used the TeleMega experience.

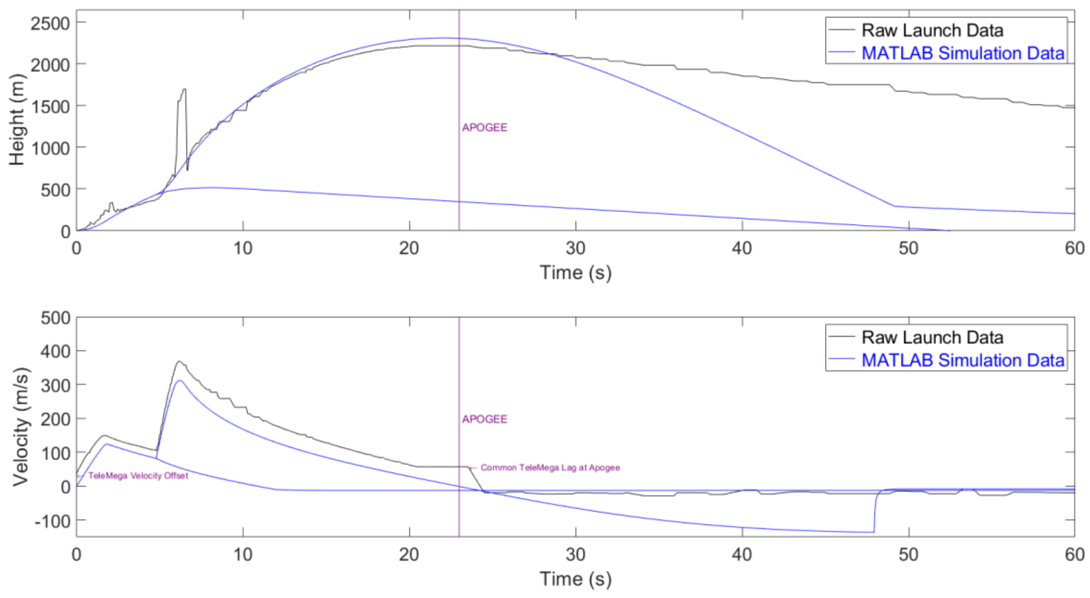


Figure 4 - Height and velocity versus time of our actual rocket flight recorded with our TeleMega plotted with our homemade MATLAB flight simulation program

Figure 5 extends figure 4 to include a 3rd graph that details an explanation of this error. When TeleMega boots up, it orients itself to where it is on the Earth, connecting to the satellites for GPS tracking. This can result in an initial velocity that is not 0 m/s, but a value above that. This TeleMega velocity offset does not affect its calculation of height versus time, but must be accounted for when plotting the actual velocity of the rocket through flight. The TeleMega recorded an initial velocity of 34.2 m/s, which was then accounted for and plotted on the 3rd subplot in figure 5. This resolved the velocity discrepancy as the MATLAB simulation velocity was slightly above the raw TeleMega launch data velocity, allowing the MATLAB simulation data to print a slightly higher maximum height from the actual apogee recording from the TeleMega.

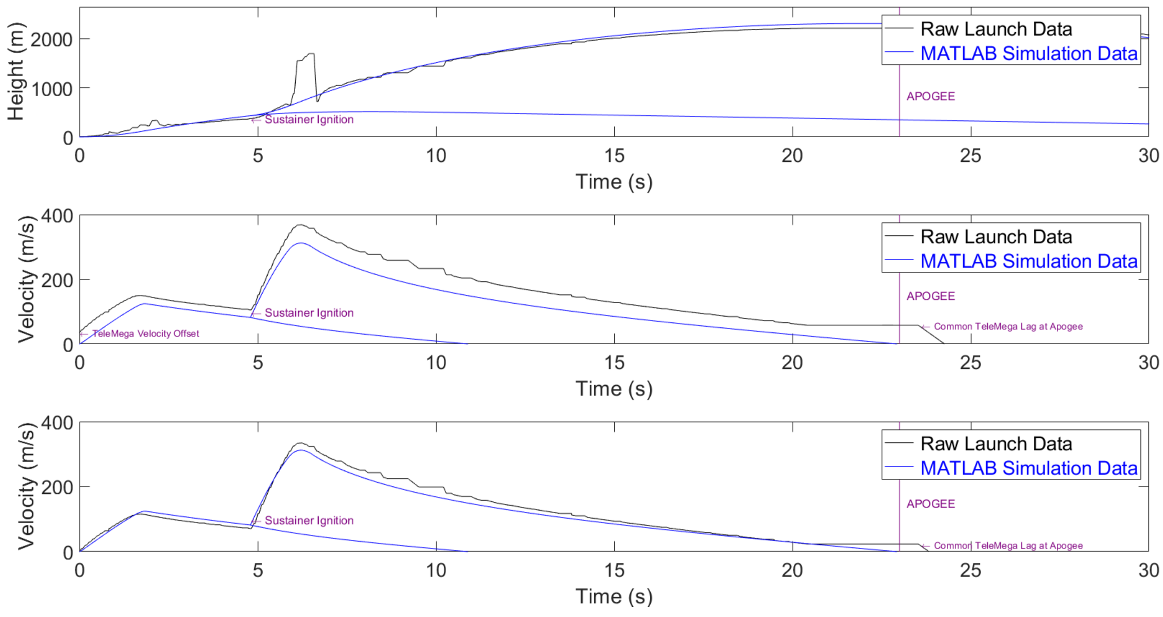


Figure 5 - A detailed view of the height and velocity versus time of our actual rocket flight recorded with out TeleMega plotted with our homemade MATLAB flight simulation software

Raw Telemega Analysis

The TeleMega uses a software called Altus Metrum that allows for communication and data retrieval from the launch computer to the TeleMega throughout the flight. This software then saves the flight, allowing us to export its data as a CSV that can then be analyzed further with MATLAB. This section provides some of the raw graphs and data tables that the Altus Metrum can produce once the flight has ended.

Figure 6 details the actual height versus time data of our flight in its raw form on Altus Metrum. These data are the exact same previously in the report except previously it was plotted in MATLAB and annotated with pertinent launch events. This graph can also clearly see the phenomena of drastic height increase from quick drops in acceleration from the sustainer and booster cutoff events. The apogee event is clear and takes place 23 seconds after launch at a final maximum height of 2215 meters or 7,267 feet.

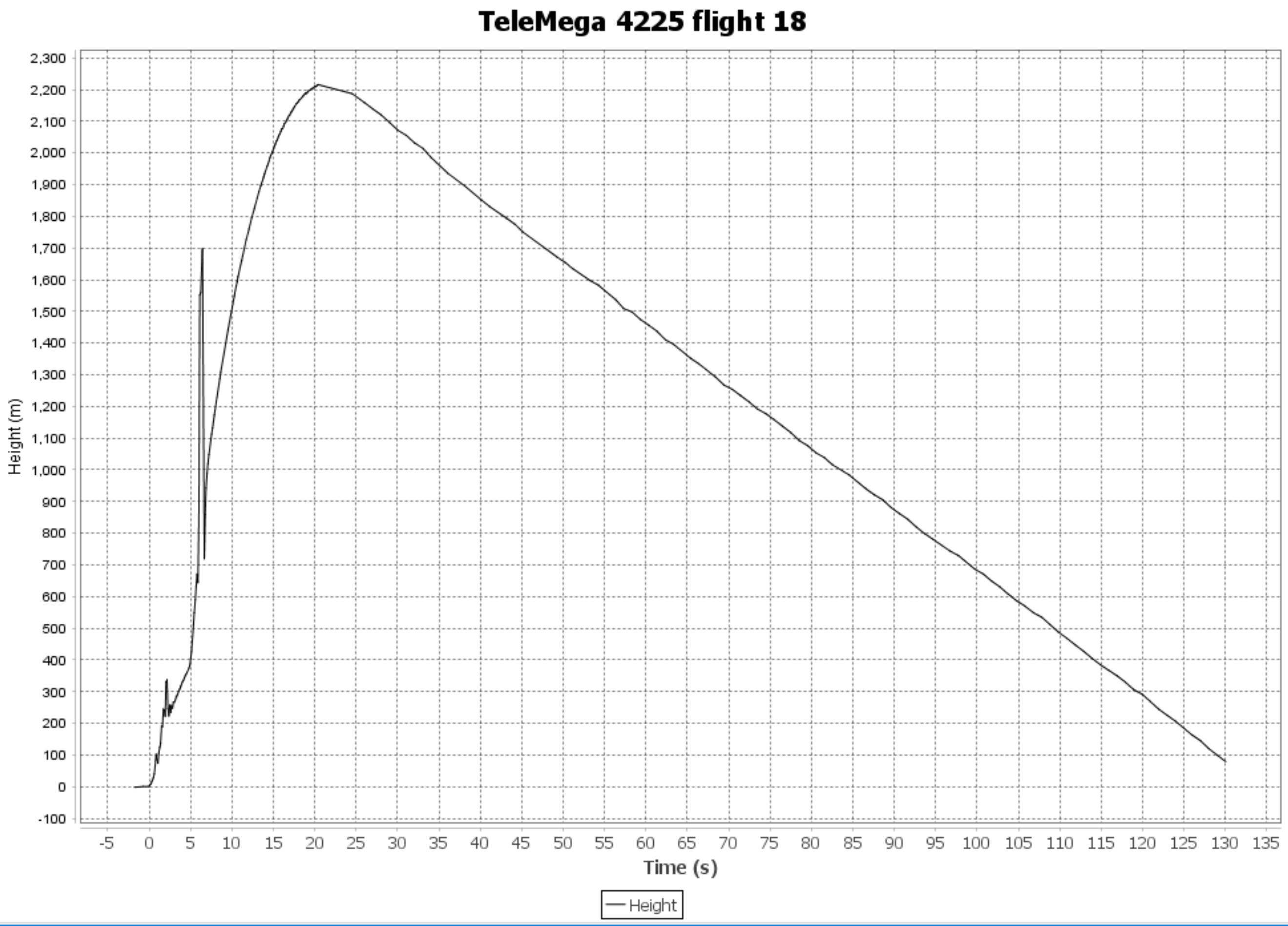


Figure 6 - Raw TeleMega flight data from the Altus Metrum software describing the rockets height versus time that was previously plotted via MATLAB

The TeleMega has GPS incorporated into its board, allowing for satellite tracking throughout the flight. The Altus Metrum software is then able to compile this data through the flight to produce a satellite view picture of the flight. Figure 7 shown below is that raw picture of our flight with the field and sustainer-landing site in view. The drift of the sustainer is clear as it slowly descended into Murdock Lake, exactly 1,275 meters (4,183 feet) away from the launch rail. It is important to note that this image is posted all over our shop to forever remember the day and is a prominent meme in many of our group chats. This launch will live in infamy for many, many years to come at UNH SEDS.

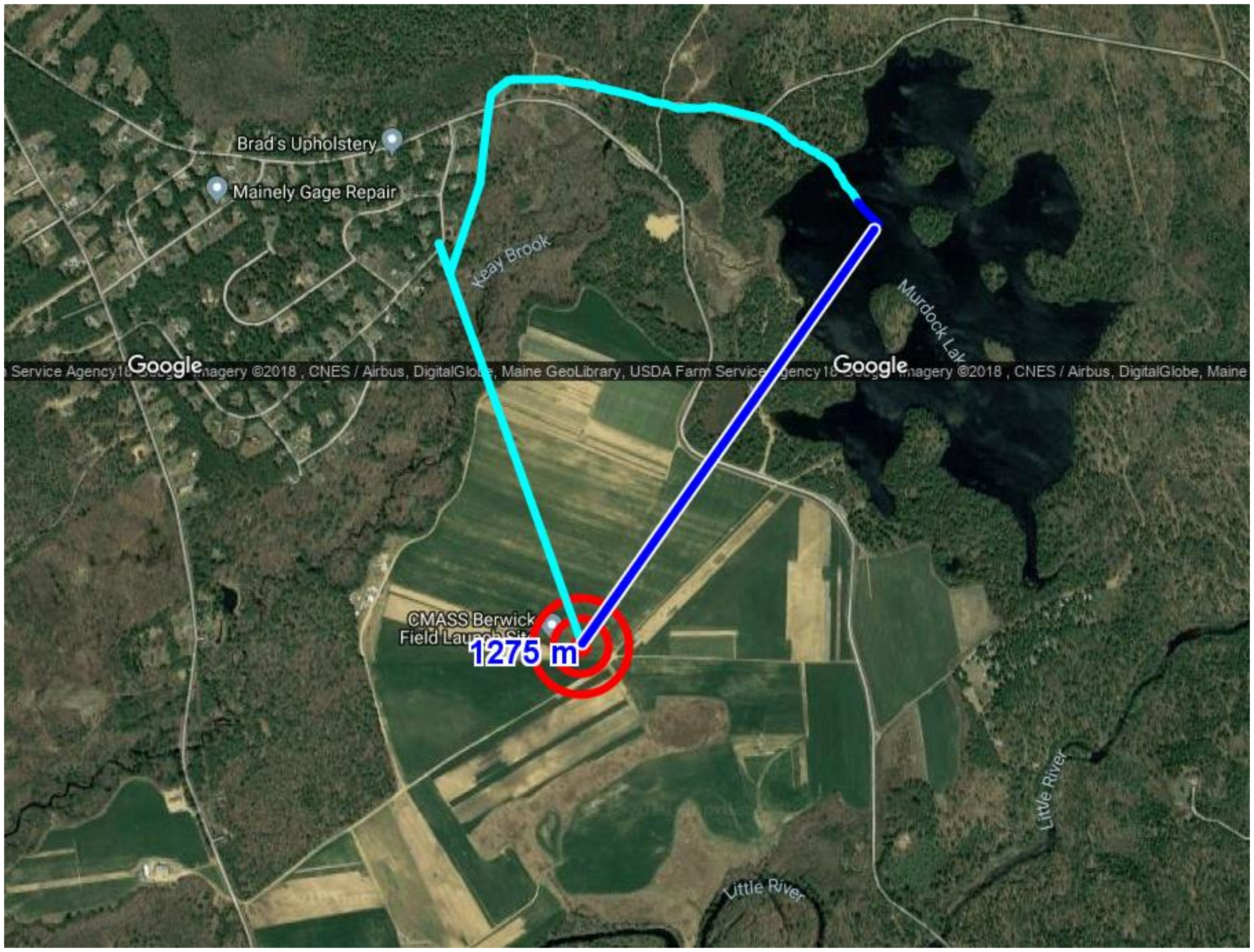
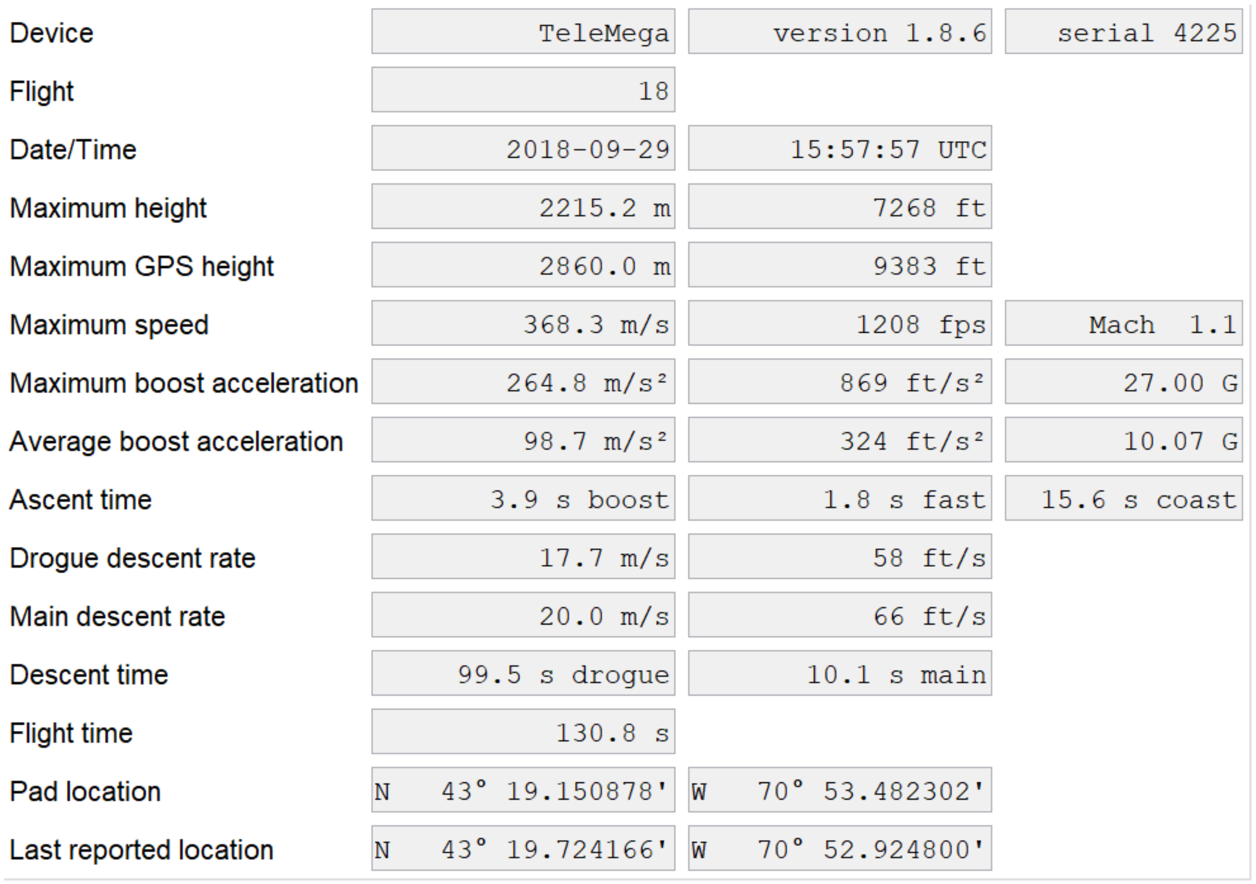

Brad's Upholstery
Mainely Gage Repair
Keay Brook
Murdock Lake
Google
Imagery ©2018 , CNES / Airbus, DigitalGlobe, Maine GeoLibrary, USDA Farm Service Agency
CMASS Berwick Field Launch Site
1275 m
Little River
Little River

Figure 7 - The TeleMega and Altus Metrum software picture of the complete satellite view of the rocket flight



Device	TeleMega	version 1.8.6	serial 4225
Flight	18		
Date/Time	2018-09-29	15:57:57 UTC	
Maximum height	2215.2 m	7268 ft	
Maximum GPS height	2860.0 m	9383 ft	
Maximum speed	368.3 m/s	1208 fps	Mach 1.1
Maximum boost acceleration	264.8 m/s²	869 ft/s²	27.00 G
Average boost acceleration	98.7 m/s²	324 ft/s²	10.07 G
Ascent time	3.9 s boost	1.8 s fast	15.6 s coast
Drogue descent rate	17.7 m/s	58 ft/s	
Main descent rate	20.0 m/s	66 ft/s	
Descent time	99.5 s drogue	10.1 s main	
Flight time	130.8 s		
Pad location	N 43° 19.150878'	W 70° 53.482302'	
Last reported location	N 43° 19.724166'	W 70° 52.924800'	

Table 1 - TeleMega's raw table output of all pertinent launch data through the whole flight

Table 1 details the raw, untouched image produced by the Altus Metrum software showing the pertinent data of the launch over its entire flight. The notable data on this table have been explained previously in the report, including a maximum height of 2215 meters or 7,268 feet (first time UNH SEDS has hit the mile-high club), a maximum speed of 1.1 Mach (first time UNH SEDS has hit the sound barrier), a 3.9 second boost and coast phase and a 1.9 second sustainer boost phase, a descent speed of 17.7 m/s (which is far slower than letting the rocket free fall), a very similar main descent rate proving that the chute release did not work properly, and the pad and last reported location of the rocket after flight. This flight pushed UNH SEDS' comfort zone, breaking barriers of what had previously been done over the last year.

Failure Analysis

Improvement Cycle for Failure Mitigation

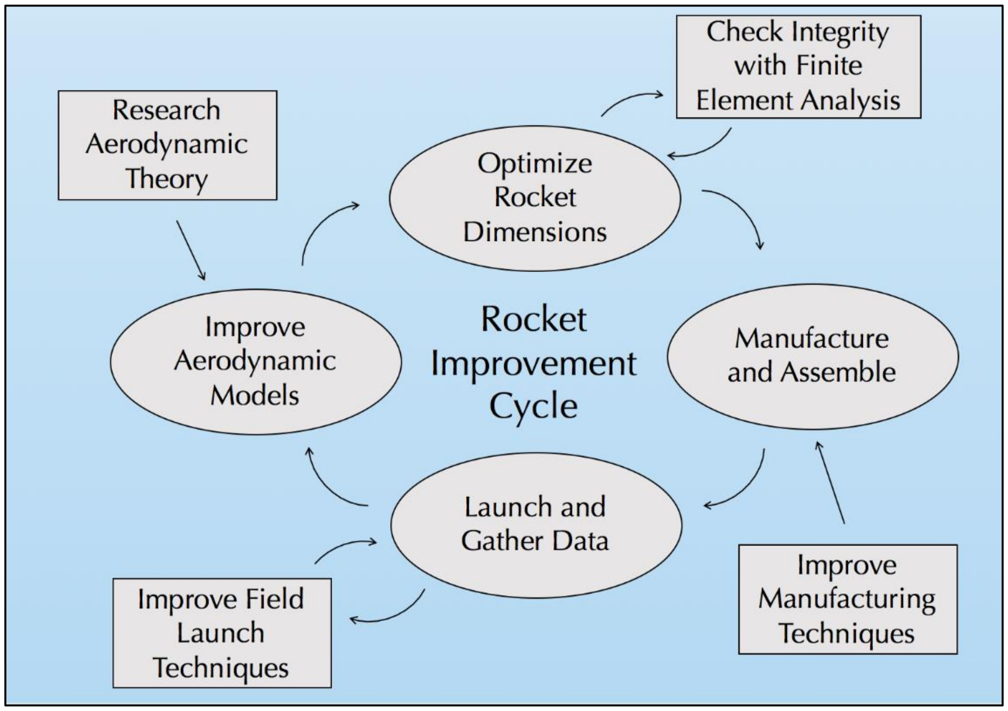

Check Integrity with Finite Element Analysis
Research Aerodynamic Theory
Optimize Rocket Dimensions
Improve Aerodynamic Models
Rocket Improvement Cycle
Manufacture and Assemble
Launch and Gather Data
Improve Field Launch Techniques
Improve Manufacturing Techniques

Figure 8. 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 constantly 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 the 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, the first two rockets having no name. 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 leading us to greater success with each flight.

Competition Launch Failure Analysis

The failure of our competition rocket flight came down to two design features: the combination of our non-optimal booster engine utilized and an insufficient launch rail length to compensate for the change in its thrust curve, and the fundamental change of our flight profile with the inclusion of a donated chute release for apogee separation.

Our first failure, the non-optimal booster engine and an insufficient launch rail length, directly resulted in the initial stability of our flight. With a below-desired takeoff velocity from the launch rail due to a non-optimal thrust profile and a non-optimal launch rail length, the speed was insufficient for stability to take hold. This instability at launch resulted in a flight profile that sent our rocket at an angle away from the zenith. The new booster engine was simulated before launch day but the team decided that, although it was slightly below what we have flown in past rocket iteration, there was still a good calculated chance that this speed would still work for the rocket in flight. The takeoff velocity was just under what was required for the fins to take hold and provide optimal stability for a straight, desired flight path. The data from our launches thus far have now given a framework on what is safe and unsafe takeoff velocities for optimal stability of flight that will be utilized for future builds.

The second failure, which was from the fundamental change in our rocket flight profile from the addition of a donated chute release for apogee separation, caused drift (which the original flight mechanics was designed to drastically lower) and pushed our rocket away from its apogee location (nearly on top of our launch location) to 1275 meters away. This failure was the result from a safety standpoint on the field. As previously discussed, the launch safety officer was worried that our parachute ejection would fail, resulting in our sustainer maintaining terminal velocity, crashing back onto the field. This change from deploying our parachute wrapped in a chute release at an apogee of 2,215 meters resulted in incredible drift. However, when it comes to any launch, safety is first. If there were even a 1 percent chance that our parachute ejection would fail, it would apply risk to all the spectators watching. We willingly agreed to add the quick release to our recovery system and change the ejection to shortly after apogee instead of a height of 300 meters during descent. Parachute ejection was nominal shortly after apogee, sending the rocket back down at a reduced descent speed with the main parachute acting as a streamer. With drift, the sustainer had an unfortunate landing directly in the middle of a small lake. For future rockets, we must ensure that the safety of all rocket launch spectators by keeping the descent speed never reaching terminal velocity with the risk of parachute deployment failure. We hope that you still consider our TeleMega data due to this unfortunate landing. The raw date file will be sent with this report to prove that no foul play has been involved with our report.

The culmination of our entire organization led us to this launch. Although it did not go perfectly, the failures we experienced were calculated risks that needed to be taken to compete. Rocketry is a noble pursuit. It is the truest profession in which failure directly leads to success. It can become discouraging that failure has been experienced more than success, but it has only pushed us more to prove that we are capable of achieving greatness. We would like to thank SEDS for organizing this competition as it has guided us to finally become rocketeers.

Multimedia Documentation

The launch took place in South Berwick, ME, on September 29th mid-day. Below are the links to the videos taken during the launch, and an assortment of pictures of the beautiful day. Several independent NAR members were present during the launch and can be contacted if needed to verify the flight.

YouTube Links

The link below is the video that tracked the booster as it landed with its recovery system. Nothing broke at landing, allowing for full recoverability of our booster.

https://youtu.be/qhKtFAfzMnM

The link below tracked the sustainer, but quickly went out of sight as it reached its apogee. Recovery of the sustainer could not be recoded as it drifted over a kilometer into a nearby lake. Data was still collected through the Telemega transmitting to our computer on the field.

https://youtu.be/kTArgz6VOts

Notable Pictures

Figure 9- Team members enjoying the launch day and preparing the rocket for launch

Figure 10- Engineering and building leads confirming rocket is go for launch

Figure 11 - The new seniors of the club experiencing their first rocket launch

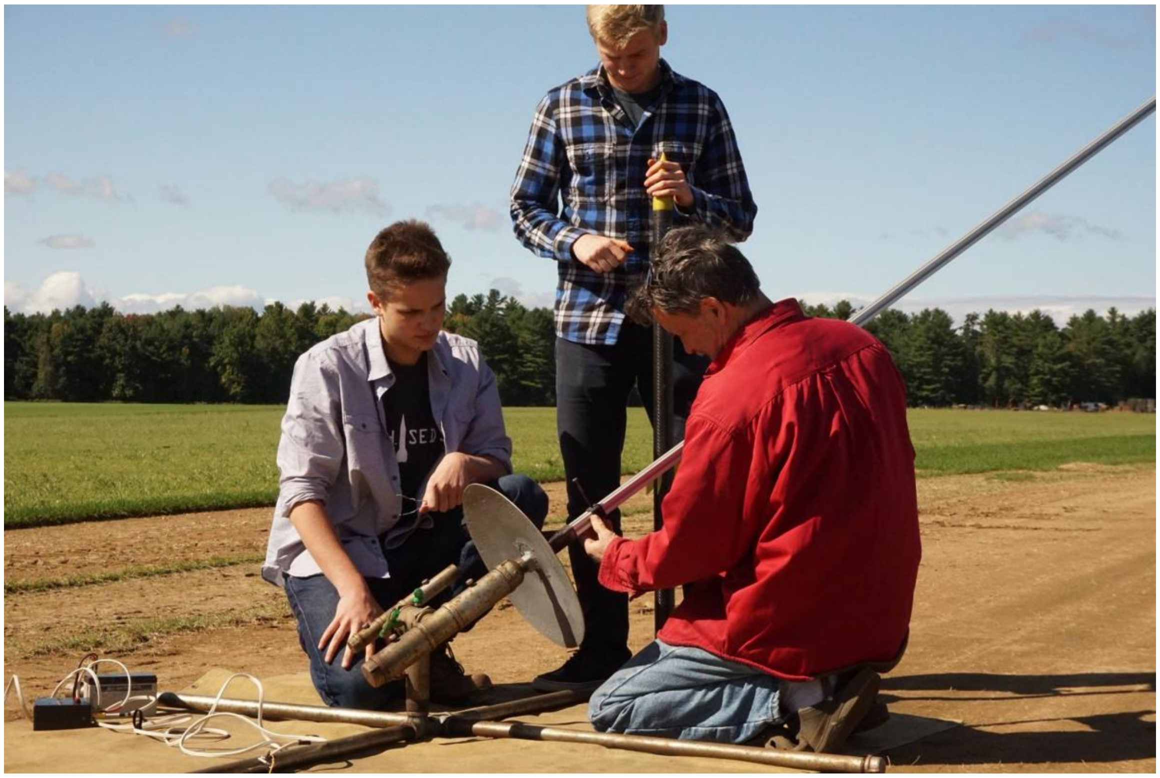


Figure 12- Attaching the multi-stage rocket to the launch pad

Figure 13- Full team picture as we approach our launch time

APPENDIX

MATLAB Simulation Code

The following code details the functions used to simulate a two-stage rocket with known parameters. The code can be easily manipulated with different constraints, such as stability, diameters, and masses. The code allowed the comparison of the flight data, MATLAB simulation data, and OpenRocket simulation data.

Launch Simulation (Parent Function)

```
clear all;
close all;

% ~ Competition Rocket 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       = 180;      % endtime
tseperation = 4.82;     % 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(482); %(tseperation/dt);
v = velocity(482); %(tseperation/dt);
a = acceleration(482); %(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



% Open Rocket Data
CompetitionRocket_OpenRocket_Sustainer = 'CR_OP_Sustainer.csv';
CompetitionRocket_OpenRocket_Booster = 'CR_OP_Booster.csv';
TimeOpenSust    = xlsread(CompetitionRocket_OpenRocket_Sustainer,'A1:A5230');
TimeOpenBoost   = xlsread(CompetitionRocket_OpenRocket_Booster,'A1:A838');
HeightOpenSust  = xlsread(CompetitionRocket_OpenRocket_Sustainer,'B1:B5230');
HeightOpenBoost = xlsread(CompetitionRocket_OpenRocket_Booster,'B1:B838');
```

```
VelocityOpenSust  = xlsread(CompetitionRocket_OpenRocket_Sustainer,'C1:C5230');
VelocityOpenBoost = xlsread(CompetitionRocket_OpenRocket_Booster,'C1:C838');

% Experimental Data
CompetitionRocket_Experimental = 'USRC_TeleMega_Flight_Data.csv';
TimeExperimental     = xlsread(CompetitionRocket_Experimental,'E10:E688');
HeightExperimentalBoost  = xlsread(CompetitionRocket_Experimental,'M10:M688');
VelocityExperimentalBoost  = xlsread(CompetitionRocket_Experimental,'N10:N688');
% TimeExperimentalBoost = TimeExperimentalBoost(1:1021);
% HeightExperimentalBoost = HeightExperimentalBoost(1:1021);


figure(1)
plot(TimeExperimental,HeightExperimentalBoost,'k')
hold on
plot(x,height,'b')
plot(x2,height2,'b')
plot(TimeOpenSust,HeightOpenSust,'r')
plot(TimeOpenBoost,HeightOpenBoost,'r')
plot([22.5,22.5],[0,2650],'Color',[.5,0,.5])
plot([0,140],[2542,2542],'r')
plot([0,140],[2310,2310],'b')
plot([0,140],[2215,2215],'k')
text(23,1250,'APOGEE','Color',[.5,0,.5],'FontSize',12)
text(60,2593,'OpenRocket Apogee Prediction','Color','r','FontSize',12)
text(60,2360,'MATLAB Apogee Prediction','Color','b','FontSize',12)
text(60,2265,'Actual Experimental Apogee','Color','k','FontSize',12)

text(4.8,374,'\leftarrow Sustainer Ignition','Color',[.5,0,.5],'FontSize',10)
text(49.2,310,'\leftarrow Simulated Parachute Deployment','Color',[.5,0,.5],'FontSize',10)
text(67.92,1334,'\leftarrow Wrapped Parachute','Color',[.5,0,.5],'FontSize',10)
text(126,206.8,'Lake Landing','Color',[.5,0,.5],'FontSize',10)
text(82.2,60,'Simulated Landing','Color',[.5,0,.5],'FontSize',10)

xlabel('Time (s)','FontSize',22)
set(gca,'fontsize',20)
ylabel('Height (m)','FontSize',22)
set(gca,'fontsize',20)
lgd = legend('\color{black} Raw Launch Data','\color{blue} MATLAB Simulation
Data','\color{red} OpenRocket Simulation Data');
lgd.FontSize = 20;
xlim([0,140])
ylim([0,2650])


figure(2)
plot(TimeExperimental,HeightExperimentalBoost,'k')
hold on
plot(x,height,'b')
plot(x2,height2,'b')
plot(TimeOpenSust,HeightOpenSust,'r')
plot(TimeOpenBoost,HeightOpenBoost,'r')
plot([22,22],[0,2650],'Color',[.5,0,.5])
plot([0,140],[2513,2513],'r')
plot([0,140],[2310,2310],'b')
plot([0,140],[2215,2215],'k')
text(23,1250,'APOGEE','Color',[.5,0,.5],'FontSize',12)
text(60,2563,'OpenRocket Apogee Prediction','Color','r','FontSize',12)
text(60,2360,'MATLAB Apogee Prediction','Color','b','FontSize',12)
text(60,2265,'Actual Experimental Apogee','Color','k','FontSize',12)
text(2.20,337,'Pressure Change from Booster Cutoff','Color',[.5,0,.5],'FontSize',10)

text(4.8,374,'\leftarrow Sustainer Ignition','Color',[.5,0,.5],'FontSize',12)


xlabel('Time (s)','FontSize',22)
set(gca,'fontsize',20)
ylabel('Height (m)','FontSize',22)
set(gca,'fontsize',20)
lgd = legend('\color{black} Raw Launch Data','\color{blue} MATLAB Simulation
Data','\color{red} OpenRocket Simulation Data');
lgd.FontSize = 20;
xlim([0,10])
ylim([0,500])

figure(3)
plot(TimeExperimental,VelocityExperimentalBoost,'k')
hold on
plot(x,velocity,'r')
plot(x2,velocity2,'r')
plot(TimeOpenSust,VelocityOpenSust,'b')
plot(TimeOpenBoost,VelocityOpenBoost,'b')
plot([23,23],[-150,500],'Color',[.5,0,.5])
plot([0,60],[368,368],'k')
plot([0,60],[310,310],'b')
plot([0,60],[281,281],'r')
plot([6.2,6.2],[-150,500],'Color',[.5,0,.5])
text(23.2,200,'APOGEE','Color',[.5,0,.5],'FontSize',12)
text(12,376,'Actual Experimental MaxV','Color','k','FontSize',12)
text(12,318,'MATLAB MaxV Prediction','Color','b','FontSize',12)
text(12,289,'OpenRocket MaxV Prediction','Color','r','FontSize',12)
text(6.2,100,'MaxV','Color',[.5,0,.5],'FontSize',10)

text(23.6,57.64,'\leftarrow Common TeleMega Lag at Apogee','Color',[.5,0,.5],'FontSize',10)
text(49.1,-50,'\leftarrow Simulated Parachute Deployment','Color',[.5,0,.5],'FontSize',10)
text(42,-115,'Simulated Free-Falling','Color',[.5,0,.5],'FontSize',10)

xlabel('Time (s)','FontSize',22)
set(gca,'fontsize',20)
ylabel('Velocity (m/s)','FontSize',22)
set(gca,'fontsize',20)
lgd = legend('\color{black} Raw Launch Data','\color{blue} MATLAB Simulation
Data','\color{red} OpenRocket Simulation Data');
lgd.FontSize = 20;
xlim([0,60])
ylim([-150,500])
```

```
figure(4)
subplot(2,1,2)
plot(TimeExperimental,VelocityExperimentalBoost,'k')
hold on
plot(TimeOpenSust,VelocityOpenSust,'b')
plot(TimeOpenBoost,VelocityOpenBoost,'b')
plot([23,23],[-150,500],'Color',[.5,0,.5])
text(23.2,200,'APOGEE','Color',[.5,0,.5],'FontSize',12)
text(23.6,57.64,'\leftarrow Common TeleMega Lag at Apogee','Color',[.5,0,.5],'FontSize',10)
text(0,35,'\leftarrow TeleMega Velocity Offset','Color',[.5,0,.5],'FontSize',10)
xlabel('Time (s)','FontSize',22)
set(gca,'fontsize',20)
ylabel('Velocity (m/s)','FontSize',22)
set(gca,'fontsize',20)
lgd = legend('\color{black} Raw Launch Data','\color{blue} MATLAB Simulation Data');
lgd.FontSize = 20;
xlim([0,60])
ylim([-150,500])

subplot(2,1,1)

plot(TimeExperimental,HeightExperimentalBoost,'k')
hold on
plot(x,height,'b')
plot(x2,height2,'b')
plot([23,23],[0,2650],'Color',[.5,0,.5])
text(23.2,1250,'APOGEE','Color',[.5,0,.5],'FontSize',12)

xlabel('Time (s)','FontSize',22)
set(gca,'fontsize',20)
ylabel('Height (m)','FontSize',22)
set(gca,'fontsize',20)
lgd = legend('\color{black} Raw Launch Data','\color{blue} MATLAB Simulation Data');
lgd.FontSize = 20;
xlim([0,60])
ylim([0,2650])


figure(5)
subplot(3,1,2)
plot(TimeExperimental,VelocityExperimentalBoost,'k')
hold on
plot(TimeOpenSust,VelocityOpenSust,'b')
plot(TimeOpenBoost,VelocityOpenBoost,'b')
plot([23,23],[-150,500],'Color',[.5,0,.5])
text(23.2,150,'APOGEE','Color',[.5,0,.5],'FontSize',12)
text(23.6,57.64,'\leftarrow Common TeleMega Lag at Apogee','Color',[.5,0,.5],'FontSize',10)
text(0,35,'\leftarrow TeleMega Velocity Offset','Color',[.5,0,.5],'FontSize',10)
text(4.8,100,'\leftarrow Sustainer Ignition','Color',[.5,0,.5],'FontSize',12)
xlabel('Time (s)','FontSize',22)
set(gca,'fontsize',20)
```

```
ylabel('Velocity (m/s)','FontSize',22)
set(gca,'fontsize',20)
lgd = legend('\color{black} Raw Launch Data','\color{blue} MATLAB Simulation Data');
lgd.FontSize = 20;
xlim([0,30])
ylim([0,400])

subplot(3,1,1)

plot(TimeExperimental,HeightExperimentalBoost,'k')
hold on
plot(x,height,'b')
plot(x2,height2,'b')
plot([23,23],[0,2650],'Color',[.5,0,.5])
text(23.2,850,'APOGEE','Color',[.5,0,.5],'FontSize',12)
text(4.8,374,'\leftarrow Sustainer Ignition','Color',[.5,0,.5],'FontSize',12)
xlabel('Time (s)','FontSize',22)
set(gca,'fontsize',20)
ylabel('Height (m)','FontSize',22)
set(gca,'fontsize',20)
lgd = legend('\color{black} Raw Launch Data','\color{blue} MATLAB Simulation Data');
lgd.FontSize = 20;
xlim([0,30])
ylim([0,2650])


subplot(3,1,3)
plot(TimeExperimental,VelocityExperimentalBoost-34.2,'k')
hold on
plot(TimeOpenSust,VelocityOpenSust,'b')
plot(TimeOpenBoost,VelocityOpenBoost,'b')
plot([23,23],[-150,500],'Color',[.5,0,.5])
text(23.2,150,'APOGEE','Color',[.5,0,.5],'FontSize',12)
text(23.6,23,'\leftarrow Common TeleMega Lag at Apogee','Color',[.5,0,.5],'FontSize',10)
text(4.8,100,'\leftarrow Sustainer Ignition','Color',[.5,0,.5],'FontSize',12)
xlabel('Time (s)','FontSize',22)
set(gca,'fontsize',20)
ylabel('Velocity (m/s)','FontSize',22)
set(gca,'fontsize',20)
lgd = legend('\color{black} Raw Launch Data','\color{blue} MATLAB Simulation Data');
lgd.FontSize = 20;
xlim([0,30])
ylim([0,400])
```

Get Acceleration

```matlab
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
```

Get Acceleration 2

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

Mass_Booster_Burnout = .596; % Weight of flown booster, constant (kg)
g = 9.81;                    % Gravity through flight (constant < 10,000 feet)
f = 0;                       % No upwards thrust
d = GetDrag2 ( v,h );        % Drag of booster through flight
a = ( f - Mass_Booster_Burnout*g - d )/ Mass_Booster_Burnout ;     % Newton's Second Law with
Drag, Gravity and no Thrust
end
```

Get Thrust

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

% ~ Competition Rocket Engines ~

% H159 - Booster Engine data (taken from thrustcurve.org)
xb_data = [0, 0.012, 0.032, .103, .171, .299, .511, .717, 1.272,
1.424,1.519,1.584,1.632,1.727,1.768,1.834,1.865,1.9]; % s
yb_data = [0, 117, 158 , 177 ,  184,  185, 178, 179 , 162  , 159,152,149,146,89,109,25,10,0];
% N

% I204  - Sustainer Engine data (taken from thrustcurve.org)
xs_data_Raw = [0 0.01 0.012 0.03 0.3 0.5 0.7 1 1.1 1.2 1.3 1.4 1.5 1.6 1.72];
xs_data=xs_data_Raw+xb_data(end)+2.9;
ys_data = cosd(35).*[0 100 356 310 286 270 251 228 215 165 125 95 52 36 0]; % Angle accounted
for not stable from boost, increasing our tilt


% 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;


elseif t > xb_data(17) && t <= xb_data(18)
```

```
    b17s = yb_data(18) - ((yb_data(18) - yb_data(17))/(xb_data(18) -
xb_data(17)))*xb_data(18);
    thrust = ((yb_data(18) - yb_data(17))/(xb_data(18) - xb_data(17)))*t + b17s;



% ~ 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;
elseif t > xs_data(14) && t <= xs_data(15)
    b14b = ys_data(15) - ((ys_data(15) - ys_data(14))/(xs_data(15) -
xs_data(14)))*xs_data(15);
    thrust = ((ys_data(15) - ys_data(14))/(xs_data(15) - xs_data(14)))*t + b14b;
else
    thrust = 0;
end


end
```

Get Mass

```matlab
function [mass] = GetMass ( t )

% ~ AETHER4 ~

Sustainer_Mass = 1.229; %kg
Booster_Mass   = 0.873; %kg
Mboostprop     = 0.187; %kg
Msustprop      = 0.185; %kg
initialMass = Sustainer_Mass + Booster_Mass; %kg
initialSustMass = Sustainer_Mass; %kg

burnTimeBoost  = 1.9;              % sec
burnTimeSust   = 1.72;             % sec
startTimesust = burnTimeBoost + 2.9; % sec
startTimecoast = startTimesust + burnTimeSust; %sec

% Conditional to print total mass of rocket at any given time.
if (t>=0 && t<burnTimeBoost)
    mass = initialMass - Mboostprop *(t/burnTimeBoost);
elseif (t>=burnTimeBoost && t<startTimesust)
    mass = initialMass - Mboostprop;
elseif (t>=startTimesust && t< startTimesust + burnTimeSust)
    mass = initialSustMass - Msustprop * (t/(burnTimeSust + startTimesust));
elseif (t>startTimecoast)
    mass = initialSustMass - Msustprop;
else
    mass = initialMass;
end
end
```

Get Drag

```
function drag = GetDrag (v,h,t)
% calculate total drag for the sustainer and booster, carries to
% sustainer then to recovery.

% Competition Rocket, Carbon Fiber


Seperation_Time = 4.82;
D = .0474;    % (m) Diameter of Nosecone


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

if v >= 0
    if t <= Seperation_Time
        l = 1.26;  % length of full rocket (m)
    elseif t > Seperation_Time
        l = 0.792; % length of sustainer body tube (m)
    end

    % ~ SKIN FRICTION DRAG ~

    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
    t = 0.005; % fin thickness
    c = 0.06; % aerodynamic cord length
    finbase = 0.05;
    finheight = 0.035;
    finarea_sf = 0.5*finbase*finheight;
    fin_totalarea_sf = finarea_sf*12; % 12 sides of fins
    bodytube_area = 2*pi*r*l;   % surface area
    crosssection_area = pi*r^2; % cross-sectional area
    ref_area_sf = fin_totalarea_sf + 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)/c)*fin_totalarea_sf)/ref_area_sf;

    Fdrag_sf = CDsf*(1/2)*rho*v^2*(ref_area_sf);

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

    % Fin Pressure Drag
    LEA = 77; % leading edge angle
    if M < 0.9
        CDpd_fins = ((1 - M^2)^(-0.417) - 1)*cosd(LEA)^2;
    elseif M > 0.9 && M < 1
        CDpd_fins = (1 - 1.785*(M - 0.9))*cosd(LEA)^2;
    elseif M > 1.0
        CDpd_fins = (1.214 - 0.502/(M^2) + 0.1095/(M^4))*cosd(LEA)^2;
    end

    Fpd_fins = CDpd_fins*(1/2)*rho*v^2*(fin_totalarea_sf);
    Fdrag_pd = Fpd_nosecone + Fpd_fins;

    % ~ 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);

    % ~ TOTAL DRAG ~
    drag = Fdrag_sf + Fdrag_pd + Fdrag_bd;



% ~ RECOVERY ~
elseif v < 0
    D = 0.05; % diameter of parachute (m)
    k = 1.0;
    drag = -k * 0.5 * rho * v^2 * pi/4 * D^2;

    if h < 300
```

```
        D = 0.6;
        k = 1.0;
        drag = -k * 0.5 * rho * v^2 * pi/4 * D^2;

    end

end

end
```

Get Drag 2

```
function drag = GetDrag2 (v,h)
% calculate total drag for the booster after seperation

% Competition Rocket, Carbon Fiber


Seperation_Time = 4.82;
D = .0474;    % (m) Diameter of Nosecone


pi = 3.14;
mu = 1.79e-5;
rho = 1.217*exp(-h/8500);
kv = mu/rho;
l = 46.8;

% ~ SKIN FRICTION DRAG ~

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
t = 0.005; % fin thickness
c = 0.06; % aerodynamic cord length
finbase = 0.05;
finheight = 0.035;
finarea_sf = 0.5*finbase*finheight;
fin_totalarea_sf = finarea_sf*6; % 12 sides of fins
bodytube_area = 2*pi*r*l;    % surface area
crosssection_area = pi*r^2; % cross-sectional area
ref_area_sf = fin_totalarea_sf + 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)/c)*fin_totalarea_sf)/ref_area_sf;

Fdrag_sf = CDsf*(1/2)*rho*v^2*(ref_area_sf);

% ~ 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
LEA = 75; % leading edge angle
if M < 0.9
    CDpd_fins = ((1 - M^2)^(-0.417) - 1)*cosd(LEA)^2;
elseif M > 0.9 && M < 1
    CDpd_fins = (1 - 1.785*(M - 0.9))*cosd(LEA)^2;
elseif M > 1.0
    CDpd_fins = (1.214 - 0.502/(M^2) + 0.1095/(M^4))*cosd(LEA)^2;
end

Fpd_fins = CDpd_fins*(1/2)*rho*v^2*(fin_totalarea_sf);
Fdrag_pd = Fpd_nosecone + Fpd_fins;

% ~ 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);

% ~ TOTAL DRAG ~
drag = Fdrag_sf + Fdrag_pd + Fdrag_bd;



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

end
```