**A group of logos on a black background

Description automatically generatedA logo with white text and red and blue circle

Description automatically generated**

Arrow

Description automatically generated with low confidence

**Flight Readiness Review Addendum**

**FAMU-FSU College of Engineering**

**2525 Pottsdamer Street**

**Tallahassee, FL 32310**

**4/1/2024**

[1 Summary of FRR Addendum 4](#_Toc162818981)

[1.1 Team Summary 4](#_Toc162818982)

[1.1.1 Team Name 4](#_Toc162818983)

[1.1.2 Mailing Address 4](#_Toc162818984)

[1.1.3 Team Mentor 4](#_Toc162818985)

[1.1.4 Team Contact Information 4](#_Toc162818986)

[1.1.5 Time Allotted to FRR Addendum 5](#_Toc162818987)

[1.2 Purpose of Flight 5](#_Toc162818988)

[1.3 Flight Summary Information 5](#_Toc162818989)

[1.3.1 Date of Flight 5](#_Toc162818990)

[1.3.2 Location of Flight 5](#_Toc162818991)

[1.3.3 Final Motor Choice 5](#_Toc162818992)

[1.3.4 Ballast Flown 5](#_Toc162818993)

[1.3.5 Final Payload Flown 5](#_Toc162818994)

[1.3.6 Air-brake System 5](#_Toc162818995)

[1.3.7 Official target altitude 5](#_Toc162818996)

[1.3.8 Predicted altitude from simulations. 5](#_Toc162818997)

[1.3.9 Measured altitude. 5](#_Toc162818998)

[1.3.10 Size and Mass of Individual Sections 5](#_Toc162818999)

[1.3.11 Recovery System 6](#_Toc162819000)

[1.3.12 Rail Size 6](#_Toc162819001)

[1.4 Changes made since FRR. 6](#_Toc162819002)

[1.1.1 Vehicle Criteria 6](#_Toc162819003)

[1.1.2 Payload Criteria 7](#_Toc162819004)

[1.1.3 Project Plan 7](#_Toc162819005)

[2 Vehicle Demonstration Re-flight 7](#_Toc162819006)

[2.1 Functional systems 7](#_Toc162819007)

[2.1.1 Avionics 7](#_Toc162819008)

[2.1.2 Recovery System 8](#_Toc162819009)

[2.1.3 Airframe 9](#_Toc162819010)

[2.2 Failures 9](#_Toc162819011)

[2.3 Payload simulation 10](#_Toc162819012)

[2.4 Flight profile data 10](#_Toc162819013)

[2.5 As-landed configuration 12](#_Toc162819014)

[2.6 Kinetic energy at landing 16](#_Toc162819015)

[2.7 Analysis 17](#_Toc162819016)

[2.8 Drag Coefficient Estimate 20](#_Toc162819017)

[2.9 Damaged Hardware 22](#_Toc162819018)

[2.10 Lessons Learned 22](#_Toc162819019)

[3 Appendix 23](#_Toc162819020)

[3.1 Additional Figures 23](#_Toc162819021)

[3.2 MATLAB Scripts 25](#_Toc162819022)

# Summary of FRR Addendum

## Team Summary

### Team Name

The team’s name is The Zenith Program.

### Mailing Address

The mailing address for the Florida Agricultural & Mechanical University – Florida State University College of Engineering Zenith Program is as follows:

FAMU-FSU AIAA

2525 Pottsdamer Street, Suite B111

Tallahassee, FL 32310

### Team Mentor

#### Mr. Tom McKeown

* **Title:** Board Member, Spaceport Rocketry Association (NAR #342 / TRA #73)
* **Email:** mckeownt@ix.netcom.com
* **Phone:** (321) 266-1928
* **NAR Flyer Number:** 57205
* **TRA Flyer Number:** 01922
* **NAR/TRA Certification Level:** Level 2

### Team Contact Information

#### Atzimba Avellaneda

* **Title:** Safety Officer/Fabrication Lead
* **Email:** aa20gx@fsu.edu
* **Phone:** (561) 248-3677

#### Nicholas Hux

* **Title:** Airframe Design Lead
* **Email:** njh20b@fsu.edu
* **Phone:** (573) 703-3123

#### Jacob Miller

* **Title:** Avionics Lead
* **Email:** jam20d@fsu.edu
* **Phone:** (352) 978-0808

#### Jacob Schmitt

* **Title:** Team Lead
* **Email:** [jds19e@fsu.edu](mailto:jds19e@fsu.edu)
* **Phone:** (417) 719-2589

#### Connor Zhou

* **Title:** Recovery Lead
* **Email:** caz20@gmail.com
* **Phone:** (407) 865-2149

### Time Allotted to FRR Addendum

The team began working directly on the Flight Readiness Review (FRR) Addendum on March 20th, 2024. This addendum is being written after the full-scale vehicle has been completed and successfully launched, so the team has allotted roughly 15 hours to the FRR addendum.

## Purpose of Flight

The Zenith Program conducted a Vehicle Demonstration Re-flight after the submission of the FRR Report to fulfill the requirements for a successful full-scale Vehicle Demonstration Flight and become eligible to perform their exhibition launch in Huntsville, Alabama.

## Flight Summary Information

### Date of Flight

The Vehicle Demonstration Re-flight occurred on March 9th, 2024.

### Location of Flight

The Vehicle Demonstration Re-flight occurred at the Spaceport Rocketry Association in Palm Bay, FL.

### Final Motor Choice

The leading choice for a motor is the AeroTech L2200G.

### Ballast Flown

No ballast was flown.

### Final Payload Flown

A payload simulator was flown. This payload simulator was a bag of sand weighing in at 5.04 lbs, meeting the 5 lb weight requirement.

### Air-brake System

N/A. No Air-brake system was used.

### Official target altitude

The official target altitude was 4340 ft.

### Predicted altitude from simulations.

The predicted altitude from simulations was 4340ft.

### Measured altitude.

The measured altitude was 4311ft.

### Size and Mass of Individual Sections

The current design of the vehicle is 104 inches from the tip of the nosecone to the end of the tail cone, while the outer diameter is 6.17 inches (in.). The masses calculated are using a G12 fiberglass body tube Nylon 12 for the nosecone and fins, and an aluminum tail cone. The total mass of the rocket is designed to be approximately 55 (lbs.) with the motor, and 44.5 lbs. without. The vehicle will be separated into 3 sections. Section 1, the top section with the payload, weighs 23.7 lbs. Section 2, the avionics bay section, weighs 5.48 lbs. Section 3, the bottom section fully loaded, weighs 25.8 lbs.

Table 1: Launch vehicle masses.

|  |  |
| --- | --- |
| Launch Vehicle | Mass (lbs.) |
| Dry Mass without ballast | 44.5 |
| Wet Mass | 55 |
| Burnout Mass | 46.43 |
| Landing Mass | 41.43 |

|  |  |
| --- | --- |
| Vehicle Section | Mass (lbs.) |
| Nosecone and fore bay with payload | 23.7 |
| Avionics bay | 5.48 |
| Thrust structure and aft bay | 25.8 |
| Total | 54.98 |

### Recovery System

The team plans to deploy a drogue parachute at apogee and a main chute at 600 ft. The drogue parachute is the Fruity Chute 24-inch Classic Elliptical, and the main chute is the Fruity Chutes Iris Ultra 144-inch Compact Parachute. Chute deployment is controlled by a redundant altimeter setup using an Altus Metrum TeleMetrum and Entacore AIM 3. The altimeters are powered by separate 3.7V LiPo batteries and send signals to the Eagle CO2 ejection system to separate the vehicle.

### Rail Size

The launch vehicle utilized 12 ft. long 1515 launch rails for all of its flights.

## Changes made since FRR.

### Vehicle Criteria

Table 2: Changes made to vehicle criteria.

|  |  |
| --- | --- |
| Description of Change | Reason for the Change |
| Fix the broken fin. | One fin had broken from the previous flight. Fiberglass cloth and two-part epoxy was used to fix the fin. |
| Parachute line length extended for one singular line. | One line was shorter than the others, causing the parachute to not fully extend. |

### Payload Criteria

Table 3: Changes made to payload criteria.

|  |  |
| --- | --- |
| Description of Change | Reason for the Change |
| Using 5-lb mass for full-scale flights | Lack of funding |

Due to a lack of funding for the fabrication of a feasible payload design, there will be no final fabricated payload. The team will be using a 5-lb mass to simulate the weight of the payload experiment for the full-scale flights. Thus, no testing can commence.

### Project Plan

There were no changes made to the project plan.

# Vehicle Demonstration Re-flight

## Functional systems

### Avionics

The avionics system functioned as intended during and after the launch. The avionics system uses a redundant dual deployment system, with two altimeters from different manufacturers. These altimeters are powered by two separate LiPo batteries, and each is connected to a CO2 ejection system that can separate each bay of the rocket. The altimeters were programmed to not fire charges at the same time, using a two-second apogee and a 50 ft altitude delay on the main parachute. The primary altimeter, the TeleMetrum, was set to fire the main charge at 600ft, with the secondary altimeter, AIM3, at 550 ft. An overview of the electronics and wiring of the system is seen below.

A computer chip with text and numbers

Description automatically generated with medium confidence

Figure 1: Flight computer layout of avionics sled.

A diagram of a device

Description automatically generated

Figure 2: Avionics electrical wiring diagram.

Each component fits into a designated slot of the avionics sled, where they are secured by screwing into the sled and using zip ties for added security. There is also an AirTag (not pictured), secured to the other side of the sled to ensure recovery. All electronic connections are checked to be secure before flight, and batteries are charged before each launch. This system functioned nominally, as no noticeable movement around the avionics sled was observed by any battery or altimeter. All wiring remained secure during and after launch, allowing for full functionality of both altimeters. Each altimeter also recorded flight data as intended, which will be analyzed in later sections. The radio link between the TeleMetrum altimeter and the ground station also functioned as intended, allowing for live flight updates and GPS location of the vehicle at all times.

### Recovery System

The recovery system functioned as intended and met all required performance criteria. After reaching apogee 16.6 seconds into the flight, the apogee ejection charges successfully separated the aft section of the rocket and allowed the drogue parachute to fully deploy. The main ejection charges successfully separated the fore section of the rocket and the main parachute fully deployed. All in all, the recovery system resulted in kinetic energy at impact of 65.2 ft-lbf. for the heaviest section, a drifted distance of 774 ft., and a descent time of about 81.4 seconds, all of which meet the recovery performance thresholds (3.3, 3.11, and 3.12, respectively) outlined in the NASA Student Launch Handbook.

### Airframe

After a fin sheared on flight two, the team conducted thorough research on the best fix on the best way to fix this. The team concluded that using fiberglass cloth and two-part epoxy was the safest and most secure method to fix the fin. Below is a zoomed-in picture of this fixed fin.

A person touching a metal object

Description automatically generated

Figure 3: Fixed Fin.

In fixing the broken fin, the connection points between the broken fins were lightly sanded so that the cracks aligned between the two more cleanly, and the two pieces fit into each other properly. Epoxy was applied to the broken ends, and they were held together. Epoxy was brushed onto the faces of the fin so that the fiberglass cloth could be laid onto the fin for extra security. One rectangular piece of fiberglass cloth was laid on either side of the fin and more epoxy was applied for proper adhesion. One larger piece of the cloth was wrapped around the fin to ensure that the edges would be susceptible to a break. Even more epoxy was applied over that to encapsulate the fiberglass cloth. After it was fully cured, team members sanded down the fin and epoxy with 120 grit and then 400 grit to achieve a smooth finish.

After this previous third test flight, there was no damage to the airframe in any manner.

## Failures

After thorough analysis, the team was unable to identify any hardware or software failures during or after the flight. Each system performed nominally, resulting in a stable flight profile with successful, predictable recovery events. The Vehicle Demonstration Re-flight did however result in two altimeter irregularities for landing detection. The TeleMetrum logged a total flight time of 164.43 seconds, which is about 66.43 seconds off the actual flight time of 98 seconds. The AIM3 detected landing slightly early as it logged a flight time of 86.8 seconds, which is 11.2 seconds early from the actual flight time.

It is important to note that the vehicle (specifically the avionics bay) landed incredibly softly in tall grass. It is predicted by the team that wind gusts through the static port holes of the avionics bay could have impacted the barometers on the altimeters. However, it is also important to note that exact flight times are easily found from flight videos and raw altimeter data and that each altimeter properly captured the entire flight profile of the rocket.

## Payload simulation

An active payload was not flown for the Vehicle Demonstration Re-flight. Instead, the payload mass was simulated with a 5 lb. sandbag. The sandbag was quick-linked to the shock cord, close to the nosecone U-bolt. The payload was retained within the bay and stationary throughout the flight.

A bag of green foil on a scale

Description automatically generated

Figure 4: Simulated mass payload.

The payload mass simulator allowed the team to test the capability of the launch vehicle to hit the target apogee with the minimum mass requirement of this year’s payload challenge. The sandbag fits snugly within the upper payload bay, allowing for minimal movement of the payload within the launch vehicle.

## Flight profile data

The flight data from the TeleMetrum and AIM3 was exported and used to create the following altitude versus time plots in MATLAB shown in the figures below.

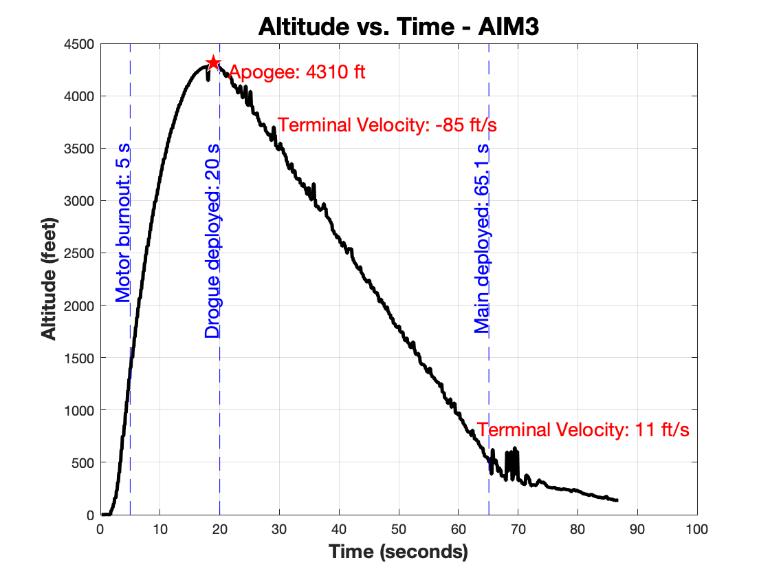
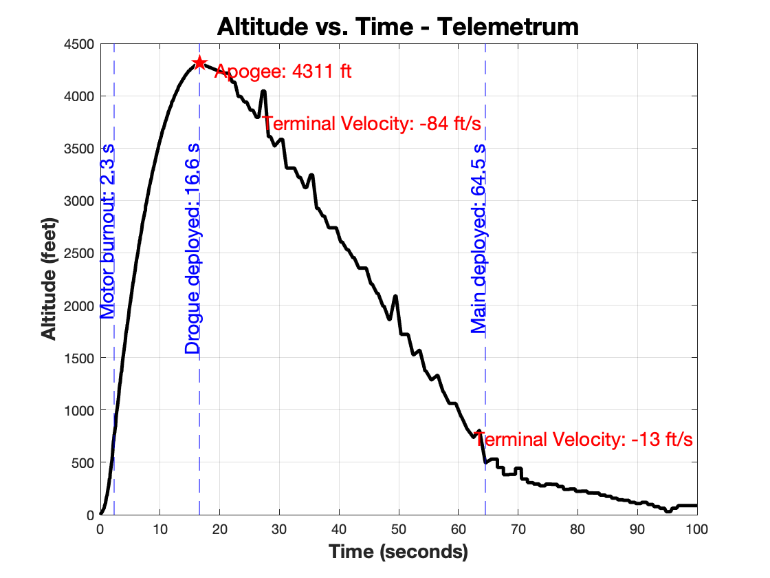


Figure 5: Altitude vs. Time Flight 3.

It is important to note that these plots were created using data that was manipulated to remove altitude spikes at apogee. Both altimeters detected a sudden spike in altitude after the vertex of the flight path had been reached, which also happened to be when the drogue charges fired. These spikes appeared to be unrealistic based on logic and video evidence. The team believes this jump is from the disturbance created by the first drogue ejection charge. This is supported by altimeter data, as immediately after the first charge fires the AIM3 altimeter records an instantaneous 115ft altitude increase, which lasts for 0.4 seconds until the data then returns to expected values that match the true flight path.

The decision was made to remove the irregularities from this spike to find the true vehicle apogee, which corresponds to the smooth vertex of the flight profile. This decision resulted in each altimeter reading almost the same value for apogee (4310 ft and 4311 ft), reinforcing the team’s belief that these corrected plots provide a much more accurate view of the flight profile. The raw, original altimeter plots can be found in the appendix, where the altitude spike is seen immediately after the first charge is fired.

Additionally, the video indicated the actual flight time was 98 seconds, these plots were bounded at 100 seconds to show only the relevant recorded data and make comparison between the two easier. Velocity versus time plots were created in MATLAB using the flight data from both altimeters and are shown in the figures below.

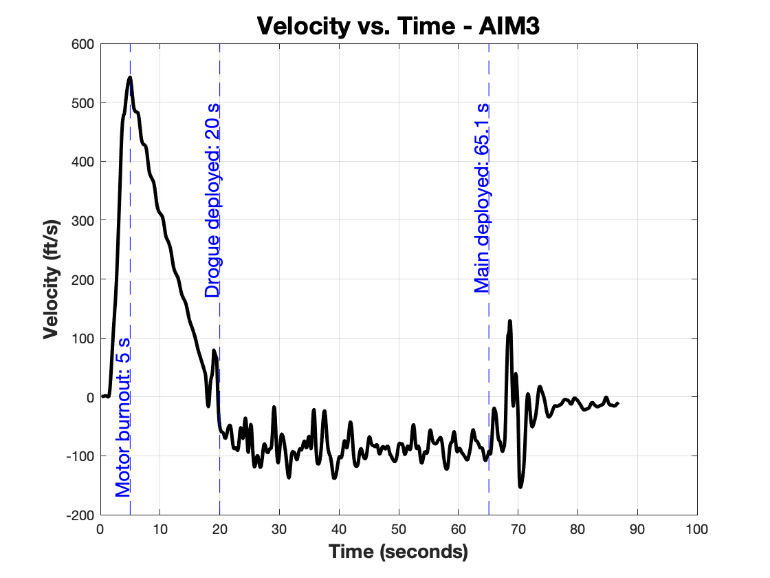
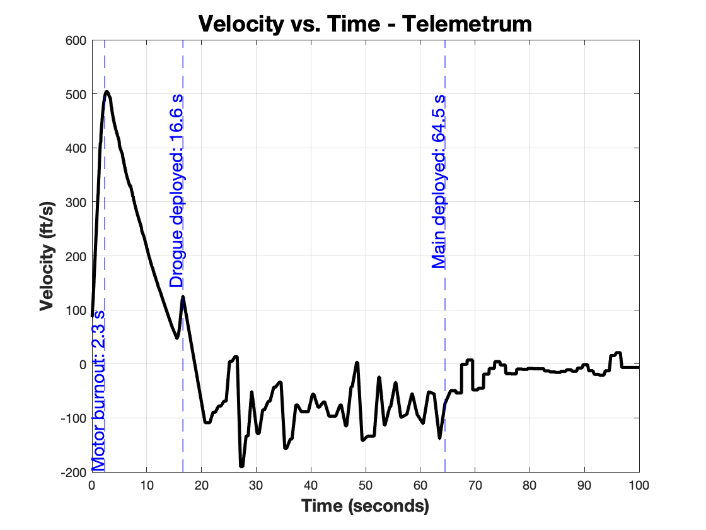


Figure 6: Velocity vs. Time Flight 3.

## As-landed configuration

The team took a wide array of photos of the final landing configuration, shown below.



Figure 7: Aft Bay Landing Configuration.

A red and silver rocket in the grass

Description automatically generated

Figure 8: Aft Bay Landing Configuration.



Figure 9: Shock Chord Extended.



Figure 10: Drogue Parachute and Avionics Bay.

A red and black object on the ground

Description automatically generated

Figure 11: Drogue Parachute and Avionics Bay.

A hand holding a red tube

Description automatically generated

Figure 12: Upper Bay.

A long red candle lying in grass

Description automatically generated

Figure 13: Upper Bay.

A red and black cone in grass

Description automatically generated

Figure 14: Nose Cone.

A yellow and black parachute on the side of a road

Description automatically generated

Figure 15: Main Parachute.

## Kinetic energy at landing

The kinetic energy at landing was calculated using the average terminal descent speed of the launch vehicle during main parachute descent, 13.3 ft/s. (TeleMetrum). This value was calculated using the data from the TeleMetrum as it was larger than that of the AIM3 (12.2 ft/s.). The resultant kinetic energies of each individual tethered section at landing are displayed in the table below.

Table 4: Launch vehicle sections, their mass and kinetic energy.

|  |  |  |
| --- | --- | --- |
| Section | Mass (lbs.) | Kinetic Energy (ft-lbf.) |
| Upper section with payload | 23.70 | 65.2 |
| Upper section | 18.66 | 51.3 |
| Avionics bay | 5.48 | 15.1 |
| Lower section | 20.87 | 57.4 |

All the above impact kinetic energies are below the established threshold of 75 ft-lbf. Thus, the Vehicle Demonstration Re-flight satisfied the impact kinetic energy requirement.

## Analysis

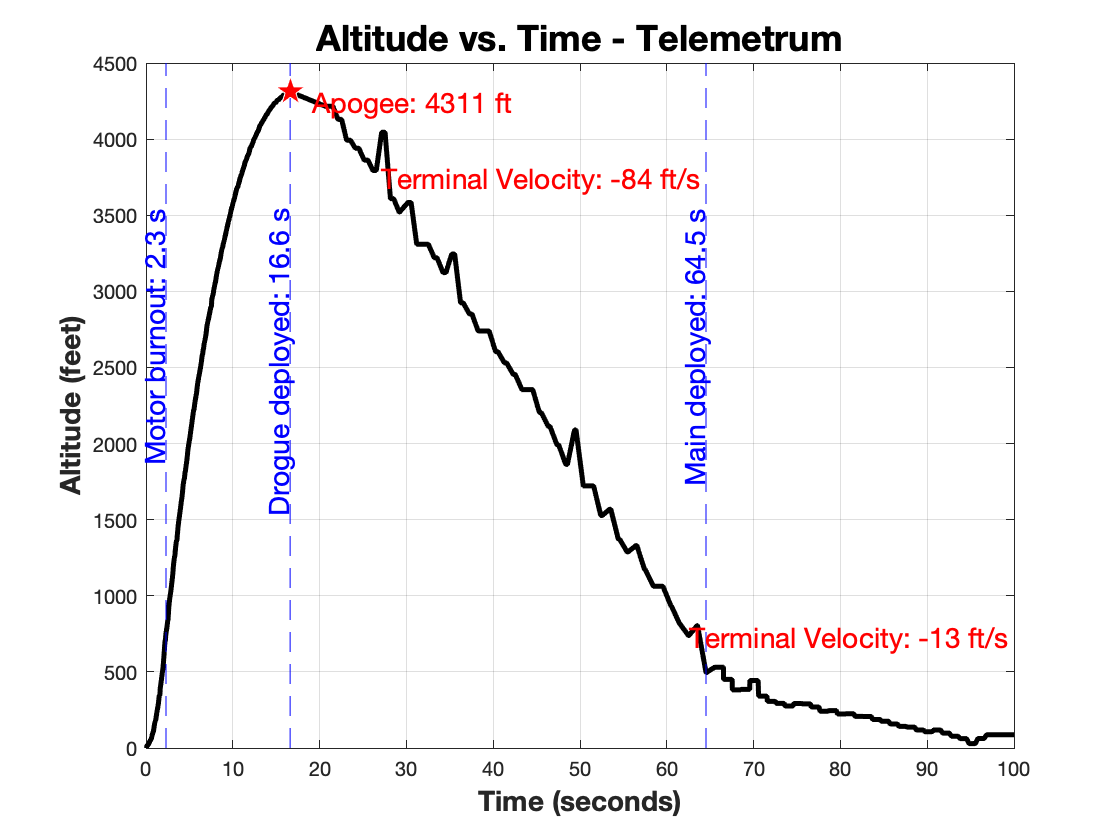


Figure 16: Flight profile from Telemetrum data.

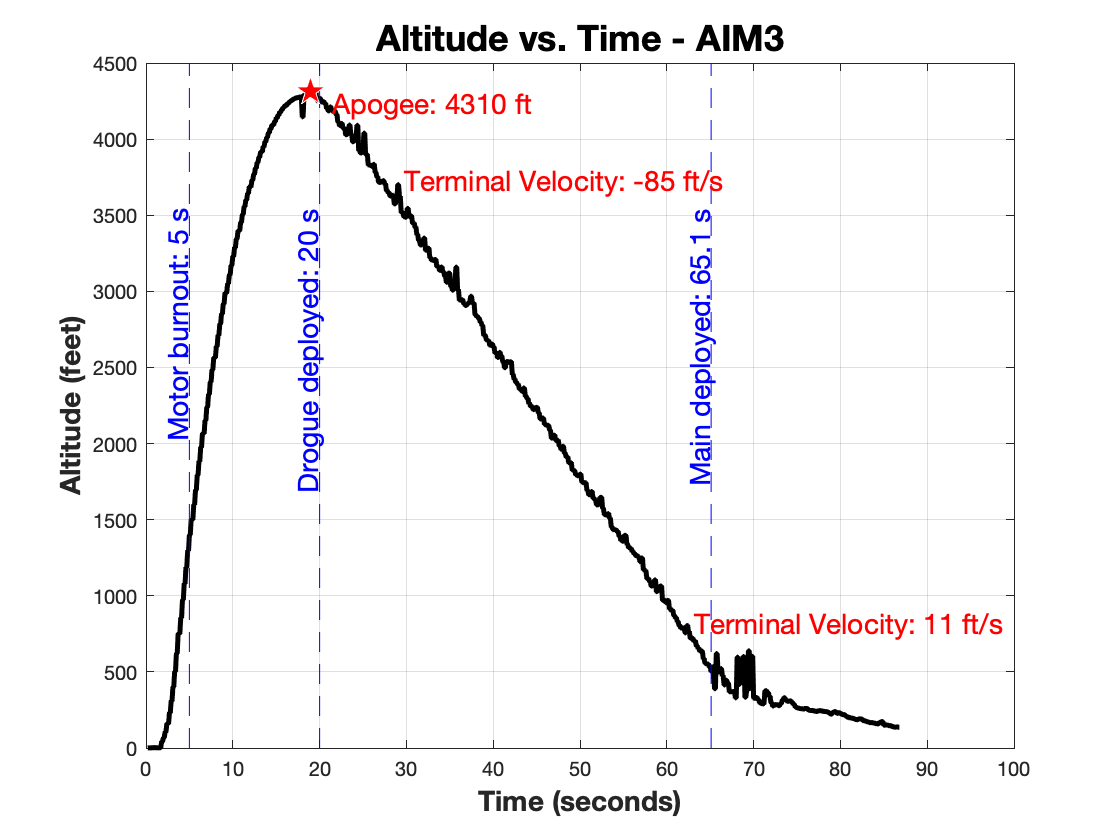


Figure 17: Flight profile from the AIM3 data.

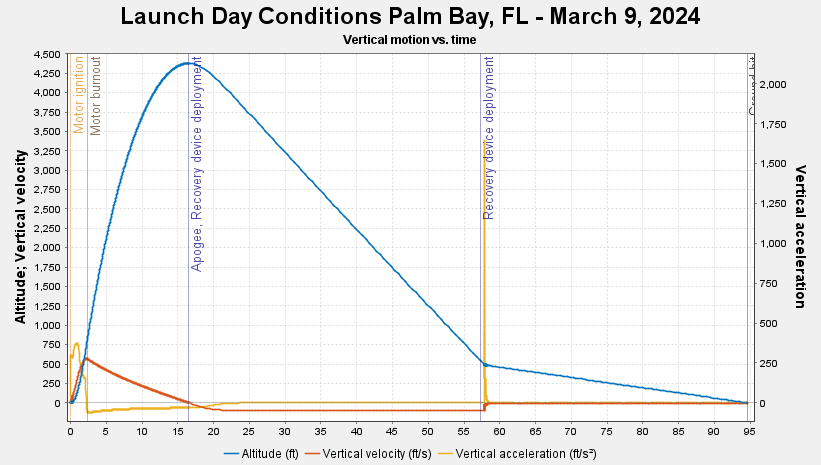
A long shot of a map

Description automatically generated

Figure 18: Google Earth flight profile.

Presented in Figures 15 and 16, is the flight profile data. Presented in Figure 17 is the data displayed on Google Earth, using a KSV file generated from the TeleMetrum. The TeleMetrum reported an apogee of 4311 ft, while the AIM3 measured 4310 ft, which is well within the NASA-allowed apogee range. The flight profile prediction with updated launch conditions simulated an apogee of 4340 ft. The terminal velocity with the drogue parachute was 85 ft/s which was slow enough for the main parachute to deploy without being compromised. The simulated and goal impact velocity was 13.3 ft/s and the actual impact velocity was roughly 12.75 ft/s between the two altimeters. The overall shapes of the velocity and acceleration are very similar to the real data when comparing the simulated data to the actual flight data. The descent time was well within the 90-second limit, at 81.1 seconds, which is completely nominal.

This flight was fully nominal. The fight apogee was within 0.68% of the predicted apogee. The broken fin from the prior flight, which was repaired using fiberglass cloth and epoxy, performed successfully. This was encouraging for the team as it reaffirmed the strength of our design.



A graph of recovery and recovery

Description automatically generated

Overall, the simulated flight closely matched the actual flight. The simulated flight only had a margin of error of 0.68 with the actual flight apogee. The drogue simulated velocity was off by 12 ft/s. The main parachute terminal velocity was slightly lower than anticipated. This was because the unpredictable wind gusts were high that day. Some sources gave readings of up to 21 mph. The ascent time was right on track with the simulation because the Open Rocket uses very current data. The descent time only had an error of 3.33, it was slightly longer which was due to the slower descent velocity of the main parachute from the increased gusts.

**Table 5: Simulated and Actual Flight Comparison**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Simulated Flight | Flight Data | % Error |
| Apogee (ft) | 4340 | 4310.5 | 0.68 |
| Drogue Terminal Velocity (ft/s) | 97 | 85 | 12.3 |
| Main Terminal Velocity (ft/s) | 13.3 | 12.75 | 4.14 |
| Ascent Time (s) | 16.2 | 16.6 | 2.41 |
| Descent Time (s) | 78.3 | 81.1 | 3.45 |

Table 5 shows the comparison between the OpenRocket simulations and the actual flight data. The apogee and terminal velocities were taken by averaging the data from both altimeters, while the precise times were found by analyzing the launch video. The table shows no significant disparity in any section, demonstrating a predictable launch and recovery.

There was a discrepancy between the expected main parachute deployment at 600 ft and the raw altimeter data, which showed the main parachute fully deployed and extended at 503 ft. A fully deployed main chute is defined as the point at which the descent rate of the vehicle slows. However, this can be explained from a video of the main deployment event, where the avionics bay is ejected directly downwards and undergoes violent motion as the parachute catches. This makes the altitude reading lower as the altimeters are on the shock cord 20 ft below the main parachute bay and moving violently, resulting in significant noise in the altitude readings that are not representative of the whole vehicle.

It is critical to note that when a Kalman filter was applied to the noisy data via the AltOS software for the TeleMetrum, the main deployment was seen to occur at 568 ft, much closer to the expected deployment altitude. The team believes this to be more accurate. The graph of the Kalman height from the TeleMetrum can be found in the appendix.

## Drag Coefficient Estimate

The predicted drag coefficient is shown below from OpenRocket software. The average is about 0.63 with a range of 0.62 to 0.66 over the duration of the flight. The actual drag of the flight was determined by using the velocity data from the altimeters and the MATLAB plot in the appendix. This code stems from a force balance after motor burnout between the drag force and the thrust.

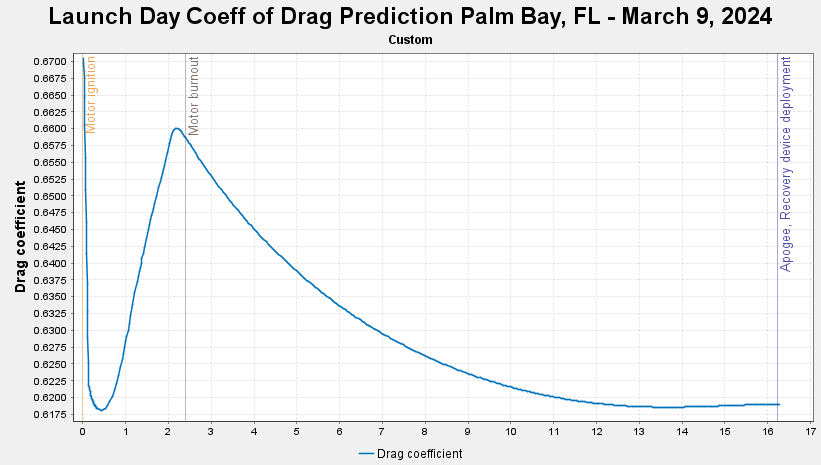


Figure 19: Predicted Drag Coefficient.

The actual coefficient of drag has an average of about 0.68 and a range from 0.67 to 0.69. This is slightly higher than the predicted coefficient of drag. This may have contributed to the apogee discrepancy from flight 3 but the apogees were so similar that the drag coefficient seemed to be within the margin of error. The nosecone finish was not accounted for properly. The nosecone was rougher than anticipated. When the real coefficient of drag was plugged into the simulations, the apogee flight simulation became 4250 ft. This is about the same amount of error as the original simulation. This further demonstrates that the coefficient of drag was within the margin of error.

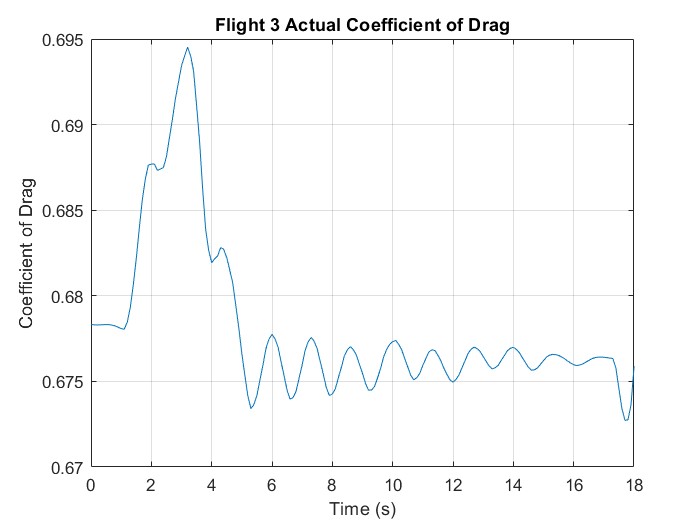


Figure 20: Actual Drag Coefficient.

## Damaged Hardware

After careful inspection of each rocket section, no hardware was found to be damaged.

## Lessons Learned

From this flight, it was concluded that each previous failure in full-scale flights 1 and 2 was indeed caused by what the team had previously concluded the causes to be, as otherwise, the flight could not have been nominal. Through this, the team learned how to (and how to not) diagnose failure and fix these errors.

This year, both testing and validation were steps that the team rushed through. The effects of this resulted in the first two full-scale flights being unsuccessful. Through this, the team learned that testing and validation is not a step to be skimmed over for next year, but rather one that requires extreme attention, planning, and evaluation.

# Appendix

## Additional Figures

A graph of a flight speed

Description automatically generated with medium confidence

**A graph of a plane

Description automatically generated**

**Kalman Filtered Height vs. Time - TeleMetrum**

**A graph with orange lines

Description automatically generated**

Figure 21: Kalman Filtered Height vs. Time - TeleMetrum.

**A graph of a graph showing a burnout

Description automatically generatedA graph of a graph showing a number of objects

Description automatically generated with medium confidence**

## MATLAB Scripts

flight3\_metrum.m

% flight3\_metrum.m

% Connor Zhou

% 2024-03-27

clc;clear;close all;

% This uses the metrum exported csv.

%% Plot Parameters

% Event times (pull from openrocket sim)

% Event text

x\_event = -1.4;

y\_event = 1300;

rot\_event = 90;

% Apogee text

x\_apogee = 2.5;

y\_apogee = -70;

% Drogue V\_terminal text

x\_drogue = -1.5;

y\_drogue = 1600;

% Main V\_terminal text

x\_main = -2;

y\_main = 200;

%% Read, Extract, Plot

% Read data from prelaunch2.csv

data = csvread('metrum\_new.csv', 1, 0); % Assuming the first row is header

% Extract data

t = data(:, 1); %s

alt\_m = data(:, 8); %m

vel\_mps = data(:, 9); %m/s

state = data(:, 4); %t/f

idx\_burnout = find(state > 4);

idx\_drogue = find(state > 5);

idx\_main = find(state > 6);

% Unit conversion

alt\_ft = alt\_m \* 3.281; %ft

vel\_ftps = vel\_mps \*3.281; %ft/s,

t\_burnout = t(idx\_burnout(1));

t\_drogue = t(idx\_drogue(1));

t\_main = t(idx\_main(1));

% Plot altitude vs. time

figure()

plot(t, alt\_ft, 'k-', 'LineWidth', 2.5);

xlabel('\bf{Time (seconds)}','FontSize', 14);

ylabel({'\bf{Altitude (feet)}'},'FontSize', 14);

title('\bf{Altitude vs. Time - Telemetrum}', 'FontSize', 18);

grid on;

xlim([0 100])

ylim([0 4500])

% Add vertical lines and annotations for flight events

line([t\_burnout, t\_burnout], ylim, 'Color', 'b', 'LineStyle', '--');

line([t\_drogue, t\_drogue], ylim, 'Color', 'b', 'LineStyle', '--');

line([t\_main, t\_main], ylim, 'Color', 'b', 'LineStyle', '--');

text(t\_burnout+x\_event, mean(ylim)+y\_event, ['Motor burnout: ', num2str(round(t\_burnout, 1)), ' s'], 'HorizontalAlignment', 'right', 'Color', 'b', 'FontSize', 14, 'Rotation', rot\_event);

text(t\_drogue+x\_event, mean(ylim)+y\_event, ['Drogue deployed: ', num2str(round(t\_drogue, 1)), ' s'], 'HorizontalAlignment', 'right', 'Color', 'b', 'FontSize', 14, 'Rotation', rot\_event);

text(t\_main+x\_event, mean(ylim)+y\_event, ['Main deployed: ', num2str(round(t\_main, 1)), ' s'], 'HorizontalAlignment', 'right', 'Color', 'b', 'FontSize', 14, 'Rotation', rot\_event);

% Calculate mean terminal velocity during each parachute descent

% Drogue parachute descent

t\_drogue\_end = t\_main; % assuming main parachute deployment marks end of drogue descent

mean\_vel\_drogue = mean(vel\_ftps(idx\_drogue));

vel\_drogue = (alt\_ft(idx\_drogue(63))-alt\_ft(idx\_drogue(161)))/(t(idx\_drogue(63))-t(idx\_drogue(161)));

% Main parachute descent

t\_main = t\_main;

t\_main\_end = t(end); % end of data

vel\_main = (alt\_ft(idx\_main(55))-alt\_ft(idx\_main(129)))/(t(idx\_main(55))-t(idx\_main(129)));

% Display mean terminal velocities

[~, max\_alt\_drogue\_idx] = max(alt\_ft(idx\_drogue));

max\_alt\_drogue = alt\_ft(idx\_drogue(max\_alt\_drogue\_idx));

center\_drogue\_alt = (max\_alt\_drogue + alt\_ft(1))/2;

text(t\_drogue + x\_drogue+0.5\*(t\_drogue\_end - t\_drogue)/2, center\_drogue\_alt+y\_drogue, ['Terminal Velocity: ', num2str(round(vel\_drogue, 0)), ' ft/s'], 'Color', 'r', 'FontSize', 14);

[~, max\_alt\_main\_idx] = max(alt\_ft(idx\_main));

max\_alt\_main = alt\_ft(idx\_main(max\_alt\_main\_idx));

text(t\_main + x\_main, max\_alt\_main + y\_main, ['Terminal Velocity: ', num2str(round(vel\_main, 0)), ' ft/s'], 'Color', 'r', 'FontSize', 14);

% Find the time of apogee

[alt\_apogee, idx\_apogee] = max(alt\_ft);

t\_apogee = t(idx\_apogee);

% Plot a filled-in red star at apogee

hold on;

apogee = plot(t\_apogee, alt\_apogee, 'rpentagram', 'MarkerSize', 15);

apogee.MarkerFaceColor = [1 0 0];

apogee.MarkerEdgeColor = [1 1 1];

text(t\_apogee + x\_apogee, alt\_apogee+y\_apogee, ['Apogee: ', num2str(round(alt\_apogee)), ' ft'], 'Color', 'r', 'FontSize', 14);

hold off

%% Velocity vs Time

figure()

plot(t, vel\_ftps, 'k-', 'LineWidth', 2.5);

xlabel('\bf{Time (seconds)}','FontSize', 14);

ylabel({'\bf{Velocity (ft/s)}'},'FontSize', 14);

title('\bf{Velocity vs. Time - Telemetrum}', 'FontSize', 18);

grid on;

xlim([0 100])

% Add vertical lines and annotations for flight events

line([t\_burnout, t\_burnout], ylim, 'Color', 'b', 'LineStyle', '--');

line([t\_drogue, t\_drogue], ylim, 'Color', 'b', 'LineStyle', '--');

line([t\_main, t\_main], ylim, 'Color', 'b', 'LineStyle', '--');

text(t\_burnout+x\_event, mean(ylim)-100, ['Motor burnout: ', num2str(round(t\_burnout, 1)), ' s'], 'HorizontalAlignment', 'right', 'Color', 'b', 'FontSize', 14, 'Rotation', rot\_event);

text(t\_drogue+x\_event, mean(ylim)+300, ['Drogue deployed: ', num2str(round(t\_drogue, 1)), ' s'], 'HorizontalAlignment', 'right', 'Color', 'b', 'FontSize', 14, 'Rotation', rot\_event);

text(t\_main+x\_event, mean(ylim)+300, ['Main deployed: ', num2str(round(t\_main, 1)), ' s'], 'HorizontalAlignment', 'right', 'Color', 'b', 'FontSize', 14, 'Rotation', rot\_event);

%% RAW DATA PLOTS

% Event text

x\_event = -2.4;

y\_event = 1000;

rot\_event = 90;

% Apogee text

x\_apogee = 2.5;

y\_apogee = -70;

% Drogue V\_terminal text

x\_drogue = 0;

y\_drogue = 1600;

% Main V\_terminal text

x\_main = 20;

y\_main = -200;

% Read data from prelaunch2.csv

data = csvread('metrum\_raw.csv', 1, 0); % Assuming the first row is header

% Extract data

t = data(:, 1); %s

alt\_m = data(:, 8); %m

vel\_mps = data(:, 9); %m/s

state = data(:, 4); %t/f

idx\_burnout = find(state > 4);

idx\_drogue = find(state > 5);

idx\_main = find(state > 6);

% Unit conversion

alt\_ft = alt\_m \* 3.281; %ft

vel\_ftps = vel\_mps \*3.281; %ft/s,

t\_burnout = t(idx\_burnout(1));

t\_drogue = t(idx\_drogue(1));

t\_main = t(idx\_main(1));

% Plot altitude vs. time

figure()

plot(t, alt\_ft, 'k-', 'LineWidth', 2.5);

xlabel('\bf{Time (seconds)}','FontSize', 14);

ylabel({'\bf{Altitude (feet)}'},'FontSize', 14);

title('\bf{Altitude vs. Time - Telemetrum}', 'FontSize', 18);

xlim([0 170])

ylim([0 5000])

grid on;

% Add vertical lines and annotations for flight events

line([t\_burnout, t\_burnout], ylim, 'Color', 'b', 'LineStyle', '--');

line([t\_drogue, t\_drogue], ylim, 'Color', 'b', 'LineStyle', '--');

line([t\_main, t\_main], ylim, 'Color', 'b', 'LineStyle', '--');

text(t\_burnout+2, mean(ylim)+y\_event+1250, ['Motor burnout: ', num2str(round(t\_burnout, 1)), ' s'], 'HorizontalAlignment', 'right', 'Color', 'b', 'FontSize', 14, 'Rotation', rot\_event);

text(t\_drogue+x\_event, mean(ylim)+y\_event, ['Drogue deployed: ', num2str(round(t\_drogue, 1)), ' s'], 'HorizontalAlignment', 'right', 'Color', 'b', 'FontSize', 14, 'Rotation', rot\_event);

text(t\_main+x\_event, mean(ylim)+y\_event, ['Main deployed: ', num2str(round(t\_main, 1)), ' s'], 'HorizontalAlignment', 'right', 'Color', 'b', 'FontSize', 14, 'Rotation', rot\_event);

% Calculate mean terminal velocity during each parachute descent

% Drogue parachute descent

t\_drogue\_end = t\_main; % assuming main parachute deployment marks end of drogue descent

mean\_vel\_drogue = mean(vel\_ftps(idx\_drogue));

vel\_drogue = (alt\_ft(idx\_drogue(63))-alt\_ft(idx\_drogue(161)))/(t(idx\_drogue(63))-t(idx\_drogue(161)));

% Main parachute descent

t\_main = t\_main;

t\_main\_end = t(end); % end of data

vel\_main = (alt\_ft(idx\_main(55))-alt\_ft(idx\_main(129)))/(t(idx\_main(55))-t(idx\_main(129)));

% Display mean terminal velocities

[~, max\_alt\_drogue\_idx] = max(alt\_ft(idx\_drogue));

max\_alt\_drogue = alt\_ft(idx\_drogue(max\_alt\_drogue\_idx));

center\_drogue\_alt = (max\_alt\_drogue + alt\_ft(1))/2;

text(t\_drogue + x\_drogue+0.5\*(t\_drogue\_end - t\_drogue)/2, center\_drogue\_alt+y\_drogue, ['Terminal Velocity: ', num2str(round(vel\_drogue, 0)), ' ft/s'], 'Color', 'r', 'FontSize', 14);

[~, max\_alt\_main\_idx] = max(alt\_ft(idx\_main));

max\_alt\_main = alt\_ft(idx\_main(max\_alt\_main\_idx));

text(t\_main + x\_main, max\_alt\_main + y\_main, ['Terminal Velocity: ', num2str(round(vel\_main, 0)), ' ft/s'], 'Color', 'r', 'FontSize', 14);

% Find the time of apogee

[alt\_apogee, idx\_apogee] = max(alt\_ft);

t\_apogee = t(idx\_apogee);

hold off

% VELOCITY VS TIME

figure()

plot(t, vel\_ftps, 'k-', 'LineWidth', 2.5);

xlabel('\bf{Time (seconds)}','FontSize', 14);

ylabel({'\bf{Velocity (ft/s)}'},'FontSize', 14);

title('\bf{Velocity vs. Time - Telemetrum}', 'FontSize', 18);

xlim([0 170])

grid on;

% Add vertical lines and annotations for flight events

line([t\_burnout, t\_burnout], ylim, 'Color', 'b', 'LineStyle', '--');

line([t\_drogue, t\_drogue], ylim, 'Color', 'b', 'LineStyle', '--');

line([t\_main, t\_main], ylim, 'Color', 'b', 'LineStyle', '--');

text(t\_burnout+2, mean(ylim)-100, ['Motor burnout: ', num2str(round(t\_burnout, 1)), ' s'], 'HorizontalAlignment', 'right', 'Color', 'b', 'FontSize', 14, 'Rotation', rot\_event);

text(t\_drogue+x\_event, mean(ylim)+300, ['Drogue deployed: ', num2str(round(t\_drogue, 1)), ' s'], 'HorizontalAlignment', 'right', 'Color', 'b', 'FontSize', 14, 'Rotation', rot\_event);

text(t\_main+x\_event, mean(ylim)+300, ['Main deployed: ', num2str(round(t\_main, 1)), ' s'], 'HorizontalAlignment', 'right', 'Color', 'b', 'FontSize', 14, 'Rotation', rot\_event);

flight3\_aim3.m

% flight3\_aim3.m

% Connor Zhou

% 2024-03-18

clc;clear;close all;

% This uses the aim3 exported csv.

%% Plot Parameters

% Event times (pull from openrocket sim)

% Event text

x\_event = -1.4;

y\_event = 1300;

rot\_event = 90;

% Apogee text

x\_apogee = 2.5;

y\_apogee = -70;

% Drogue V\_terminal text

x\_drogue = -1.5;

y\_drogue = 1600;

% Main V\_terminal text

x\_main = -2;

y\_main = 200;

%% Read, Extract, Plot

% Read data from prelaunch2.csv

data = csvread('newaim3.csv', 1, 0); % Assuming the first row is header

% Extract data

t = data(:, 1); %s

alt\_m = data(:, 5); %m

vel\_kmph = data(:, 6); %km/h

lineA\_on = data(:, 7); %t/f

lineB\_on = data(:, 8); %t/f

idx\_apogee = find(lineA\_on > 0);

idx\_main = find(lineB\_on > 0);

% Unit conversion

alt\_ft = alt\_m \* 3.281; %ft

vel\_ftps = vel\_kmph / 1.097; %ft/s

% Event time retrieval

% Calculate acceleration (rate of change of velocity)

acceleration = diff(vel\_ftps) ./ diff(t);

% Find the index when altitude first exceeds 50 feet

idx\_alt\_above\_50 = find(alt\_ft > 50, 1);

% Find the index where acceleration becomes negative

idx\_negative\_acceleration = find(acceleration(idx\_alt\_above\_50:end) < 0, 1);

% Calculate the corresponding index in the original data

if ~isempty(idx\_negative\_acceleration)

idx\_negative\_acceleration = idx\_alt\_above\_50 + idx\_negative\_acceleration - 1;

% Find the time at which acceleration becomes negative

t\_burnout = t(idx\_negative\_acceleration);

else

disp('Acceleration does not become negative above 50 feet.');

end

t\_drogue = t(idx\_apogee(1));

t\_drogue\_end = t(idx\_main(1));

t\_main = t\_drogue\_end;

t\_main\_end = t(end);

% Plot altitude vs. time

figure()

plot(t, alt\_ft, 'k-', 'LineWidth', 2.5);

xlabel('\bf{Time (seconds)}','FontSize', 14);

ylabel({'\bf{Altitude (feet)}'},'FontSize', 14);

title('\bf{Altitude vs. Time - AIM3}', 'FontSize', 18);

grid on;

xlim([0 100])

ylim([0 4500])

% Add vertical lines and annotations for flight events

line([t\_burnout, t\_burnout], ylim, 'Color', 'b', 'LineStyle', '--');

line([t\_drogue, t\_drogue], ylim, 'Color', 'b', 'LineStyle', '--');

line([t\_main, t\_main], ylim, 'Color', 'b', 'LineStyle', '--');

text(t\_burnout+x\_event, mean(ylim)+y\_event, ['Motor burnout: ', num2str(round(t\_burnout, 1)), ' s'], 'HorizontalAlignment', 'right', 'Color', 'b', 'FontSize', 14, 'Rotation', rot\_event);

text(t\_drogue+x\_event, mean(ylim)+y\_event, ['Drogue deployed: ', num2str(round(t\_drogue, 1)), ' s'], 'HorizontalAlignment', 'right', 'Color', 'b', 'FontSize', 14, 'Rotation', rot\_event);

text(t\_main+x\_event, mean(ylim)+y\_event, ['Main deployed: ', num2str(round(t\_main, 1)), ' s'], 'HorizontalAlignment', 'right', 'Color', 'b', 'FontSize', 14, 'Rotation', rot\_event);

% Calculate mean terminal velocity during each parachute descent

% Drogue parachute descent

t\_drogue\_end = t\_main; % assuming main parachute deployment marks end of drogue descent

mean\_vel\_drogue = mean(vel\_ftps(idx\_apogee));

vel\_drogue = (alt\_ft(idx\_apogee(end))-alt\_ft(idx\_main(1)))/(t(idx\_apogee(end))-t(idx\_main(1)));

% Main parachute descent

t\_main = t\_main;

t\_main\_end = t(end); % end of data

vel\_main = (alt\_ft(idx\_main(10))-alt\_ft(end))/(t(idx\_main(10))-t(end));

vel\_main = 11.2975;

% Display mean terminal velocities

[~, max\_alt\_drogue\_idx] = max(alt\_ft(idx\_apogee));

max\_alt\_drogue = alt\_ft(idx\_apogee(max\_alt\_drogue\_idx));

center\_drogue\_alt = (max\_alt\_drogue + alt\_ft(1))/2;

text(t\_drogue + x\_drogue+0.5\*(t\_drogue\_end - t\_drogue)/2, center\_drogue\_alt+y\_drogue, ['Terminal Velocity: ', num2str(round(vel\_drogue, 0)), ' ft/s'], 'Color', 'r', 'FontSize', 14);

[~, max\_alt\_main\_idx] = max(alt\_ft(idx\_main));

max\_alt\_main = alt\_ft(idx\_main(max\_alt\_main\_idx));

text(t\_main + x\_main, max\_alt\_main + y\_main, ['Terminal Velocity: ', num2str(round(vel\_main, 0)), ' ft/s'], 'Color', 'r', 'FontSize', 14);

% Find the time of apogee

[alt\_apogee, idx\_apogee] = max(alt\_ft);

t\_apogee = t(idx\_apogee);

% Plot a filled-in red star at apogee

hold on;

apogee = plot(t\_apogee, alt\_apogee, 'rpentagram', 'MarkerSize', 15);

apogee.MarkerFaceColor = [1 0 0];

apogee.MarkerEdgeColor = [1 1 1];

text(t\_apogee + x\_apogee, alt\_apogee+y\_apogee, ['Apogee: ', num2str(round(alt\_apogee)), ' ft'], 'Color', 'r', 'FontSize', 14);

hold off

%% Velocity vs Time

figure()

plot(t, vel\_ftps, 'k-', 'LineWidth', 2.5);

xlabel('\bf{Time (seconds)}','FontSize', 14);

ylabel({'\bf{Velocity (ft/s)}'},'FontSize', 14);

title('\bf{Velocity vs. Time - AIM3}', 'FontSize', 18);

grid on;

xlim([0 100])

% Add vertical lines and annotations for flight events

line([t\_burnout, t\_burnout], ylim, 'Color', 'b', 'LineStyle', '--');

line([t\_drogue, t\_drogue], ylim, 'Color', 'b', 'LineStyle', '--');

line([t\_main, t\_main], ylim, 'Color', 'b', 'LineStyle', '--');

text(t\_burnout+x\_event, mean(ylim)-100, ['Motor burnout: ', num2str(round(t\_burnout, 1)), ' s'], 'HorizontalAlignment', 'right', 'Color', 'b', 'FontSize', 14, 'Rotation', rot\_event);

text(t\_drogue+x\_event, mean(ylim)+300, ['Drogue deployed: ', num2str(round(t\_drogue, 1)), ' s'], 'HorizontalAlignment', 'right', 'Color', 'b', 'FontSize', 14, 'Rotation', rot\_event);

text(t\_main+x\_event, mean(ylim)+300, ['Main deployed: ', num2str(round(t\_main, 1)), ' s'], 'HorizontalAlignment', 'right', 'Color', 'b', 'FontSize', 14, 'Rotation', rot\_event);

%% RAW DATA PLOTS

% Read data from prelaunch2.csv

data = csvread('aim3.csv', 1, 0); % Assuming the first row is header

% Extract data

t = data(:, 1); %s

alt\_m = data(:, 5); %m

vel\_kmph = data(:, 6); %km/h

lineA\_on = data(:, 7); %t/f

lineB\_on = data(:, 8); %t/f

idx\_apogee = find(lineA\_on > 0);

idx\_main = find(lineB\_on > 0);

% Unit conversion

alt\_ft = alt\_m \* 3.281; %ft

vel\_ftps = vel\_kmph / 1.097; %ft/s

% Event time retrieval

% Calculate acceleration (rate of change of velocity)

acceleration = diff(vel\_ftps) ./ diff(t);

% Find the index when altitude first exceeds 50 feet

idx\_alt\_above\_50 = find(alt\_ft > 50, 1);

% Find the index where acceleration becomes negative

idx\_negative\_acceleration = find(acceleration(idx\_alt\_above\_50:end) < 0, 1);

% Calculate the corresponding index in the original data

if ~isempty(idx\_negative\_acceleration)

idx\_negative\_acceleration = idx\_alt\_above\_50 + idx\_negative\_acceleration - 1;

% Find the time at which acceleration becomes negative

t\_burnout = t(idx\_negative\_acceleration);

else

disp('Acceleration does not become negative above 50 feet.');

end

t\_drogue = t(idx\_apogee(1));

t\_drogue\_end = t(idx\_main(1));

t\_main = t\_drogue\_end;

t\_main\_end = t(end);

% Plot altitude vs. time

figure()

plot(t, alt\_ft, 'k-', 'LineWidth', 2.5);

xlabel('\bf{Time (seconds)}','FontSize', 14);

ylabel({'\bf{Altitude (feet)}'},'FontSize', 14);

title('\bf{Altitude vs. Time - AIM3}', 'FontSize', 18);

grid on;

% Add vertical lines and annotations for flight events

line([t\_burnout, t\_burnout], ylim, 'Color', 'b', 'LineStyle', '--');

line([t\_drogue, t\_drogue], ylim, 'Color', 'b', 'LineStyle', '--');

line([t\_main, t\_main], ylim, 'Color', 'b', 'LineStyle', '--');

text(t\_burnout+x\_event, mean(ylim)+y\_event, ['Motor burnout: ', num2str(round(t\_burnout, 1)), ' s'], 'HorizontalAlignment', 'right', 'Color', 'b', 'FontSize', 14, 'Rotation', rot\_event);

text(t\_drogue+x\_event, mean(ylim)+y\_event, ['Drogue deployed: ', num2str(round(t\_drogue, 1)), ' s'], 'HorizontalAlignment', 'right', 'Color', 'b', 'FontSize', 14, 'Rotation', rot\_event);

text(t\_main+x\_event, mean(ylim)+y\_event, ['Main deployed: ', num2str(round(t\_main, 1)), ' s'], 'HorizontalAlignment', 'right', 'Color', 'b', 'FontSize', 14, 'Rotation', rot\_event);

% Calculate mean terminal velocity during each parachute descent

% Drogue parachute descent

t\_drogue\_end = t\_main; % assuming main parachute deployment marks end of drogue descent

mean\_vel\_drogue = mean(vel\_ftps(idx\_apogee));

vel\_drogue = (alt\_ft(idx\_apogee(end))-alt\_ft(idx\_main(1)))/(t(idx\_apogee(end))-t(idx\_main(1)));

% Main parachute descent

t\_main = t\_main;

t\_main\_end = t(end); % end of data

vel\_main = (alt\_ft(idx\_main(10))-alt\_ft(end))/(t(idx\_main(10))-t(end));

vel\_main = 11.2975;

% Display mean terminal velocities

[~, max\_alt\_drogue\_idx] = max(alt\_ft(idx\_apogee));

max\_alt\_drogue = alt\_ft(idx\_apogee(max\_alt\_drogue\_idx));

center\_drogue\_alt = (max\_alt\_drogue + alt\_ft(1))/2;

text(t\_drogue + x\_drogue+0.5\*(t\_drogue\_end - t\_drogue)/2, center\_drogue\_alt+y\_drogue, ['Terminal Velocity: ', num2str(round(vel\_drogue, 0)), ' ft/s'], 'Color', 'r', 'FontSize', 14);

[~, max\_alt\_main\_idx] = max(alt\_ft(idx\_main));

max\_alt\_main = alt\_ft(idx\_main(max\_alt\_main\_idx));

text(t\_main + x\_main, max\_alt\_main + y\_main, ['Terminal Velocity: ', num2str(round(vel\_main, 0)), ' ft/s'], 'Color', 'r', 'FontSize', 14);

% Find the time of apogee

[alt\_apogee, idx\_apogee] = max(alt\_ft);

t\_apogee = t(idx\_apogee);

hold off

% VELOCITY VS TIME

figure()

plot(t, vel\_ftps, 'k-', 'LineWidth', 2.5);

xlabel('\bf{Time (seconds)}','FontSize', 14);

ylabel({'\bf{Velocity (ft/s)}'},'FontSize', 14);

title('\bf{Velocity vs. Time - AIM3}', 'FontSize', 18);

grid on;

% Add vertical lines and annotations for flight events

line([t\_burnout, t\_burnout], ylim, 'Color', 'b', 'LineStyle', '--');

line([t\_drogue, t\_drogue], ylim, 'Color', 'b', 'LineStyle', '--');

line([t\_main, t\_main], ylim, 'Color', 'b', 'LineStyle', '--');

text(t\_burnout+x\_event, mean(ylim)-100, ['Motor burnout: ', num2str(round(t\_burnout, 1)), ' s'], 'HorizontalAlignment', 'right', 'Color', 'b', 'FontSize', 14, 'Rotation', rot\_event);

text(t\_drogue+x\_event, mean(ylim)+300, ['Drogue deployed: ', num2str(round(t\_drogue, 1)), ' s'], 'HorizontalAlignment', 'right', 'Color', 'b', 'FontSize', 14, 'Rotation', rot\_event);

text(t\_main+x\_event, mean(ylim)+300, ['Main deployed: ', num2str(round(t\_main, 1)), ' s'], 'HorizontalAlignment', 'right', 'Color', 'b', 'FontSize', 14, 'Rotation', rot\_event);