JB3-2102

Design, Analysis, and Test of a High-Powered Model Rocket-2

A Major Qualifying Project Report
Submitted to the Faculty of the
WORCESTER POLYTECHNIC INSTITUTE
in Partial Fulfillment of the Requirements for the
Degree of Bachelor of Science
in Aerospace Engineering

by

Julia Bigwood | Robert Connor | Colby Gilbert

Caroline Kuhnle | Alec Mitkov | Nathaniel Rutkowski

Christian M. Schrader | James Ternent

April 13th, 2021

Approved by:

John J. Blandino
Professor, Aerospace Engineering Department
Worcester Polytechnic Institute

Abstract

This paper describes the design, analysis, assembly, and test of a high-powered model rocket designed to use an actively controlled set of grid fins for stabilization, and a cold gas thruster for its terminal descent. The grid fins were modeled using SOLIDWORKS and 3D printed for subsystem level testing using an Arduino microcontroller. A mechanical separation system for the second stage was designed using linear actuators and 3D printed clasps. Aerodynamic loads on the vehicle airframe, grid fins, and stabilizing fins were evaluated using computational fluid dynamic (CFD) tools in Ansys Fluent. Results from the CFD analysis were used as inputs to a dynamical simulation of the vehicle trajectory and attitude, implemented in MATLAB. Ansys Workbench was used for structural analysis. An analysis of the composite motor was completed using Cantera and COMSOL to model the chemical equilibrium reaction and evaluate the temperature distribution in the motor during flight. These results were used to provide chamber conditions in a MATLAB model for ideal rocket performance. Finally, components for the cold gas descent propulsion system were identified and the required thrust and on-time for various landing velocities evaluated. Results are presented from these analyses as well as a description of prototype construction and testing completed at the subsystem level.

Acknowledgements

The MQP team wants to recognize and thank the individuals and groups listed below for their support and help over the course of the project.

- Project Advisor Professor Blandino for his guidance and support over the course of the project
- Professor Hera for training and helping the team with Ansys and Fluent
- Alicea Hyland for helping and confirming the Fluent model for ascent was correct
- Dr. Erica Stults and the WPI Rapid Prototyping lab for helping with 3D printing and design advice

Table of Contents

Table of Figures

Table of Authorship

	Abstract	ALL
	Acknowledgements	Caroline Kuhnle
1	**Introduction**	
1.1	Literature Review	ALL
1.1.1	Airframe and Recovery System	ALL ARS
1.1.2	Propulsion, Thermal, & Separation Systems	ALL PTSS
1.1.3	Flight Dynamics Analysis	ALL FDA
1.2	Overall Project Goals	All
1.3	Project Design Requirements, Constraints, and other Considerations.	All
1.4	Tasks and Timetable	Caroline Kuhnle
2	**Airframe and Recovery Systems**	ALL ARS
2.1	Methodology	Colby Gilbert
2.1.1	Airframe	Robert Connor &Alec Mitkov
2.1.2	Grid Fins	ALL ARS
2.1.3	Flight Camera	Robert Connor
2.1.4	Nosecone	Alec Mitkov
2.2	Analysis	Colby Gilbert
2.2.1	Finite Element Analysis (FEA) with Ansys Airframe Stress Distribution	Colby Gilbert
2.2.2	Computational Fluid Dynamics (CFD) Analysis with Ansys	Alec Mitkov
2.2.3	Airframe Stress Distribution (Analysis Task 1)	Alec Mitkov
2.2.4	Grid Fin Aerodynamic Loads (ARS Analysis Task 2)	Alec Mitkov
2.2.5	Grid Fin Stress Distribution (Analysis Task 3	Alec Mitkov & Colby Gilbert
3	**Propulsion, Thermal, and Separation Systems**	
3.1	Methodology	
3.1.1	Engine Systems	Nathaniel Rutkowski
3.1.2	Ignition Systems	Nathaniel Rutkowski
3.1.3	Motor Performance Analysis (Analysis Task 1)	Nathaniel Rutkowski
3.1.4	Thermal Distribution Model (Analysis Task 2)	Nathaniel Rutkowski
3.2	Separation	Julia Bigwood
3.2.1	Black powder Charges	Julia Bigwood
3.2.2	Mechanical Clasp Separation	Julia Bigwood
3.2.3	Cold Gas Thruster	James Ternent
3.2	Analysis	
3.2.1	Motor Performance Analysis (Analysis Task 1) Results	Nathaniel Rutkowski

3.2.2	Thermal Distribution Model (Analysis Task 2) Results	Nathaniel Rutkowski
3.2.3	Analysis Task 3 Results	Julia Bigwood
3.2.4	Analysis Task 4 Results	
4	**Flight Dynamics Analysis**	Caroline Kuhnle
4.1	Methodology	ALL FDA
4.1.1	Stability Fin Design	Caroline Kuhnle
4.1.2	Avionics Implementation	Caroline Kuhnle
4.1.3	Avionics Bay	Christian M. Schrader
4.2	Analysis	ALL FDA
4.2.1	Analysis Task 1 (Vehicle Dynamics and Performance Model)	Christian M. Schrader
4.2.2	Analysis Task 2 (Vehicle Aerodynamic Loads Simulation)	Caroline Kuhnle
5	**Summary, Conclusions, Recommendations, Broader Impacts**	ALL ARS
5.1	Airframe and Recovery System	Colby Gilbert
5.1.1	Analysis task 1	Colby Gilbert
5.1.2	Analysis task 2	Colby Gilbert
5.1.3	Analysis task 3	Colby Gilbert
5.2	PTSS	
5.2.1	Motor Performance Analysis (Analysis Task 1)	Nathaniel Rutkowski
5.2.2	Thermal Distribution Model (Analysis Task 2)	Nathaniel Rutkowski
5.2.3	Mechanical Separation System Model (Analysis Task 3)	Julia Bigwood
5.2.4	Cold Gas Thruster Descent (Analysis Task 4)	James Ternent
5.3	Flight Dynamics Analysis	
5.3.1	Analysis task 1	Christian M. Schrader
5.3.2	Analysis task 2	Caroline Kuhnle
5.4	Broader Impact	Christian M. Schrader, Julia Bigwood,Nathanial Rutkowski

1 Introduction

Project Andromeda is an innovative high-powered model rocket developed as our Major Qualifying Project (MQP) for the Aerospace Engineering Degree completion at Worcester Polytechnic Institute (WPI). Project Andromeda employs an array of innovative subsystems including grid fins, active stabilization, mechanical stage separation, and propulsive landing. A mission profile of the proposed flight in described in

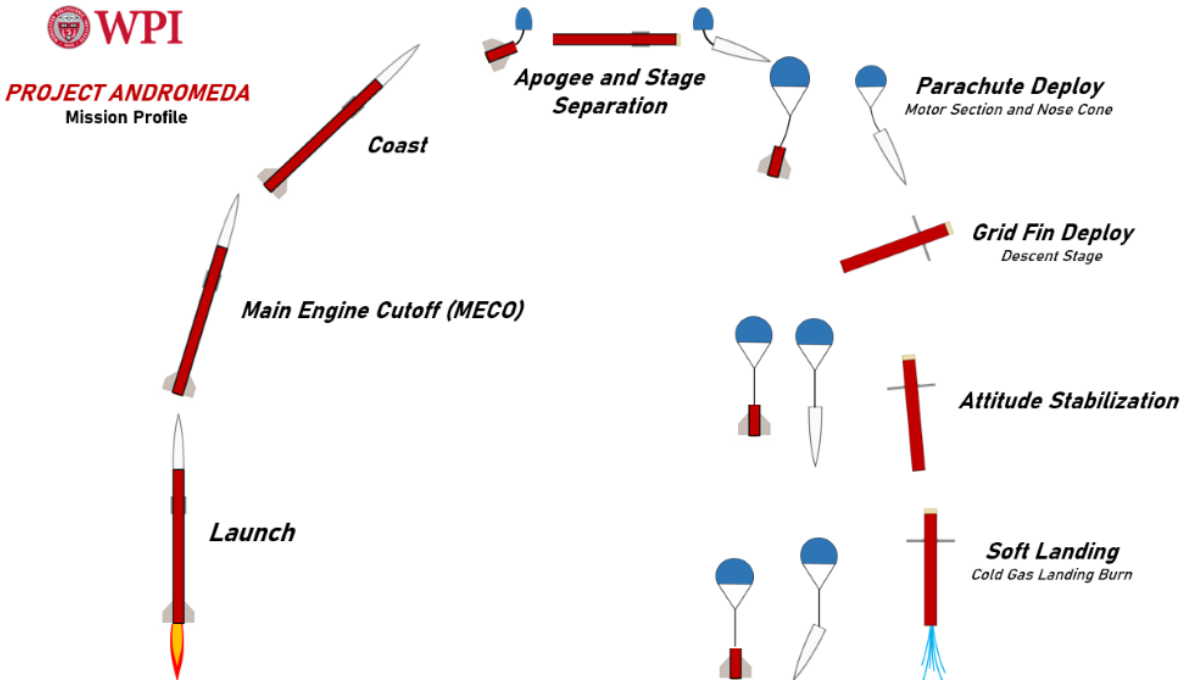


Figure 1-1.

Figure 1-1 Project Andromeda Mission Profile

The MQP team was organized into three individual subteams; Airframe and Recovery System (ARS), Propulsion, Thermal, and Stage Separation (PTSS), and Flight Dynamics Analysis (FDA). Each subteam is responsible for their respective innovative subsystems described later on in this section.

1.1 Literature Review

A literature review was conducted to learn about methods used in High-Powered Model Rocketry. The team focused on different systems including airframe, recovery system, propulsion system, separation system, and flight dynamics.

1.1.1 Airframe and Recovery System

1.1.1.1 Airframe of a Model Rocket

The airframe is the main structure of the rocket and serves as the framework for all of its subsystems. It encloses the avionics bay, the motor bay, the cold gas thruster bay, the recovery system, any payloads, and the separation systems. An example of a basic model rocket can be seen in Figure 1-2, with major components and general locations shown. The airframes of rockets are "generally smooth thin-walled cylinders with a high length to diameter ratio" [1]. The fins and motor mount are located at the aft end of rocket, with the electronics, payload and recovery systems located close to the center of the airframe. A space for the nosecone is located at the forward end of the airframe.

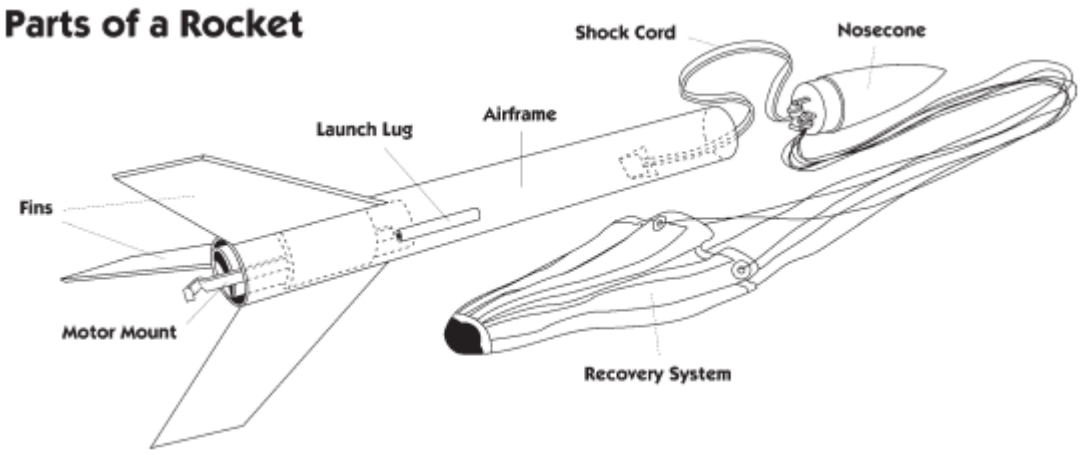


Figure 1-2 Parts of a Rocket [2] © National Association of Rocketry 2020

For a high-powered rocket, the airframes are "typically made of non-metallic, high strength-to-weight ratio composite materials like carbon fiber, fiberglass, phenolic[1] and PVC" [1]. High powered rockets are usually defined as any rocket that exceeds 1500 grams including the

[1] Phenolic is a thermosetting resin or plastic made by condensation of phenol [121]

weight of the motor(s). Furthermore, motors are broken into ‘classes’ and levels. Level I motors consist of H and I impulse class motors, while level II motors consist of J,K, and L impulse class motors. [3] In the case of a model rocket, the airframes are usually made of spiral wound paper [4]. A lightweight material is important and must be able to withstand the force generated by the motor and protect the subsystems during flight and recovery. The National Association of Rocketry (NAR) Safety Code recommends only using metal if necessary, in the construction of high-powered rockets, to limit the danger of a launch or in-flight accident [5]. Another common choice for airframe material is Blue Tube 2.0 from Always Ready Rocketry. At 1.25 g/cm^3, Blue Tube 2.0 is 28% more dense than phenolic tubing, but 36% lighter than fiberglass [6]. Blue Tube 2.0 is a common airframe material for model rockets, due to its high strength and low weight.

Figure 1-3 Blue Tube 2.0 [5] © Always Ready Rocketry 2020

A body-fixed coordinate system can be represented selected to easily reference the axes of a rocket. The body-fixed coordinate system is referenced to the airframe at launch and is used throughout the design and construction of a rocket. The side and top views of rocket, showing such a coordinate system can be seen in Figure 1-4. The origin is placed at the aft end of the rocket with the motor bay and fins, centered in the middle of the airframe cylinder. The X, Y, and Z axis

directions were chosen to line up with the roll, pitch, and yaw respectively, as well as keep consistent with industry standards [7].

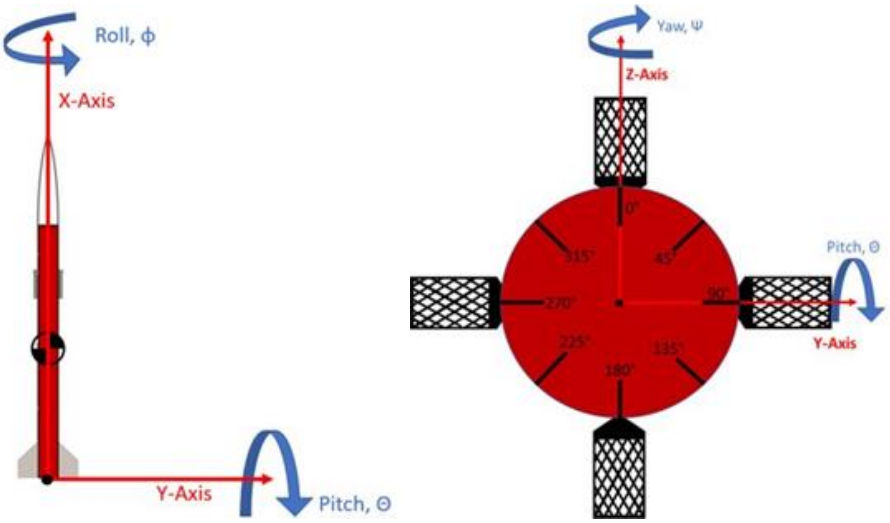


Figure 1-4 Rocket Coordinate System – Side View and Top View

1.1.1.2 Avionics Bay

The purpose of the avionics bay is to house all avionics on board the model rocket. The purpose of the avionics bay is to protect the avionics equipment on board so that they will be intact after landing. In a rocket built as part of the 2018-2019 MQP [8], the avionics bay held the battery, altimeter, Inertial Measurement Unit (IMU), and all other control systems which is what our avionics bay will also hold.

Figure 1-5 Example of an Avionics Bay [8]

Figure 1-5 is the completed design for the e-bay of a rocket built as part of the 2018-2019 MQP [8]. The bulkheads used were made of plywood with 0.5 in thickness to increase strength. Our group also used a coupler tube in order allowing the e-bay to also serve as a connection between the motor bay and the payload bay. [8]

An MIT rocketry team housed the avionics bay in the avionics tube (which is similar to an e-bay), which was constructed with a 12 in. long coupler tube. The tube was attached to the rocket by bulkheads and provided a place for the recovery system to attach [9]. Our group plans to use bulkheads as well, to both hold the bay in place and to attach other areas of the rocket.

Figure 7 is a cross section of an avionics bay used in many Level II model rockets, it is sold by SMT designs and is commercially available to be used in high powered model rockets. As can be seen, it houses all electronics within the rocket.

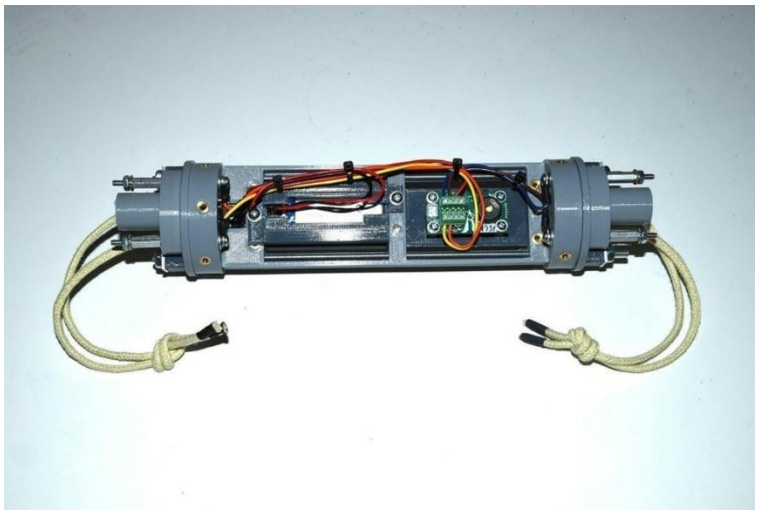


Figure 1-6 Avionics Bay ©SMT Designs 2020

Electronic bays such as the one shown in Figure 1-6 are readily available for purchase. However, our group planned on construction our own in order to optimize the avionics bay for our purposes.

1.1.1.3 Motor Bay

Figure 1-7 shows the engine of a model rocket is located at its aft. The goal of the motor bay is to house the motor and to provide a point of support for the fins. Since our group chose to use a Level II motor this will determine the size of the bay. If a motor is Level II, it means that it requires a level II certification to launch. This certification encompasses high-powered motors with an impulse up to 5120 N-sec [10].

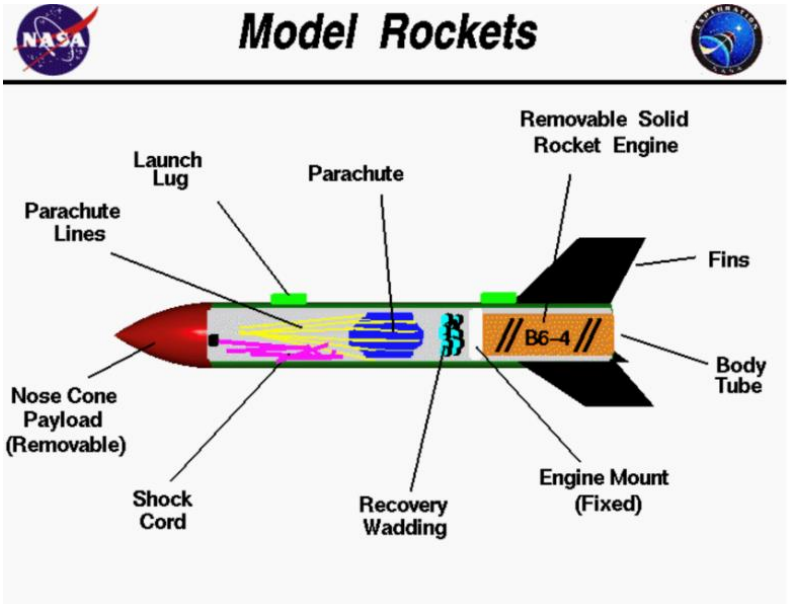


Figure 1-7 Model Rocket Cross Section © NASA 2014

The motor bay of the high-powered model rocket from the STAR program at UCAL Berkley (in Figure 1-8) had three main parts to their motor bay: a motor mount, and engine block, and three centering rings. The purpose of these is to hold the motor in place to ensure that the structural integrity of the rocket is not compromised during ascent. For the motor mount, the team chose to use kraft phenolic due to its high heat resistance. The rings were placed 9” apart from each other and were made of plywood to keep the motor in place. [11]

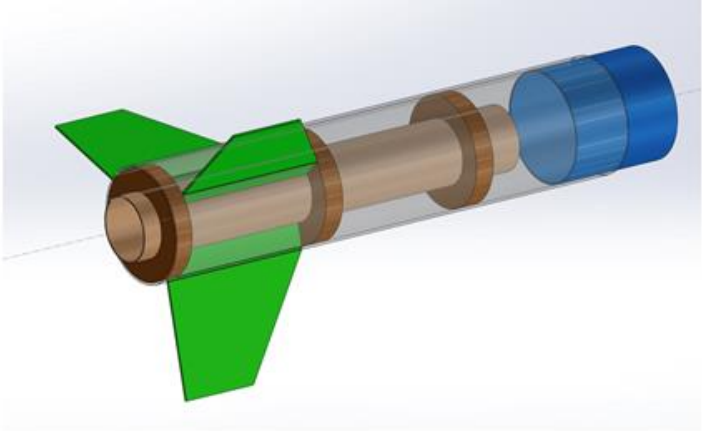


Figure 1-8 Booster Stage Interior © Star Program 2017

1.1.1.4 Nosecone

The nosecone of a rocket is the surface that encounters the freestream velocity of air on the ascent stage of a rocket, and therefore has a large effect on the performance of the rocket depending on the drag acting on the nosecone.

The nosecone is located at the top, or forward, portion of the airframe, attached to the body of the rocket itself. The nosecone fits into the cylindrical body of the rocket by what is referred to as the neck of the nosecone. The neck fits tightly inside the airframe but fits loosely enough that it can be successfully ejected during recovery. Despite a slightly smaller neck diameter, the diameter of the nosecone itself must match the diameter of the rocket body to prevent additional drag forces. The nosecone cannot be constructed with metal in order to comply with NAR standards to prevent property damage or injury upon impact [12].

Nosecones come in a variety of shapes and sizes. This is illustrated in Figure 1-9, which shows the corresponding drag force that each nosecone design will produce for model rockets with a 24 mm diameter [13]. Since the nosecone is located at the top of the rocket, its shape and aerodynamic design have the largest impact on the drag forces that the rocket will encounter [13]. At apogee, the drag force acting on the nose cone will be at a minimum; therefore, the nosecone is ejected at apogee to minimize the drag that the rocket experiences and maximize the altitude of the apogee.

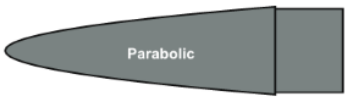
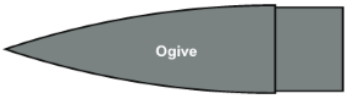
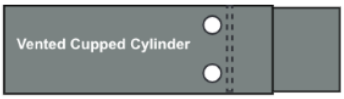
Nose Shape	Wind Speed	Temp	Drag Force
	39.28 mph	72.0° F	4.477 g
	39.28 mph	72.0° F	4.942 g
	39.27 mph	72.0° F	4.149 g
	39.27 mph	72.0° F	4.791 g
	39.26 mph	72.5° F	4.561 g
	39.25 mph	72.0° F	5.248 g
	39.24 mph	72.0° F	8.659 g
	39.26 mph	72.0° F	10.459 g
	39.19 mph	72.5° F	10.399 g

Figure 1-9 Drag on Nosecones [13] © Apogee Components 2013

Model rocketry nosecones are typically ejected using a black powder charge at apogee. The magnitude of the force required to successfully eject the nosecone with a black powder charge is expressed in 1. The force needed to eject the nosecone is dependent on the shear pins and the forces required to break them. In Eq. 1, A is the cross-sectional area of the pin, and τ is the shear strength [9].

$$F = A\,\tau \qquad 1$$

Making a 3D model of the optimal nosecone shape with the minimal drag, the long elliptical shape, and 3D printing it would be an easy strategy for obtaining the minimal drag configuration for a nosecone. However, commercially available nosecones are significantly cheaper and easier to use since they come in one piece. The drawback with commercially available nosecones is that although they come in various shapes and sizes it is difficult to find the long elliptical shape commercially.

1.1.1.5 Recovery System

One of the primary goals of Project Andromeda is that the high-powered rocket must be reusable in some aspect, and as such must be recoverable. In model rocketry, safely retrieving a rocket after it has been launched is usually a primary objective, in which the recovery system is a device, mechanism or process that allows the rocket to return safely to the ground [14].

Traditionally, model rockets will deploy a parachute after engine burnout with an ejection charge, a small explosive charge that helps separate rocket components, helping the rocket descend in a gentle and controlled manner as shown in Fig. 1-14, a NASA illustration of the flight of a model rocket [1].

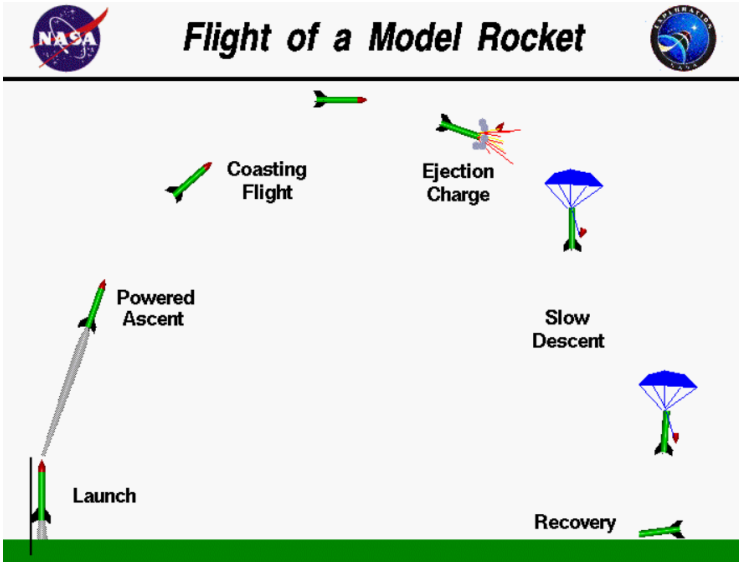


Figure 1-10 Flight of a Standard Model Rocket [1] © NASA 2020

Often in parachute recovery systems, if the parachute is large or the apogee is high, there is a drogue parachute that deploys first and helps stabilize the rocket then pulls out the main parachute to increase chances of successful parachute deployment and recovery. At apogee, the nose cone of a standard model rocket can be ejected and is usually attached by a cord that pulls the parachutes out of the fuselage for deployment. The main parachute is sized such that the descent velocity of the rocket is 3.5 to 4.5 m/s, with an upper bound of 6 m/s, and made of light fabric materials with high strength like nylon or polyester for larger parachutes with rockets larger than 300g in mass [15].

Parachute recovery systems are the most common recovery system for model rockets, however there are many methods for recovering rockets including streamer recovery, tumble recovery, gliding recovery and more [14]. Recently, novel descent and powered recovery systems have been developed and deployed by private spaceflight industry players Blue Origin and SpaceX in order to recover large-scale rockets with precision, without needing to carry a parachute [16]. Both Blue Origin's New Shepard and SpaceX's Falcon 9 rockets use aerodynamic stabilization of their rockets before igniting the engine for a landing burn, although they employ different methods for aerodynamic control and stabilization. As shown in Fig. 1-15, Blue Origin's New Shepard rocket uses a forward mounted air-brake style fairing for aerodynamic stability and to slow down during descent. An air brake refers to a flat plate that points windwards to increase drag and slow down the rocket. The fairing is at the top of the rocket as shown:

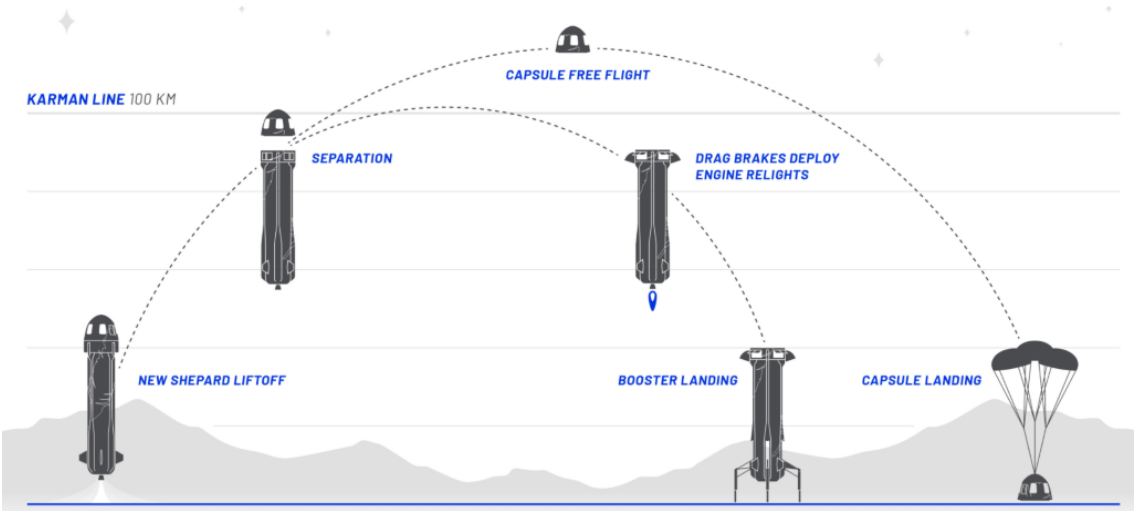


Figure 1-11 Blue Origin's New Shepard Mission Profile [17] © Blue Origin 2020

The aerodynamic drag brakes on the New Shepard are a passive stabilization system, wherein the design of the ring-shaped fairing has no control input and simply utilizes aerodynamic forces to slow down the descent of the rocket and passively transfers the center of pressure for stability [17]. This differs from SpaceX's approach with Falcon 9 (Fig. 1-12), which requires a higher precision control system for the aerodynamic control surfaces since the rocket is returning from a sub-orbital trajectory and needs to control for the divergence error early on in the descent stage to land on a precise spot [16].

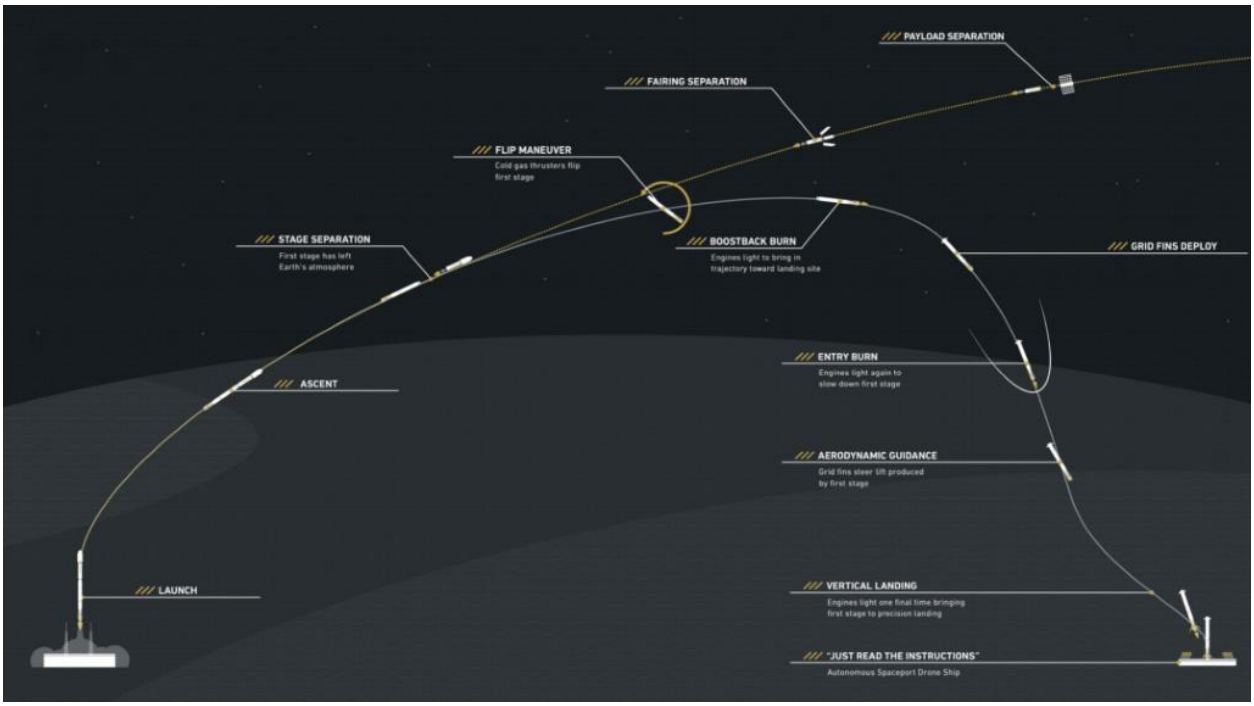


Figure 1-12 SpaceX Falcon 9 Sample Mission Profile, Falcon User Guide [7] © SpaceX 2020

The Falcon 9 uses grid-fins, which are lattice-shaped plates that provide dynamic stability and control of trajectory for Falcon 9. Historically, grid-fins have been used on missiles for both passive stability control and attitude control by varying the deflection angle of the grid-fins, which causes differential drag that results in a moment force exerted on the airframe [18]. SpaceX adopted the grid fin system for Falcon 9 to have control of roll, pitch and yaw angles on descent without using the engine and fuel necessary for landing. On the Falcon 9, these grid-fins are stowed, or folded down, on ascent in order to minimize drag and are deployed upon re-entry for drag and control, as illustrated in Figure 1-12**.** At the end of the mission profile, the Falcon 9 conducts a hover-slam burn, or suicide burn, where it uses an engine burn to cancel the velocity to

zero right above the surface of the landing site in a pseudo-hover since it cannot throttle deeply enough to achieve a true hover. This is why it is known as a hover-slam or suicide burn; if the burn is mistimed the rocket will either cancel its velocity too early and fall, or it will not reduce its velocity fast enough and impact the landing site with the excessive velocity [19].

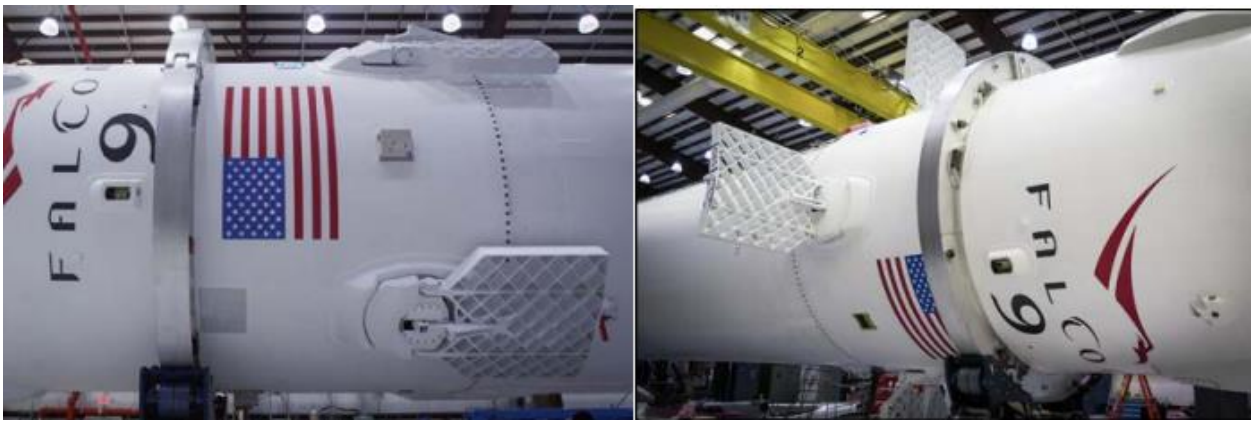


Figure 1-13 Falcon 9 Grid Fins. Left: Stowed on Ascent; Right: Deployed for Re-entry © National Academy of Sciences and SpaceX 2020

Grid fins are useful for both stabilization of attitude and as drag devices to decelerate the body they are mounted to. The advantages of using a grid fin pattern over a flat plate body or conventional planar fins are a higher strength to weight ratio associated with the grid fin pattern, as well as a lower hinge moment [18]. This means that the hinge mechanism for the grid fins, as well as the grid fins themselves, can be significantly lighter and still withstand the forces and moments encountered during flight.

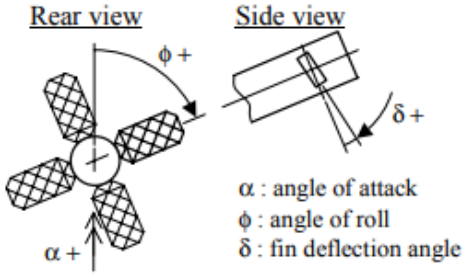


Figure 1-14 Grid Fin Model Angles [20] © ICAS 2010

As shown in Figure 1-14 , the deflection angle of the grid fins is the active control input that will modify the coefficients of normal force, axial force and hinge moment of the grid fin, which can modify the roll, pitch and yaw of the rocket when all grid fins are actuating using a control algorithm. The freestream angle of attack α is the angle between the rocket's longitudinal axis and the freestream velocity during flight, which also modifies the effective deflection angle of the fin. When modeling and simulating the grid fins effect on trajectory and flight dynamics, it is easier to break down the forces acting on each grid fin as Lift and Drag, as with airfoils, and their resultant pitching and rolling moments as functions of deflection angle [20]. The forces and their resultant moments, illustrated in Figure 1-15 to Figure 1-17 are for a rocket during descent with a grid-fin positioned on the forward end of the rocket.

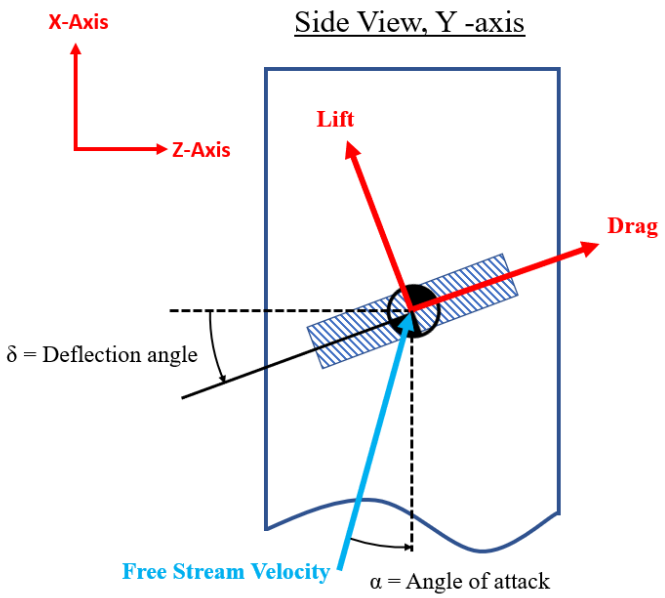


Figure 1-15 Forces on a Single Grid Fin, Y-axis Side View

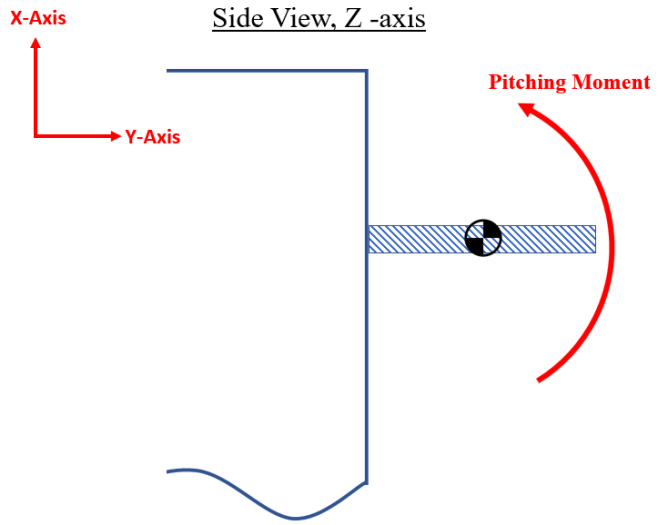


Figure 1-16 Moments on a Single Grid Fin, Z-axis Side View

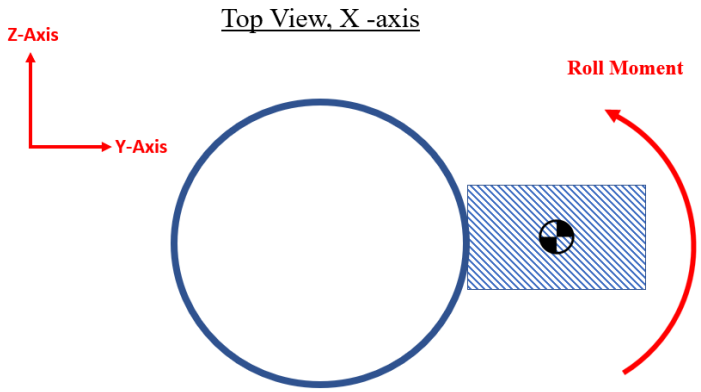


Figure 1-17 Moments on a Single Grid Fin, X-axis Top View

Grid fins can be designed to have a variety of grid patterns, shapes, and grid densities, each resulting in a different relationship between the resulting aerodynamic axial and normal forces on the grid fins versus deflection angle, angle of attack and flow velocity [21]. Because they are relatively simple and data for them is readily available in the literature, the Project Andromeda team only considered square-grid patterned grid fins. Even with square grids, there is still a large range of grid densities. Based on the dimensions of the Project Andromeda rocket, and the

performance of various grid fins patterns, the ARS team designed a grid fin, discussed in Section 2.1.2.

1.1.2 Propulsion, Thermal, & Separation Systems

Three types of common chemical propulsion systems are liquid rocket engines, hybrid rocket engines and solid rocket motors. Each chemical propulsion system operates on the same principles; by producing a change in momentum by ejecting mass at a high velocity. The exhaust gases have an increased enthalpy as a result of a combustion reaction. This thermal energy is then converted to directed kinetic energy in a converging-diverging nozzle. The change in momentum produced by the combustion reaction and subsequent acceleration of gases in the nozzle is imparted on the rocket producing a force, $\boldsymbol{T}$, in the direction opposite of the velocity of ejected mass, $\boldsymbol{v}$, as characterized by Eq. 2, where m is the mass of the vehicle [22].

$$\boldsymbol{T} = \frac{d(m\boldsymbol{v})}{dt} \qquad 2$$

A combustion reaction is triggered by the interaction between oxidizer and fuel when the two are mixed. When exposed to an external ignition (heat) source, the oxidizing compound reacts with the fuel to produce an exothermic (heat releasing) reaction. This release of energy heats the reaction products creating a high-enthalpy mixture. The high temperature gas mixture produced from this reaction is accelerated in the converging-diverging nozzle [23].

1.1.2.1 Solid Rocket Motors

High power model rockets may use either commercially available liquid rocket engines, solid rocket motors or hybrid rocket engines [24] [25] [26] [27]. Solid rocket motors (SRMs) are commonly used for model rocketry due to their simplicity. Traditionally, model SRMs use black powder fuel grains which make them relatively inexpensive and easy to ignite due to the propellants' high flammability. However, black powder has a lower energy density, and the motors are limited in size due to increased risk of cracking and the propellants inherent volatility. Thus, for higher power rocket motors a composite propellant is typically used [28]. Composite propellant grains are a mixture of solid fuel and oxidizer which are held together using a binder. A common composite propellant grain is ammonium perchlorate composite propellant (APCP) which consists

of ammonium perchlorate, metal additives, and a synthetic rubber binder [29]. Composite propellants have an advantage over black powder motors as they have a higher energy density which reduces the size of the motor for a given thrust class compared to that of a black powder motor. Additionally, composite propellants have a higher fracture strength, so they are less likely to crack [28].

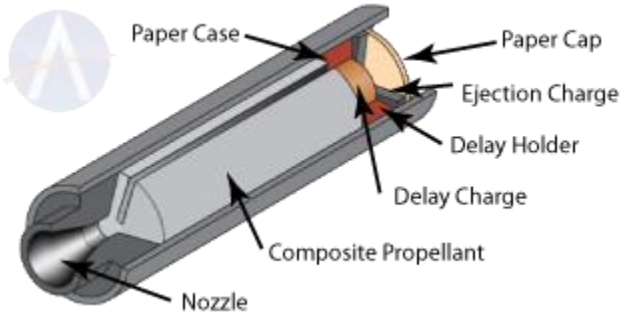


Figure 1-18 Cut-a-way View of a Composite Propellant Motor [30] © Apogee Components 2016

The composite propellant model rocket motors are simple to use. Black powder and composite propellant motors have nearly the same components as shown in Figure 1-18. A phenolic plastic case and molded phenolic nozzle are used in the composite propellant motor since these motors operate at higher temperatures and pressures. The propellant grain is molded with either a slot or center port (i.e. a cavity within the grain), which acts as the combustion chamber. This port allows for an ignitor to be placed into the SRM to ignite the motors from the aft end. Once the fuel grain is ignited a delay charge is ignited shortly thereafter. The delay charge is made from a material which burns slower than the propellant grain and is set such that it continues to burn until the rocket reaches its apogee. At apogee, the delay charge will ignite the ejection charge which can be used for stage separation or nose cone ejection [30].

Figure 1-19 Example of The Stamped Motor Code [31] © National Association of Rocketry 2015

Model rocket motors that are sold in the United States are available over a wide impulse and thrust range and are categorized based on the motor's impulse. The information necessary to understand the motor's characteristics is based on a three-part code that has been developed by the NAR. Every motor sold in the United States is stamped with this three-part code. For example, in Figure 1-19, the motor is stamped with the code C6-3. The letter "C" signifies that this motor is part of a group of motors where the total impulse ranges between 5 to 10 N-s. The number "6" specifies the average thrust for this motor to be 6 N. The final number "3" specifies, in seconds, the time delay (in seconds) between the burnout of the motor and the detonation of the recovery charge. This time delay is important as it allows for the rocket to travel to its apogee prior to activating the recovery charge. Longer delays are preferred for lighter rockets as they will have a higher apogee than a heavier rocket for the same motor. Some motors are designed without a delay charge, meaning they are not intended for use in activating a recovery system. These motors are commonly used as the lower stage motors in multistage rockets or when the delay charge is handled electronically by the onboard computer [31].

Table 1-1 Classification of Commercially Available Model Rocket Motors [32]

Classification	Impulse Range	Impulse Limit	Category
Model Rocket	1/8A	0.3125	Micro
	1/4A	0.625	Low Power
	1/2A	1.25	
	A	2.5	
	B	5	
	C	10	
	D	20	
	E	40	

	F	80	Mid Power
	G	160	
High Power	H	320	Level 1
	I	640	
	J	1280	Level 2
	K	2560	
	L	5120	
	M	10240	Level 3
	N	20480	
	O	40960	

Table 1-1 shows the classification of model rocket motors that are readily available. Motors that are classified as model rocket motors range from less than 1 N-s of impulse up to 160 N-s of impulse. These low to mid-power motors are readily available from commercial vendors. Motors that fall under the high-power classification require Level 1, 2 or 3 certifications for the operator from the NAR or Tripoli Rocketry Association to fly and in some cases purchase. Level 1 motors range from 320 to 640 N-s of impulse while Level 2 motors range from 640 to 5120 N-s. The largest hobbyist motors available are the Level 3 motors which range from 5120 to 40960 N-s of impulse [31].

1.1.2.2 Staging

Staging is a unique aspect of model rocketry that adds complexity to the rocket. A staged rocket can reach higher altitudes with a given motor by allowing the rocket to drop mass as it ascends. This allows the top stage of the rocket to travel farther because it doesn't need to carry the additional mass of lower stage [32]. Staging occurs when multiple motors are burned, each after the previous one has terminated. The first motor detaches from the fuselage after it has completed its burn and in turn, decreases the mass of the remaining stages. Rocket motors are staged either directly or indirectly, and there are advantages and disadvantages to each method.

Direct staging occurs when the upper stage motor is ignited by the lower stage motor. Because no additional electronics or launch equipment is needed, this method is the simplest and least expensive. Figure 1-20 shows the typical engine of a rocket with a black-powder

stage separation system. When the bottom stage of the rocket, also referred to as the booster stage, finishes its burn, a significant amount of heat is emitted through the nozzle into the upper stage. The booster stage is comprised solely of fast-burning propellant and can be visualized in Figure 1-21; delay grain and ejection charge are not needed for the first stage. The fast-burning propellant hurls burning particles upwards, and these hot gases instantly ignite the top stage [32]. If booster stages used a delay charge, there would be a potential safety hazard. A delayed top stage ignition may cause the rocket to turn so that it no longer points up when the top stage ignites. For the upper stage(s), a slow-burning propellant is used to delay the burn time, giving the rocket additional time to continue its ascent and reach the maximum potential altitude. Slow-burning propellant is called delay grain, which are also designed to emit a significant amount of smoke to provide easier visibility of the rocket trajectory. The propellant used for model rockets using direct staging is black powder propellant because the motors burn linearly and produce thrust. A linear burn is responsible for turning propellant into a bulkhead, without which there wouldn't be enough internal pressure to create thrust [32].

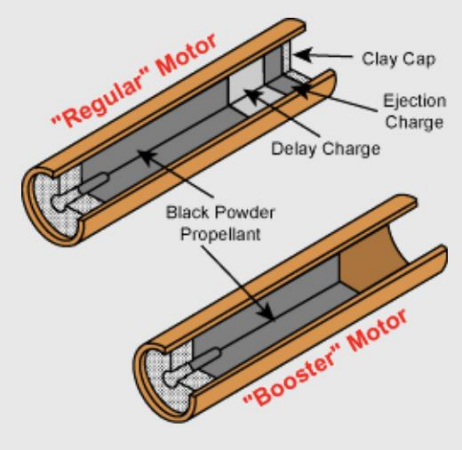


Figure 1-20 Black-powder Stage Separation Rocket Engine © Apogee Rockets 2019

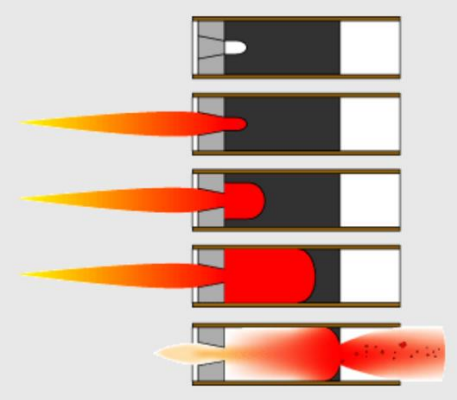


Figure 1-21 Burning Particles Produced from Black-powder Stage Separation © Apogee Rockets 2019

Indirect staging occurs when an independent source ignites the top stage of the rocket. Since a separate ignition system is required, it must be carried on the rocket, adding mass to the system. Indirect staging requires an electronic ignition, which utilizes a power source, igniter, and control device. The power source is typically a battery, but it could also be a pre-charged capacitor. An ideal ignitor will use as low a current as possible while being sufficiently reliable. The control device is needed to control the timing for the staging event. For electronic ignition, a reliable control device could be anything from a mercury switch to a radio control system [33].

Both staging methods enable rockets to reach a greater altitude by reducing mass. Even using a simpler staging approach such as direct staging is more complicated than a single stage rocket. However, staging means the thrust of the top stages have less mass to accelerate. For orbital launch vehicles, multi-staging is the most efficient and cost-effective way to transport payloads [34].

1.1.2.3 Ignition

There are a variety of ignition systems available for model rocket motors and selecting the proper igniter depends on the type of motor fuel grain. For different fuel formulations the ignition temperature can vary so it is necessary to select an igniter that will achieve a sufficiently high temperature. This can be accomplished by selecting an igniter with a pyrogen that will burn at or above the required ignition temperature. The pyrogen is a highly flammable material which covers a thin wire. When current travels through the wire it will heat up the wire and eventual ignites the pyrogen before melting the wire [35].

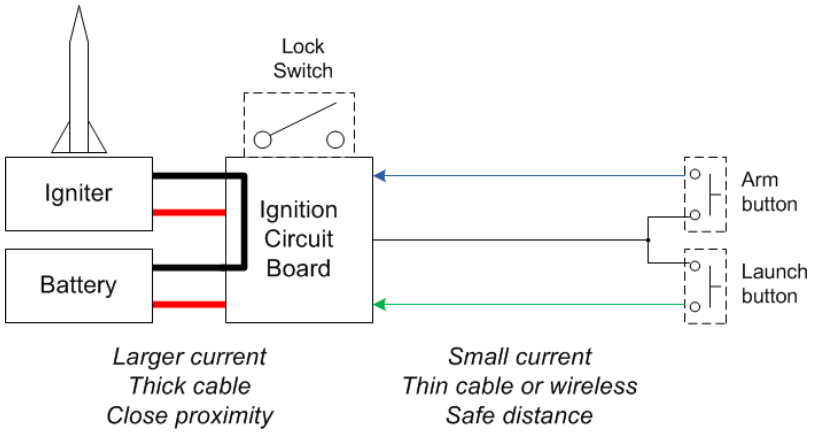


Figure 1-22 Rocket Ignition System Used on Launch Pad [36] © Robot Room 2020

Designated launch sites will have a power source and controller for supplying power to the igniter. The igniter will be connected to a power and ground lead wire and placed into the rocket motor as shown in

Figure 1-22. Wires need to run a distance of at least 100 feet to the controller, and 200 feet for larger, staged rockets to allow the rocket to be launched from a safe distance [5] [36]. It is common for rocket motors to be sold along with an igniter that is suitable for starting the motor which can be brought to the launch site with the rocket. Careful consideration is essential if the provided igniter is not used or requires replacement [37].

1.1.2.4 Motor Mount

Motor mounts are used to secure the motor throughout the rocket trajectory. Without a stable mount, the motor could damage the interior of the model rocket. In turn, this could prevent the recovery system from deploying properly and cause the entire model rocket to

crash [38]. On a smaller scale, if the motor is mounted crooked, the rocket will experience a destabilizing pitching moment, as shown in Figure 1-23. The motor is mounted correctly on the leftmost rocket and it has good alignment. The rocket on the right has a poorly constructed motor mount which will cause problems when the rocket is flown.

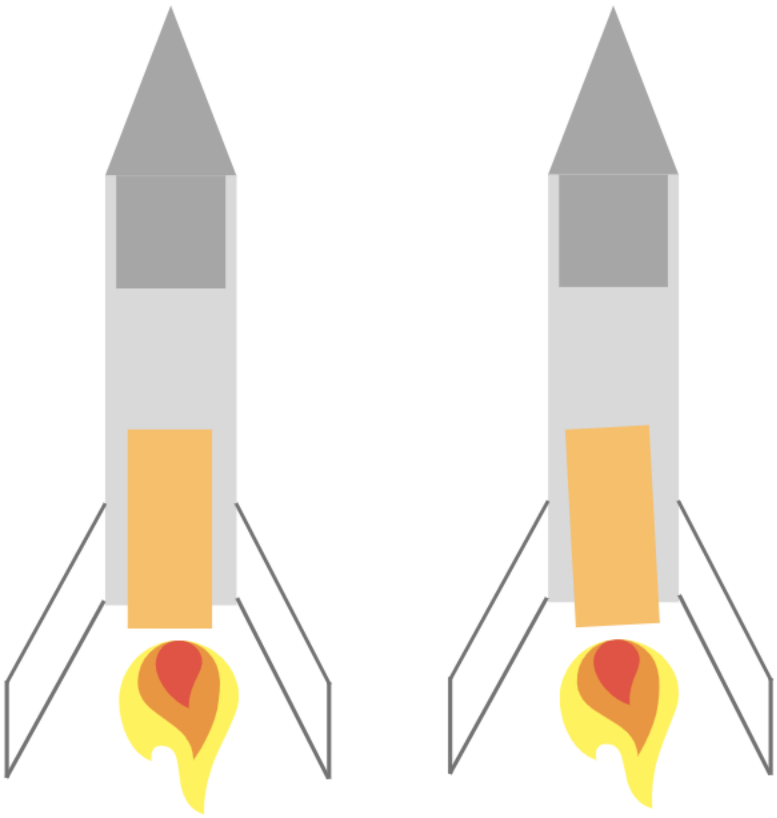


Figure 1-23 Diagram of a Straight Motor (left) Compared With an Angled Motor (right).

The generic engine mount has four components: the engine mount tube, centering rings, an engine block, and an engine hook. The rocket motor itself is inserted into the engine mount tube, which is then secured to the interior of the model rocket using two centering rings. In the front of the engine mount tube, a paper ring called an engine block is secured. The engine block is responsible for reinforcing the motor. Lastly, engine hooks are a type of motor retention often used in low-power hobby rockets as shown in Figure 1-24.

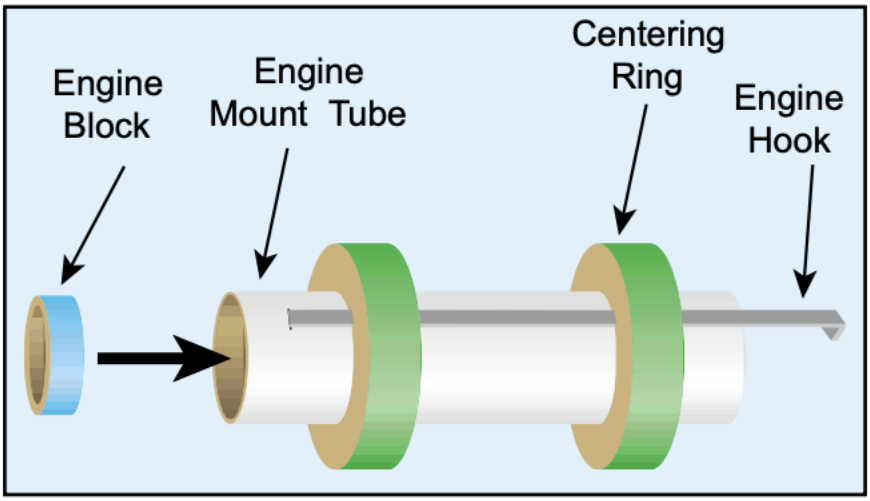


Figure 1-24 Diagram of a Low Power Motor Retention System © Apogee Components 2019

1.1.2.5 Mechanical Stage Separation

Stage separation occurs when multiple stages of a rocket detach and can help the rocket ascend to its maximum potential altitude. The ideal time for a rocket to undergo separation is at apogee, or right before apogee. Because there is minimal drag force acting on the rocket at apogee, the nose cone and upper stages can be ejected most easily. The force used to eject the upper stages and nose cone can vary. Some of the most common ejection methods include a black powder charge, mechanical separation system, spring separation system, and electromagnetic separation system.

The most common separation system is a black powder charge. This is the simplest way to eject the upper stages and nose cone. A black powder charge requires filling powder into a black powder cap. The amount of black powder is determined based on the rocket dimensions such as rocket length and fuselage diameter. An igniter wire is connected to an electronic system inside the rocket to the blast cartridge [32]. The electronic system triggers the blast cartridge by sending a current through the ignition wire in order to heat and ignite the black powder. When black powder undergoes combustion, the pressure inside of the rocket is vastly different from the ambient pressure [32].

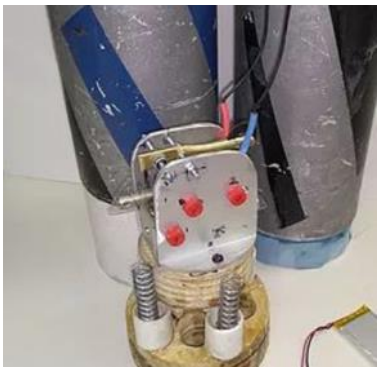


Figure 1-25 Astrojays' Upper Coupler © The John Hopkins University 2018

The John Hopkins University Astrojay's mechanical stage separation mechanism was revied and considered. Linear actuators, located in an upper coupler shown in Figure 1-25 were signaled using an altimeter, a component of their avionics systems powered by two 9 Volt batteries. Springs were used to extend the resting position of the rods and the actuators depressed when the lower coupler slid on, extending into the holes in the lower coupler. This was responsible for locking the two stages together, which created the translational and rotational holding forces necessary to prevent premature stage separation. When both couplers are engaged, the springs in the upper stage depressed and provided the necessary separation force when the actuators retracted at apogee. Unfortunately, the vertical orientation of the linear actuators proved problematic as it resulted in misalignment since the actuators were not properly sized for their rocket. In addition, the force exerted by the springs interfered with the couplers. The Astrojays tested their system multiple times, but when the separation failed due to shearing failure, they decided to forgo the more complicated method [39].

Instead, the Astrojays used a motor in the upper stage which was mounted using 6 screws to a base plate fixed on the outer body of the rocket illustrated in Figure 1-26 and Figure 1-27. The motor mount was powered by multiple battery packs, and the altimeter produced an input signal to ignite the motor. Using a screw set, the steel couple could be attached to the motor pinion. A round nut attached to a polyvinyl chloride tube mounted in the rocket's lower stage using two base plates. In order to assemble the rocket, the upper stage was screwed into the lower stage. When the rocket reach apogee, the motor was ignited, thus turning the screw from the nut. The torque of the motor causing lower stage rotation is prevented by static friction force between the two

stages [39]. In addition, the nut and screw fit helped prevent undesired torque. After testing the separation system, it was proved viable by the Astrojays team.

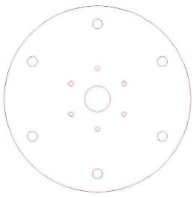


Figure 1-26 Astrojays' Motor Mount Plate Configuration © The John Hopkins University 2018

Figure 1-27 Components of the Astrojays' Mounting System © The John Hopkins University 2018

Among commercial space-flight companies, SpaceX's mechanical stage separation was considered. SpaceX uses three mechanical latches [7]. Once the engines on the first stage shut down, the latches are released via redundant actuators-driven by a high-pressure helium circuit. The high-pressure helium circuit also assists in creating an outward force to help push the upper stage after the latches release. This approach is utilized by SpaceX's Falcon 9 rocket [7].

1.1.2.6 Nozzle Design

A nozzle is used to convert enthalpy into kinetic energy. A supersonic nozzle is used by rocket engines and is designed to convert this energy as efficiently as possible. This is achieved in a converging-diverging nozzle by converting high pressure subsonic flow into lower pressure supersonic flow. This is achieved by first decreasing the area to choke the flow (i.e. achieving the maximum mass flux possible for the upstream stagnation conditions) and then increasing the area.

These types of nozzles are called converging-diverging nozzles or De Laval nozzles [22]. A basic converging diverging nozzle is shown in Figure 1-28.

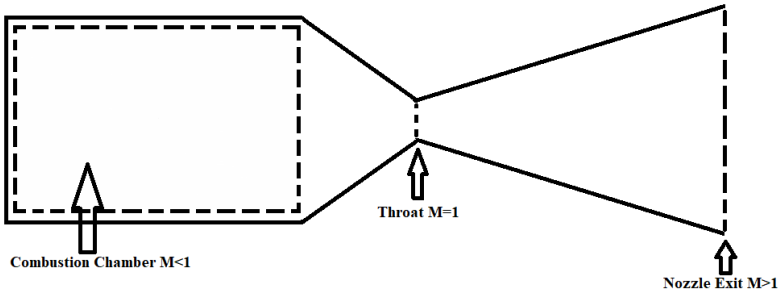


Figure 1-28 Converging Diverging Nozzle

When designing a new nozzle, it is common practice to use ideal nozzle theory to bound performance that can be corrected later using nozzle efficiencies and other semi-empirical factors. Actual measured thrust for a chemical rocket is between 1 to 6 percent lower than the calculated ideal thrust. Design based on ideal nozzle theory is useful in large part because thermodynamic properties that govern performance can be evaluated using simple mathematical relationships. The following assumptions can be made for ideal performance analysis [22]:

The flow in the nozzle is assumed to be

1. Steady State
2. Quasi One-Dimensional
3. Adiabatic
4. Frictionless
5. Frozen (i.e. non-reacting)
6. Described by the Ideal Gas
7. axially directed only

The simplest nozzle to design is an optimally expanded nozzle with constant chamber temperature, chamber pressure, and mass flow rate. Initially, the throat temperature and the throat pressure can be obtained from the chamber temperature, chamber pressure, and the specific heat ratio of the propellant in Eq. 3 and Eq. 4.

$$p_t = p_c\left(1 + \frac{k-1}{2}\right)^{\frac{-k}{k-1}} \quad (3)$$

$$T_t = T_c\left(\frac{1}{1 + \frac{k-1}{2}}\right) \quad (4)$$

The throat area can be calculated from the mass flow rate q, the throat pressure and temperature, and properties of the propellant in Eq. 5.

$$A_t = \left(\frac{q}{p_t}\right)\sqrt{\frac{R'T_t}{Mk}} \quad (5)$$

The Mach number at the nozzle exit can be obtained using the ratio of the chamber pressure to the ambient pressure and the specific heat ratio of the propellant in Eq. 6.

$$M_e = \sqrt{\left(\frac{2}{k-1}\right)\left[\left(\frac{P_c}{P_a}\right)^{\frac{k-1}{k}} - 1\right]} \quad (6)$$

The exit area of the nozzle can be obtained from rearranging the area ratio equation as the area of the throat has already been calculated in Eq. 7.

$$A_e = \left(\frac{A_t}{Nm}\right)\left[\frac{\left(1 + \frac{k-1}{2}Nm^2\right)}{\frac{k+1}{2}}\right]^{\frac{k+1}{2(k-1)}} \quad (7)$$

The final variable needed to calculate the ideal thrust is this velocity from nozzle which can be obtained using the equation below.

$$V_e = \sqrt{\left(\frac{2k}{k-1}\right)\left(\frac{R'T_c}{M}\right)\left(1 - \left(\frac{p_e}{p_c}\right)^{\frac{k-1}{k}}\right)} \quad (8)$$

The thrust equation is calculated from the mass flow rate multiplied by the nozzle exit velocity and the difference between the nozzle exit pressure and the ambient pressure multiplied by the exit area as shown in Eq. 9 [22].

$$F = qV_e + (p_e - p_a)A_e \quad (9)$$

Since this is an optimally expanded nozzle the thrust equation can be simplified to the mass flow rate times the nozzle exit velocity as shown in Eq. 10.

$$F = qV_e \quad (10)$$

The equation above are the simple mathematical relationships modeling the thermodynamic properties that nozzles use. However, a nozzle must be designed in conjunction with the propulsion system. Different propellant should be modeled as each can affect the thrust and vehicle mass in different ways. Although lighter propellant can obtain higher exit velocities and high thrust for a certain mass flow rate, they require larger tanks due to their low density. If there is a size or mass constraint, a heavy propellant would be preferred as the amount of storage needed for the same impulse is considerably less. Although nozzle characteristics are calculated based on only a few parameters, the design needs to consider the vehicle and its operating conditions.

1.1.2.7 Cold Gas Thruster

Cold gas thrusters use inert gas stored at the ambient temperature to provide thrust in a simple and reliable way. The propulsion system consists of one or more high-pressure tanks, an electrical control valve, a pressure regulator, and multiple simple nozzles. The gas is usually stored between 300 and 1,000 MPa or about 100 to 10,000 psi. Cold gas thrusters have successfully been used for over 60 years as reaction control systems. Although they are simple and reliable, the tanks are thick and heavy to contain the high-pressure gas. The specific impulse is also low, typically between 50 to 120 seconds [22].

The most typical gas used for cold gas thrusters is nitrogen. It provides a good balance of specific impulse and density. Hydrogen and helium both offer a significant boost in specific impulse, but they have extremely low densities. They would require large tanks that would make designing a rocket difficult. Hydrogen is flammable and poses a safety risk to the spacecraft since

the propellant is at risk of detonation. Carbon Dioxide has a slightly lower specific impulse, but it is about 50% more dense that nitrogen. This could be attractive for lower stages where the structural mass is more important. However, carbon dioxide is stored as a liquid and the propulsion system would need to ensure the thrust chamber is long enough for the carbon dioxide to evaporate. To illustrate the difference between the possible cold gases, the density at 3000 psi and the specific impulse at 450 psi are provided in Table 1-2 [40].

Table 1-2 Density and Ideal Specific of Cold Gas Propellants [41]

Propellant	Density (g/cc)	Ideal Specific Impulse (s)
Hydrogen	0.04	237
Helium	0.04	156
Nitrogen	0.26	64
Ammonia	Liquid	85
Carbon Dioxide	Liquid	53
Oxygen	0.28	59

The typical operating pressure for a spacecraft's cold gas system is 120 psi. Higher operating pressures will give more specific impulse and thrust, but the gains could be offset by a need for heavy components that can handle the higher pressure. There is also a limit to the operating pressure. If the operating pressure is high enough that the gas dips below its boiling point, the gas will condense and form a two-phase flow. This would make the performance predictions based on an ideal nozzle invalid [40].

A cold gas thruster system will be designed differently for a spacecraft's reaction control system versus a propulsion system closer to sea level. There are many things to consider especially when dealing with such high pressures of cold gas and the changing state of pressure and mass flow rate as the gas tank empties. Cold gas thrusters can be operated at varying pressures based on the gas used which will affect the decision on which to use.

1.1.3 Flight Dynamics Analysis

1.1.3.1 Flight Dynamics

Flight dynamics are important to consider when designing a rocket. The flight dynamics are used in order to predict the time of flight, altitude, and trajectory of the rocket. In the case of a model rocket both dynamic and aerodynamic parameters affect its flight path. In general, the most

important factors of the flight dynamics are the forces acting on the rocket, the stability of the rocket, and the controls of the rocket [41]. These factors will be discussed further in this section and the following sections.

Figure 1-29 Four Forces of Rocket Flight [42] © NASA 2014

Four major forces act on a rocket in flight. These four forces are represented in the figure above. For a rocket to move, a thrust force is required that will overpower the weight of the rocket thus propelling it upwards. The air is resistant to the rockets motion and creates a drag force that is in the opposite direction of thrust. Lift is the force generated by the movement of the particles of air as the rocket move at an inclined angle. A larger lift will be created when there is an angle produced between the rockets axis of symmetry and its velocity vector also known as the angle of attack [43]. This force is used to stabilize a rocket when it rotates in flight. There are more forces that act on a rocket in flight that will be mentioned later in the paper but for now these four forces are the major forces to consider for the stability of the rocket.

The stability of a rocket is an important factor in the performance of flight. Gusts of wind or thrust instability can affect the path of a rocket and can cause it to wobble or go off course. Rocket stability is based upon the difference between the location of the center of pressure and the center of gravity. The center of pressure is the average location of pressure where all aerodynamic forces, lift and drag, act. The center of gravity is the average location of that weight. In order to create a stable rocket, the center of pressure has to be behind the center of gravity [44]. For model rocket the stability margin is the length between the center of gravity and center of pressure over the airframes reference diameter. This should be no less than one [45]. If it is much less than one that means the rocket is less stable and will most likely not fly straight. If the margin is much larger

than one it can be too stable and if a huge gust of wind comes it will not be able to right itself to follow the intended flight path.

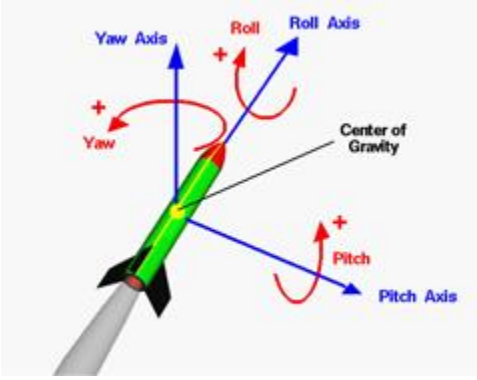


Figure 1-30 General Rocket Rotations [46] © NASA 2014

The orientation of the rocket is affected in general by the rotation of the parts of the rocket along its principle axis. This principle axis is located at the center of gravity of the rocket. Around the center of axis, three major rotations can be created by torques or moments on the rocket. The first axis is called the roll axis. This is along the center line of the rocket from the tip of the nose cone to the bottom of rocket this is where the rolling motion acts. The center of gravity is found to be along this axis. The second axis is the yaw axis which is perpendicular to the roll axis and is located at the center of gravity. This is where a yaw motion acts that causes the rockets nose to move side to side. The final axis is called the pitch axis which is perpendicular to both the roll and yaw axes at the center of gravity. This is where a pitching motion acts that moves the nose of the rocket up and down [46]. These rotations are important with helping to control the path and stability of the rocket [47]. The control methods and the method our team has selected will be discussed further in Section 1.1.3.2.

1.1.3.2 Controls

While passively stable rockets are relatively simple to design, they come with some tradeoffs. The stabilizing fins required for passive stabilization increase the drag on the rocket thereby reducing the apogee of the vehicle [48]. Higher stability in rockets also makes them susceptible to weather cocking, a phenomenon where the rocket's flight path is redirected under a crosswind [49]. Additionally, the stabilizing aerodynamic forces become less powerful as altitude

increases, and thus atmospheric pressure, reduces, eventually becoming too small to effectively stabilize the vehicle. This necessitates the use of active control.

Control involves the design of systems to get them to exhibit a desired behavior other than what the system naturally tends to. This can be to keep a system stabilized in a state where it wouldn't normally be or to improve the system's ability to quickly stabilize. Fins can technically be considered a form of passive control since they create stability in the desired orientation. Active control is distinct from passive control as it involves the use of sensors to measure the state of the rocket in order to determine a proper input to correct the state to a desired one. There are many methods for actively controlling a rocket. Four common types can be seen in Figure 1-31. Most modern rockets use thrust vector control, or gimbaled thrust, to maintain stable flight [50].

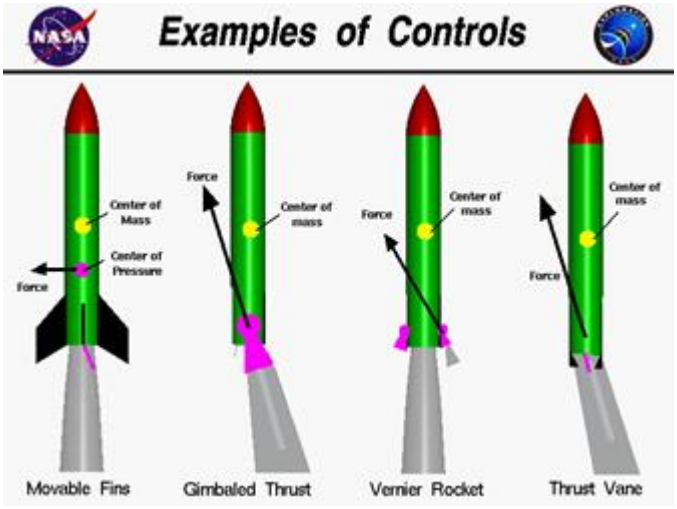


Figure 1-31 Common Methods of Attitude Control [50] © NASA 2014

Within the field of Controls, there are many options for developing control laws. One of the most popular forms is the Proportional, Integral, and Derivative (PID) controller, which computes an error value to a desired set point. As can be seen in Eq. 11, the error function, its integral, and its derivative are multiplied by a set of predetermined coefficients to determine the control input. However, PID controllers function as Single Input Single Output (SISO) systems whereas more complex problems, such as controlling rocket attitude, typically become Multiple Input Multiple Output (MIMO) problems [51]. Another option is the more advanced Linear Quadratic Regulator (LQR), which minimizes a control cost function. This is based off two

matrices, one that describes the penalty for states that are not at the set point and one that describes the cost incurred by utilizing the control inputs. These are used to compute K, the control matrix. When the output vector of the current state is multiplied by K, it produces the optimal control input vector as seen in Figure 1-32.

$$u = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt} \quad 11$$

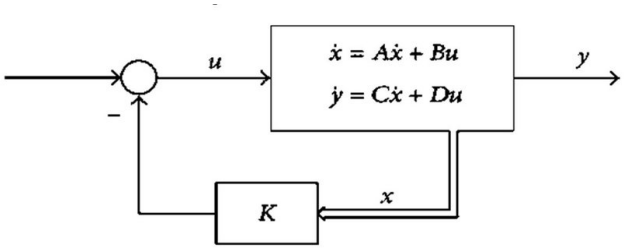


Figure 1-32 A Simplified Block Diagram of an LQR Controller [52] © Kibogora Polytechnic 2019

One problem with these two methods is that they are only applicable to linear systems. Because rocket dynamics are highly nonlinear, the system would need to be linearized about a setpoint which can be challenging [53]. One of the alternatives to these is Model Predictive Control (MPC) which incorporates control restraints. Using a model of the system, a control input is optimal for not only the current moment, but also future times, out to a time horizon, as predicted by the model as can be seen in Figure 1-33 [51]. Because MPC involves making many predictions very quickly, it can be computationally expensive but has grown in popularity as control systems become faster.

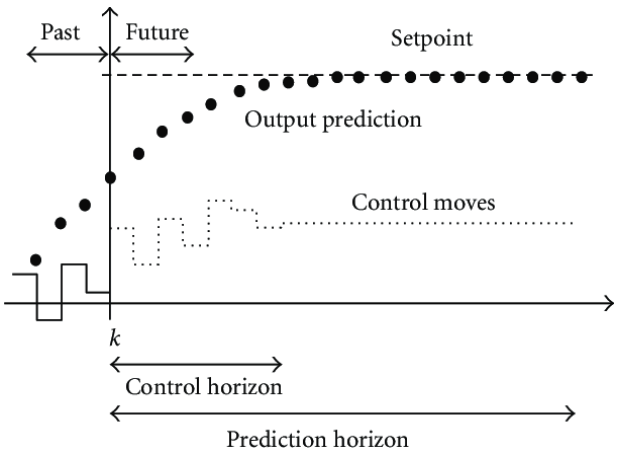


Figure 1-33 An Example of the Future State and Control Input Predictions of an MPC Controller [54] © Universidade Federal de Uberlândia 2012

Outside of classical and optimal control theory, reinforcement learning has also been used to develop control algorithms. A type of machine learning algorithm, reinforcement learning involves training an agent to take actions, in this case decide control inputs, to maximize reward. The result of this training is neural network composed of several layers of interconnected, weighted, nodes, usually called neurons, as can be seen in Figure 1-34. The advantage of this method is that it is not dependent on having an accurate model of the system. All that is required is an accurate simulator and a quantitative measure of the performance of the controller [48]. Reinforcement learning has unique disadvantages. It can be difficult to correctly train a controller without it converging to a local maximum. Training machine learning algorithms is also very computationally expensive.

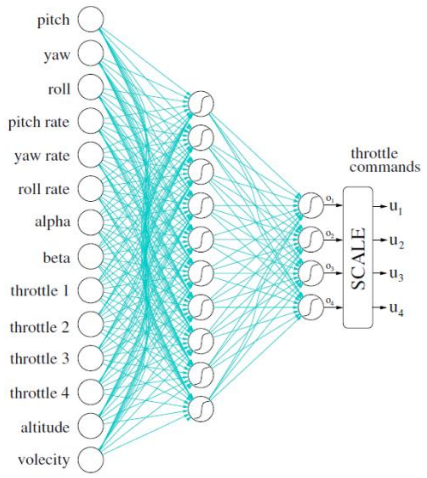


Figure 1-34 A Visualization of a Neural Network for Guidance of a Finless Rocket [48] © University of Texas, Austin 2003

While actively controlled rockets have better performance than passively controlled rockets, they are rarely seen in model rocketry [48]. This is primarily due to the significant monetary and complexity cost of active control systems which necessitate the addition of actuators, specialized embedded systems, and custom software [49]. It is worth noting that, in the past few years, thrust vector control has gained popularity in a small niche of model rocketry. The most notable example is BPS.Space, who sells the only commercially available model rocket thrust vectoring flight computer [55]. An example of a model scale thrust vector-controlled rocket can be seen in Figure 1-35.

Figure 1-35 A Model Scale Thrust Vector-controlled Rocket [55] © Barnard Propulsion Systems LLC 2020

Active control was attempted by a previous aerospace MQP at WPI. The Design and Integration of a High-Powered Model Rocket-II MPQ designed fins with an actuated flap [56]. They explored several methods of creating a control law for the vehicle which were LQR, non-linear estimation of the K matrix, and a series of educated guess. The team's Simulink diagram can be seen in Figure 1-36. Unfortunately, the team was unable to develop a control law that met their standards due to time constraints and the difficulty associated with linearizing the equations of motion of the vehicle.

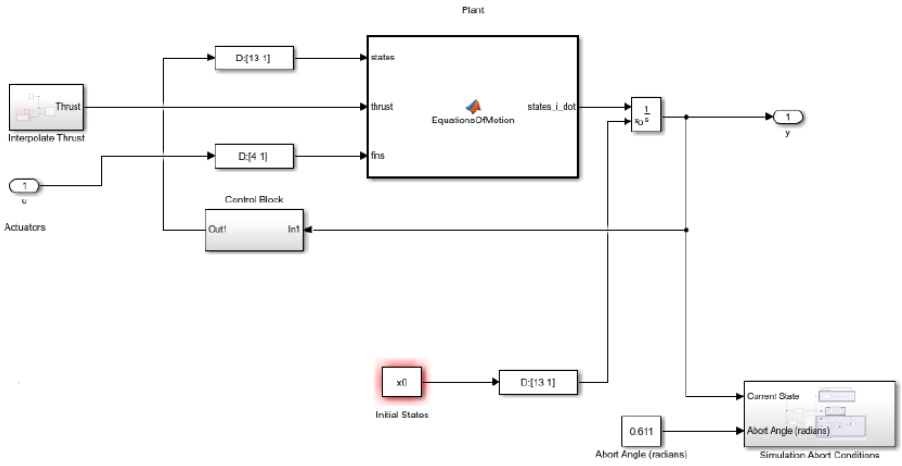


Figure 1-36 A Block Diagram of the MQP Team's Control System [56] © Worcester Polytechnic Institute 2019

1.1.3.3 Dynamical Simulation

In order to predict a rocket's flight path and validate control laws, it is necessary to create a dynamical simulation. Based off rocket and environmental parameters, a dynamical simulator needs to predict the vehicle's state over time. This is typically done using a detailed mathematical model of the rocket that describes the derivative of a state vector, which includes time varying parameters like mass, position, velocity, attitude, and rotational velocity. A layout example of dynamical simulator can be seen in Figure 1-37.

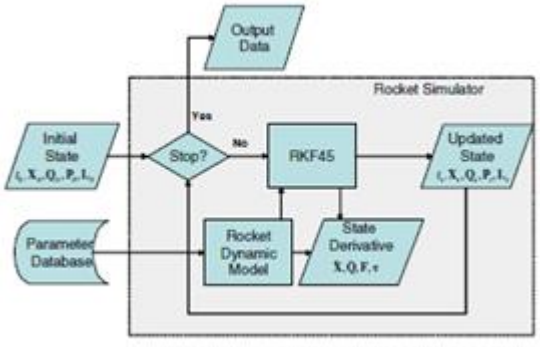


Figure 1-37 A Block Diagram of a Dynamical Simulation of a Rocket [57] © American Society of Civil Engineers 2010

In order to fully model the rocket, the simulation requires six degrees of freedom, that is, three translational and three rotational degrees. In some situations, this can be simplified to just the three translational degrees of freedom by assuming the angle of attack to be zero or when simulating parachute descent where aerodynamic forces are not strongly affected by attitude [57].

The equations of motion are derived from Newton's 2nd Law of Motion. By summing the forces and moments on the vehicle, the acceleration and angular acceleration can be found. The four primary forces on a rocket are thrust, gravity, drag, and lift. The equations of motion can be used to create a state space model. This can be numerically integrated to find the state over time using methods such as Runge-Kutta methods, like RK4 or RKF45. MATLAB implements a number of these algorithms using functions such as ode45 and ode23 [58].

It is common in literature to use the Monte Carlo method in the simulation to model uncertainty in parameters and initial conditions [57] [59] [60]. This typically involves making

selected parameters and initial conditions gaussian random variables. Over the course of many iterations, the mean and standard deviation of various outputs can be computed. This can be used to predict not only a landing position, but also an area around it where the vehicle is most likely to land [57]. In situations involving control input, a Monte Carlo simulation can also be used to predict the probability of success of the control law [60]. It is possible to implement the Monte Carlo method using a wrapper around the main dynamical simulation code [57].

A dynamical simulator was created as a part of the previous rocket MQP. While the team initially planned on developing a full six degree of freedom simulation, time constraints prevented them from fully implementing it [53]. They utilized a three degree of freedom simulation to solve Newton's 2nd Law equations of motion by assuming the angle of attack to be zero. The attitude of the vehicle was described using Euler's Equations for rigid body dynamics, but they were not implemented in the final simulator.

1.1.3.4 Aerodynamic Modeling

When an object moves through a fluid a set of forces or loads act on it. The four governing forces mentioned in section 1.1.3.1 are a such an example. In a nonidealized case, there are more than four forces acting on the object. In general, when an object is traveling through a fluid, the forces that have to be considered are the surface force, pressure force, viscous force, unsteady force, body force, friction force, Coriolis force, centripetal force, and tangential force [61]. Based on the situation, a set of assumptions could be made to simplify the problem. For example, in the case of a laminar flow the unsteady force would go to zero so that would be one less variable to consider. In the case of an inviscid flow, the viscous force could be considered zero [62]. However, these assumptions do not always work in the real world.

One of the most accurate ways of examining the aerodynamics forces and loads on an object is through a wind tunnel test. However, this method is time consuming and expensive. An alternative method is simplified equation calculation but in this case many assumptions are made and the answers are in the ideal situation with generalized solution over the surface. This is less accurate and not the best option if detailed oriented results are the goal. Therefore, the method of Computational Fluid Dynamics (CFD) is an alternative option. With CFD, the variation of force across a surface can be found and the calculations are more accurate than hand calculations. In

addition, CFD takes less time and costs less than running a test in a wind tunnel [63]. However, there are some challenges too.

Coordinates: (x,y,z)
Velocity Components: (u,v,w)
Time : t Pressure: p Heat Flux: q
Density: ρ Stress: τ Reynolds Number: Re
Total Energy: Et Prandtl Number: Pr

Continuity:
$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u)}{\partial x} + \frac{\partial (\rho v)}{\partial y} + \frac{\partial (\rho w)}{\partial z} = 0$$

X – Momentum:
$$\frac{\partial (\rho u)}{\partial t} + \frac{\partial (\rho u^2)}{\partial x} + \frac{\partial (\rho uv)}{\partial y} + \frac{\partial (\rho uw)}{\partial z} = -\frac{\partial p}{\partial x} + \frac{1}{Re_r}\left[\frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + \frac{\partial \tau_{xz}}{\partial z}\right]$$

Y – Momentum:
$$\frac{\partial (\rho v)}{\partial t} + \frac{\partial (\rho uv)}{\partial x} + \frac{\partial (\rho v^2)}{\partial y} + \frac{\partial (\rho vw)}{\partial z} = -\frac{\partial p}{\partial y} + \frac{1}{Re_r}\left[\frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{yz}}{\partial z}\right]$$

Z – Momentum
$$\frac{\partial (\rho w)}{\partial t} + \frac{\partial (\rho uw)}{\partial x} + \frac{\partial (\rho vw)}{\partial y} + \frac{\partial (\rho w^2)}{\partial z} = -\frac{\partial p}{\partial z} + \frac{1}{Re_r}\left[\frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \tau_{zz}}{\partial z}\right]$$

Energy:
$$\frac{\partial (E_T)}{\partial t} + \frac{\partial (uE_T)}{\partial x} + \frac{\partial (vE_T)}{\partial y} + \frac{\partial (wE_T)}{\partial z} = -\frac{\partial (up)}{\partial x} - \frac{\partial (vp)}{\partial y} - \frac{\partial (wp)}{\partial z} - \frac{1}{Re_r Pr_r}\left[\frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} + \frac{\partial q_z}{\partial z}\right]$$
$$+ \frac{1}{Re_r}\left[\frac{\partial}{\partial x}(u\tau_{xx} + v\tau_{xy} + w\tau_{xz}) + \frac{\partial}{\partial y}(u\tau_{xy} + v\tau_{yy} + w\tau_{yz}) + \frac{\partial}{\partial z}(u\tau_{xz} + v\tau_{yz} + w\tau_{zz})\right]$$

Figure 1-38 Navier-Stokes Equations [64] © NASA 2015

For CFD there are three governing equations: mass conservation equations, momentum conservation equation, and energy conservation equation. The general form of the equations most often used in CFD is shown above in Figure 1-38. These are the Navier-Stokes equations [64]. These equations in theory could be solved by hand but in reality, they cannot be because there are too many unknown variables that change over time. Therefore, CFD was initially created to numerically solve these equations [65].

There are a few things needed in order to solve the equations for a model. The geometry of the object, the selected domain for modeling, and the initial properties of the domain must be defined. The boundary conditions constraints are needed to solve the problem and must be defined, such as the velocity of the fluid at the wall [66]. In addition, a mesh needs to be created around the domain of either the object being subjected to the fluid or the fluid itself. Meshing, as stated by Simscale "is the process of discretization of the domain into small cells or elements to apply the mathematical model under the assumption of linearity." [67] This is done in order to analyze a

complex domain. Once all of these conditions are met, the CFD model can be run. Due to the equations being solved numerically, this system will be run over and over again until it converges at every timestep to produce an answer. If this is not accomplished at first, it may require a different mesh. This process will be repeated until it converges.

In past MQPs, they have used Ansys Fluent and Simscale to model the aerodynamics of their rockets [53]. Simscale seems to be easier to operate however Ansys Fluent appear to perform better fluid analysis. In addition, we have access to Ansys Fluent software already and know it is better for modeling in three dimensions. However, it has also been described as a more challenging platform to use [68]. We plan to initially try both softwares for modeling the aerodynamics this year.

1.2 Overall Project Goals

Project Andromeda had two overall project goals. They are as follow:

- To work as a team to design, build and test an advanced high-power model rocket that's overall vehicle performance is dependent on the mass and the individual components and assemblies.
- Provide specialized training and the chance to apply software tools: MATLAB, Ansys Static Structural Analysis, Ansys Fluent, Ansys Dynamic Analysis, Cantera, others.

The individual subteams, ARS, PTSS and FDA, set more specific subteam goals to be accomplished by their members. They are as follows.

- Airframe and Recovery System (ARS)
    - Design, build, and test airframe and components within
        - Identify critical locations of high stress throughout the airframe and internal structure during peak acceleration loads using MATLAB and Ansys
    - Design, build, and test an innovative recovery system
        - Evaluate the aerodynamic forces and moments on a single grid fin and on the airframe with a set of grid fins

        - Identify critical locations of high stress throughout the grid fin during peak acceleration loads.

- Propulsion, Thermal, and Separation Systems (PTSS)
    - Select and model a commercially available motor.
    - Estimate thrust and impulse using Cantera and MATLAB.
        - Model temperature distribution from combustion through the propellant grain, motor casing, and rocket body using COMSOL.
        - Design, fabricate, and test a mechanical linear stage separation system.
    - Calculate the force required to separate the stages.
        - Design a mechanical stage separation mechanism to separate the lower and upper stage of the rocket.
        - Perform ground tests the determine the performance of the separation system.
    - Design, fabricate, and test a cold gas descent thruster.
        - Model the thrust and impulse of a cold gas thruster using MATLAB
        - Test the cold gas thruster to determine its performance.
- Flight Dynamics Analysis (FDA)
    - Perform analysis of aerodynamic loads on the vehicle during flight.
    - Create a six degree of freedom dynamical simulation of the rocket's flight dynamics.
    - Analyze key rocket performance parameters (apogee, flight time, ect.) to support design work and flight planning.
    - Design and verify a descent and landing control algorithm for controlling the grid fins and descent thruster.
    - Support the implementation of the control algorithm and integration with the recovery system.
    - Support the design and integration of the avionics system.

1.3 Project Design Requirements, Constraints, and Other Considerations

The *design requirements* for the entirety of Project Andromeda were shared between three subteams (ARS, PTSS and FDA). The design requirements consisted of:

- An innovative rocket with an active recovery system to enable reusability and demonstrate non-parachute-based recovery systems
- Utilize a cold gas thruster system as part of the recovery system
- Utilize grid fins for attitude control on descent as part of the recovery system
- Use a camera mounted on the airframe to record video during testing and flight
- Black powder charge separation for nose cone and mechanical separation for motor section
- Use a Level II high power motor for ascent to achieve an acceptable thrust to weight and achieve high altitude required to reach stable terminal descent during recovery

The *design constraints* for the entirety of Project Andromeda were shared between three subteams (ARS, PTSS and FDA). The design constraints consisted of:

- The rocket must have a velocity of at least 15 m/s (49.2 ft/s) at the time it disconnects from the launch rail to ensure that it is dynamically stable.
- The rocket must be able to be assembled and prepped for flight in less than two hours to ensure that it can be prepared at a launch site.
- The rocket must have a landing velocity of 6 m/s (19.7 ft/s) or lower to ensure a safe landing.
- The rocket must be lightweight to allow for an acceptable thrust-to-weight of 5:1.
- The rocket must leave the launch rails at a velocity in which the aerodynamics forces have a significant effect on the rockets' stability.
- The internal components must be non-interfering with nearby components and be encompassed by the rocket's airframe.
- The internal components must be easily accessible when rocket is disassembled.

The *design considerations* for the entirety of Project Andromeda were shared between three subteams (ARS, PTSS and FDA). The design standards consisted of:

- All parts of the projects must be completed under WPI's COVID-19 Guidelines [69].
- All launch activities will be done in accordance with the NAR High Power Safety Code [5].

1.4 Tasks and Timetable

In order to achieve the goals listed in Section 1.2, specific tasks for each subteam were identified. A list of the analysis tasks for each subteam is provided in Table 1-3, along with the Section of the report containing the methodology and analysis/results for the task. A detailed description of analysis task is provided for each subteam in Table 1-4, Table 1-5, and Table 1-6.

Table 1-3 Analysis Task Section Indicator

Airframe and Recovery System Subteam	**Section**
ARS Analysis Task 1: Airframe Stress Distribution	2.2.3
ARS Analysis Task 2: Grid Fin Aerodynamic Loads	2.2.4
ARS Analysis Task 3: Grid Fin Stress Distribution	2.2.5

Propulsion, Thermal, and Separation System Subteam	**Section**
PTSS Analysis Task 1: Motor Performance Model	3.3.1
PTSS Analysis Task 2: Temperature Distribution	3.3.2
PTSS Analysis Task 3: Mechanical Separation System Model	3.3.3

PTSS Analysis Task 4: Cold Gas Descent Thruster Model	3.3.4

Flight Dynamics Analysis Subteam	**Section**
ARS Analysis Task 1: Airframe Stress Distribution	4.2.1
ARS Analysis Task 2: Grid Fin Aerodynamic Loads	4.2.2

Table 1-4 Airframe and Recovery System (ARS) Subteam Analysis Tasks

ARS Analysis Task 1: Airframe Stress Distribution
Problem Statement:
Identify critical locations of high stress throughout the airframe and internal structure during peak loads.
Solution Methodology:
• Tool(s): Ansys Mechanical Workbench / Dynamic Analysis, SOLIDWORKS Stress Analysis • Inputs: Airframe solid model including information on materials, joint/bond point models, aerodynamic loads (forces and moments), impulse from impact events
Analysis Products:
• Map of stress distribution • Identification of critical locations/joints
Use of Results:
• Stress data will be used to determine weak points within airframe design during landing impact • Determine method(s) to improve structural integrity of the airframe and internal structures e.g. electronics bay and motor bay

ARS Analysis Task 2: Grid Fin Aerodynamic Loads
Problem Statement:
Evaluate the aerodynamic forces and moments on a single grid fin and on the airframe with a set of grid fins
Solution Methodology:
• Tool: Ansys Fluent • Required Inputs: Grid fin solid model, initial and boundary conditions (flight velocity, etc.), fluid properties, mesh characteristics, number and placement of fins on the airframe • Evaluate forces and moments produced by grid fin options
Analysis Products:
• Net forces and moments on airframe for each "case" considered. A "case" consists of a combination of fin type, placement, number, and flight conditions. • Estimate of loads on single fin
Use of Results:

• Forces and moments on vehicle (with multiple fins) will be used as an input in the vehicle dynamics model • Loads on single fin will be used as input in grid fin stress analysis.

ARS Analysis Task 3: Grid Fin Stress Distribution
Problem Statement:
Identify critical locations of high stress throughout the grid fin during peak acceleration loads.
Solution Methodology:
• Tool(s): Ansys Mechanical Workbench / Dynamic Analysis, SOLIDWORKS Stress Analysis • Required Inputs: Fin solid model including information on materials, landing speed aerodynamic loads on the fin • Estimate stresses on the fin as a function of landing speed and vehicle orientation
Analysis Products:
• Map of stress distribution • Identification of critical locations/joints
Use of Results:
• Identify maximum allowable landing speed (as a function of vehicle orientation) • Determine method(s) to improve structural integrity of fin and fin-airframe attachment method

Table 1-5 Propulsion, Thermal and Separation Systems (PTSS) Subteam Analysis Tasks

PTSS Analysis Task 1: Motor Performance Model
Problem Statement:
Create a simplified model for the selected motors, that can be used to estimate performance (thrust and Isp) that can be compared with published data
Solution Methodology:
• Tools: Cantera, MATLAB • Major Assumptions: chemical equilibrium in chamber, frozen flow, steady state, isentropic flow in nozzle • Required Inputs: Propellant composition and properties, chamber and nozzle geometry, ambient conditions (p, T) • Formulate model that couples equilibrium chemistry with flow through nozzle
Analysis Products:
• Composition of combustion products, chamber pressure and temperature, mass flow rate, thrust, specific impulse
Use of Results:
• Compare predicted performance with manufacturer data • Estimate heat flux for thermal analysis

PTSS Analysis Task 2: Temperature Distribution
Problem Statement:
Estimate the temperature distribution through the propellant grain, motor casing, and rocket body subject to heat flux from the motor
Solution Methodology:

• Tool: COMSOL • Required Inputs: Material and property data for motor, casing, and rocket body. External, flight boundary conditions (velocity, ambient pressure, temperature, air properties)
Analysis Products:
• Temperature and heat flux distribution through motor and airframe structure
Use of Results:
• Conduct thermal analysis on the motor body tubes to see if there are any points of risk for structural degradation due to overheating (adhesives used for joints, etc.)

PTSS Analysis Task 3: Mechanical Separation System Model
Problem Statement:
Create a simplified model for the mechanical-based stage separation system
Solution Methodology:
• Tools: MATLAB, Excel, force balance (scale) • Experimentally measure force required for separation • Perform stress analysis (depending on final design) • Required Inputs: Force required to jettison the nosecone (or other stage), geometry (length, coil and wire diameters)
Analysis Products:
• Number and compression distance required for selection of commercially available necessary materials
Use of Results:
• Evaluate design options (number, geometry, and compression distance) for commercially available necessary products

PTSS Analysis Task 4: Cold Gas Descent Thruster Model
Problem Statement:
Create a simplified model for the cold gas descent thruster(s)
Solution Methodology:
• Tools: MATLAB • Required Inputs: reservoir conditions, gas properties, valve and nozzle characteristics • Formulate model using ideal (isentropic flow) to calculate mass flow rate, thrust, and specific impulse as a function of time from time from when thruster is actuated until propellant is spent
Analysis Products:
• Performance plots (and curve fits) for thrust, mass flow rate, and Isp
Use of Results:
• Used by FDA team to model descent trajectory and vehicle attitude

Table 1-6 Flight Dynamics Analysis (FDA) Subteam Analysis Tasks

FDA Analysis Task 1: Vehicle Dynamics and Performance Model
Problem Statement:
Create an integrated model that can be used to estimate the vehicle attitude dynamics (angles and rates) as a function of time from launch to impact, as well as the rocket trajectory (including max altitude and range)

Solution Methodology:
• Tool: MATLAB • Required Inputs: rocket geometry and inertia properties; center of pressure, center of mass; thrust, simplified drag and moment models/data from related analysis tasks, wind profile in given topography, avg. wind speed across the rocket's altitude range • Formulate model consisting of *two, coupled systems* of nonlinear ODEs, one for attitude dynamics (Euler solver) and one for the equations of motion (Newton's 2nd law) describing the vehicle trajectory. Euler equations are solved at each time step as the equations of motion are solved for the trajectory
Analysis Products:
• Simulation of rocket trajectories, capturing statistically randomized variation in wind speed and direction • Evaluation of forces and moments acting on vehicle of given design and flight state (attitude and velocity) when subject to wind disturbances and their effect on trajectory
Use of Results:
• Landing probability distribution plot (safety plot) • Data that can be used to compare stability and performance of innovative fin design with baseline • Provide estimates of perturbation on vehicle attitude (angles and rates) when subject to transient wind disturbances. • Estimate upper limit on wind disturbance to maintain stable flight • Evaluation of effectiveness of innovative fin design, i.e. compare vehicle's ability to maintain stable flight with baseline design

FDA Analysis Task 2: Vehicle Aerodynamic Loads – Simulation
Problem Statement:
Estimate the aerodynamic forces and moments on the vehicle as a function of velocity and vehicle attitude
Solution Methodology:
• Tools: SimScale, Ansys Fluent • Required Inputs: rocket geometry and inertia properties, center of pressure, center of mass, drag and moment coefficient data for similar vehicles (from literature), initial and boundary conditions, fluid properties, mesh characteristics, wind profile in given topography, avg. wind speed across the rocket's altitude range
Analysis Products:
• Pressure contours, plots of forces and moments acting on vehicle of given design and flight state (attitude and velocity) • Estimate of forces and moments acting on vehicle of given design and flight state (attitude and velocity) when subject to wind disturbances
Use of Results:
• Provide load estimates to be used in structural stress analysis • Provide forces and moment data that can be used in "table-lookup" for simulation of vehicle trajectory and attitude • Provide estimates of perturbation on vehicle attitude (angles and rates) when subject to transient wind disturbances.

To track the project's progress during the year in order to stay on schedule and achieve the tasks a Gantt chart was created. It was made from an example template from excel and is located in Appendix A: Gantt Chart.

2 Airframe and Recovery System

Using the information gathered and described in the literature review section, each subteam was able to design and analyze their respective subsystems. The ARS team proceeded to create the designs for the recovery system components of the Project Andromeda rocket. After we completed our initial designs, we applied the general process of analysis and iteration as results inspired design changes. The primary subsystems that the ARS team is responsible for included the following: an airframe capable of handling the stress of ascent and recovery, grid fins capable of changing the attitude of the rocket and reducing terminal velocity. The ARS subteam was also responsible for demonstrating that the grid fin design can handle the induced stresses.

2.1 Methodology

The first stage of the Project Andromeda rocket, containing the ascent motor mount and stability fins, will separate at apogee and descend with a parachute, as in the mission profile shown in Figure 2-1, which is also presented in Section 1.

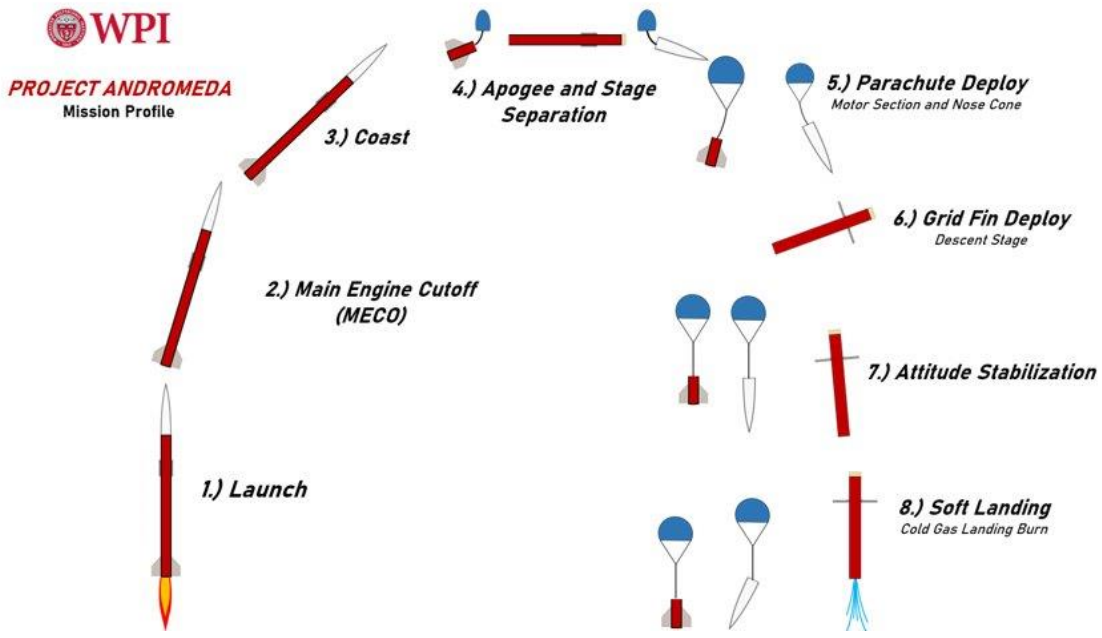


Figure 2-1 Project Andromeda Mission Profile

The nose cone will also eject at apogee and descend with a separate parachute. Project Andromeda's second stage, the descent stage, makes up the bulk of the rocket. The descent stage will utilize grid fins for attitude stabilization and drag braking; however, the grid fins will not be able to steer to control the trajectory in order to reach the landing site. The grid fins will only be able to stabilize the rocket attitude yaw, pitch, and roll and their respective angular rates. The rocket was intended to be capable of autonomously controlling its attitude to align the rocket during descent to be vertical and perpendicular to the ground. This means that day-of-launch conditions such as wind or turbulence could bring the rocket anywhere downrange of the launch site. In order to achieve a final recovery, the rocket descent stage will use a compressed air tank feeding a small cold gas thruster to slow down its vertical velocity to a safe landing speed. The goal is that the descent stage landing velocity after the cold gas thruster activates will be close to the descent velocity of 6 m/s as with a parachute. This cold gas thruster system is discussed in more detail in Section 3.1.8.

2.1.1 Airframe

The Airframe and Recovery System subteam chose to use Blue Tube 2.0 as the material for the airframe of the rocket. Blue Tube 2.0 has a high strength and light weight, making it a great material to use as the structure of the airframe. Project Andromeda utilizes two stages, splitting the airframe into two sections. The first section is the motor stage, and houses the motor bay, stability fins, and drogue chute. The second section is the descent stage, and accommodates the cold gas thruster bay, the avionics bay, the grid-fin recovery system bay, and the connection to the nosecone. These stages are connected by a coupler tube, which is also made from Blue Tube 2.0. The two different rocket geometries for Ascent and Descent are shown in Figure 2-2.

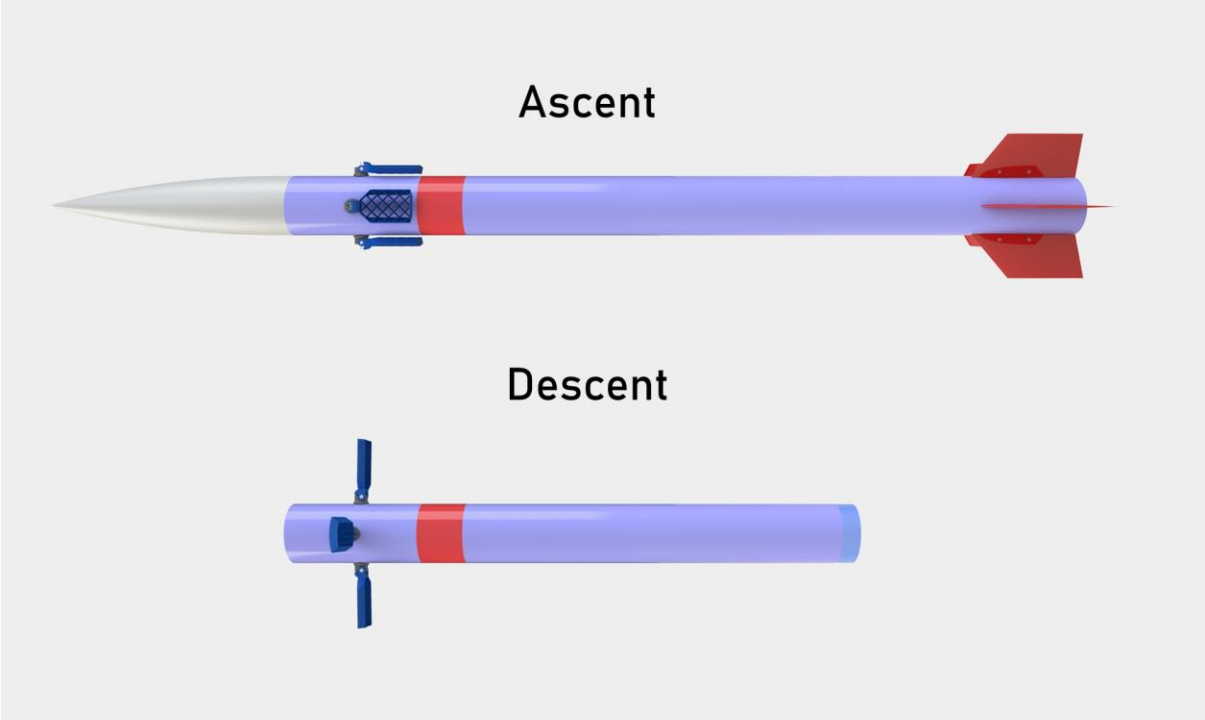


Figure 2-2 Ascent and Descent Geometry

The motor stage airframe is 16 in. (40.6 cm) in length, and the descent stage airframe is 36 in. (91.4 cm) in length. Both sections use a 4 in. (10.2 cm) piece of Blue Tube and have an outer diameter of 4.024 in. (10.221 cm) and an inner diameter of 3.91 in. (9.931 cm). A 4 in. airframe was chosen in order to provide ample space for all subsystems, specifically the cold-gas thruster bay and motor bay. The coupler tube used has an outer diameter of 3.888 in. (9.875 cm) in order to produce a clean fit into each section of 4 in. airframe. The general layout of the rocket was designed using OpenRocket to help with stability and estimate the maximum velocity reached in ascent, as shown in Figure 2-3

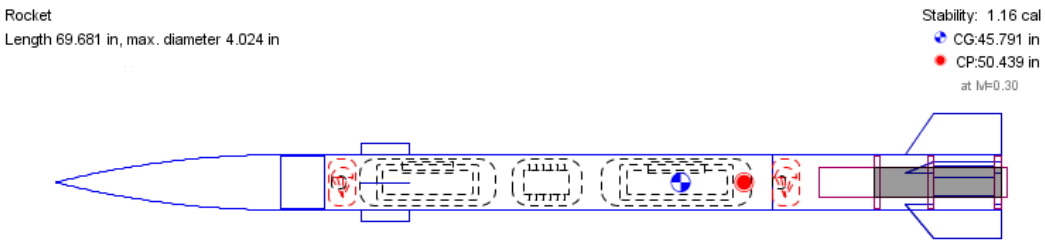


Figure 2-3 OpenRocket Diagram

As is the case for most model rockets, bulkheads are designed to hold different segments of the rocket together. For our high-powered rocket, bulkheads were used to secure the motor bay, cold gas thruster and tank, and avionics bay. The airframe and recovery systems subteam was responsible for creating the CAD assembly of the full rocket with all subsystems, including designing the bulkheads that will hold the cold gas tank, the avionics bay, and the grid fins. Figure 2-4 shows a cross section of the Project Andromeda rocket full CAD assembly, with parachutes for the ascent stage and nosecone omitted.

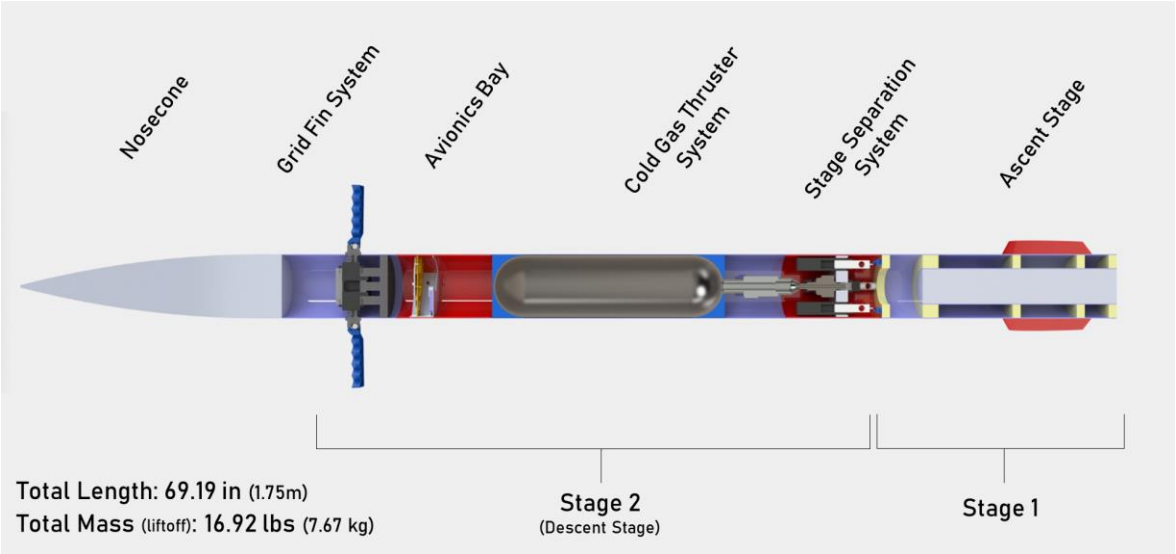


Figure 2-4 Cross Section, Project Andromeda Rocket

The bulkhead that connects the grid fin section will attach the servos that deploy the fins. Figure 2-5 below shows an exploded view of the rocket, with each subsystem and all bulkheads shown separated from the airframe.d

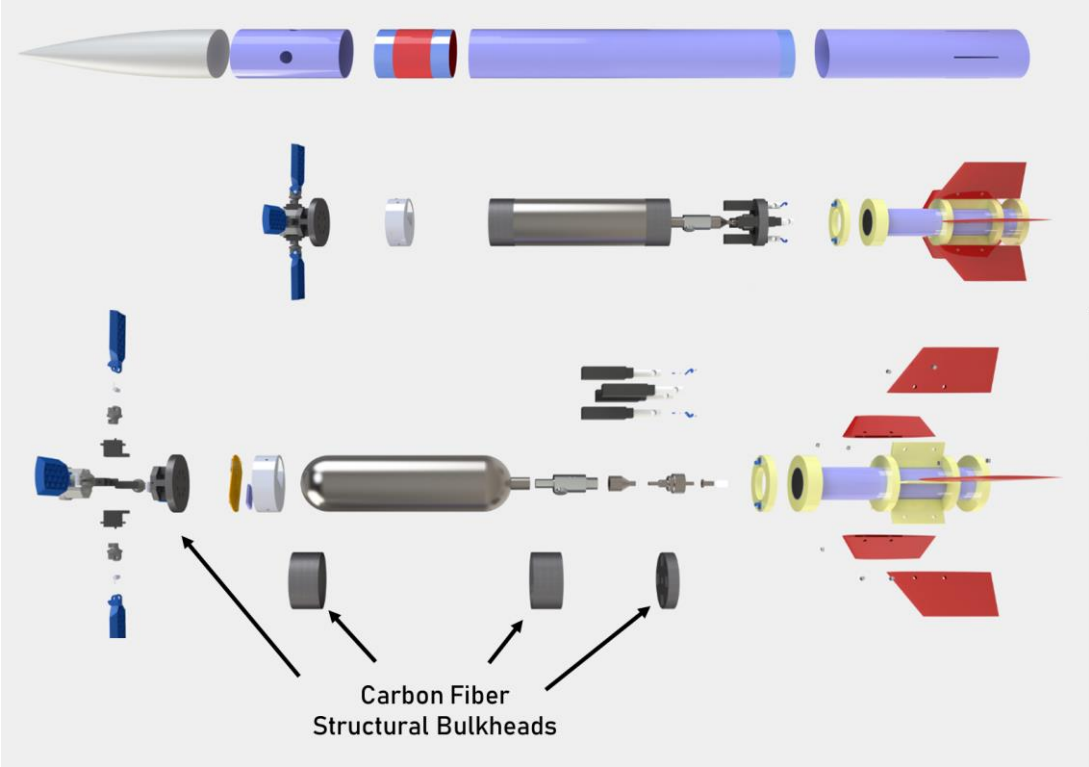


Figure 2-5 Exploded View, Project Andromeda

As can be seen from Figure 2-5, our group chose to make our bulkheads out of a composite material. Our first choice was to machine them out of aluminum, and then drill holes in them to decrease the mass. These holes would have been modeled in SOLIDWORKS and Ansys in order to create the lightest and most efficient design. However, this would have taken too much time to machine. Therefore, we pivoted our ideas to use a composite material which would easier to manufacture when 3D printed with the carbon fiber reinforced 3D printers available at WPI.

2.1.2 Grid Fins

The final design of the grid fin system designed by the ARS subteam can be seen in the 3D rendering in Figure 2-6.

Figure 2-6 Grid Fin System 3D Render

The initial design of grid fins was focused on ensuring the fins could complete their function during recovery operations, while staying clear during ascent. The grid fins must be stowed on ascent and deployable after apogee by the flight computer. To increase the chances of a successful grid fin deployment, the grid fins should be deployed in such a way that the drag on the rocket will tend to pull them open. Further, we included a spring on the hinge of the grid fin to assist in deployment in case of opposing wind or misaligned freestream velocity. With that in mind, we created a conceptual drawing of the mechanism for a single grid fin, shown in Figure 2-7.

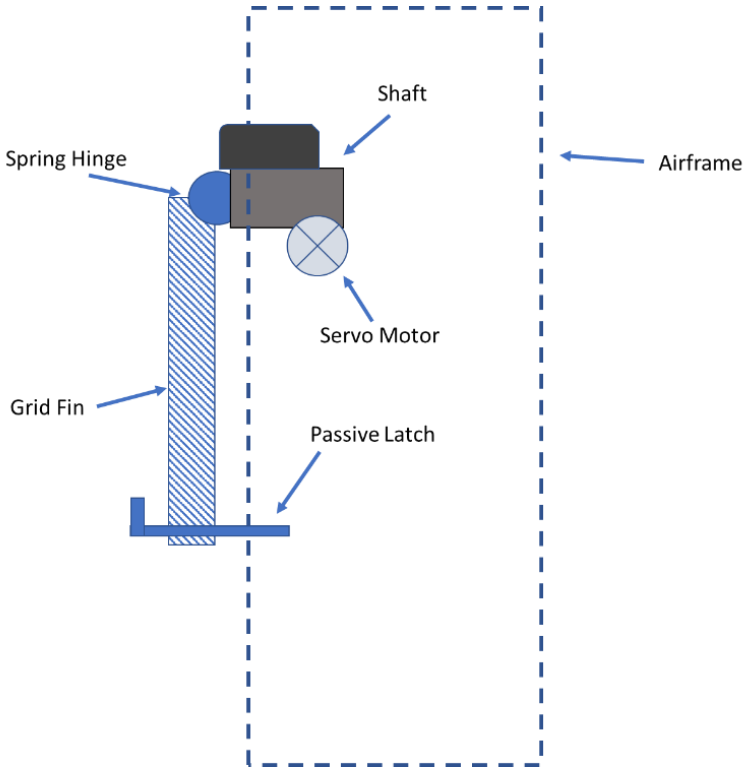


Figure 2-7 Single Grid Fin Overview, Stowed on Ascent

The grid fin is held in place on ascent by a solid L-shaped “passive” or fixed static latch that is mounted directly to the airframe, which prevents the grid fin from swinging outward. The grid fins are to be actuated by a servo motor that is capable of deflecting the grid as shown in Figure 1-14 and Figure 1-15. The servo motor is mounted to the grid fin via a shaft, either directly onto the shaft or geared to the shaft as depicted in Figure 2-7. Attached to the shaft is a hinge, which has a torsional spring on the hinge pin. When the grid fin is stowed on ascent, the spring is to be under tension such that the grid fin is pressed against the passive latch.

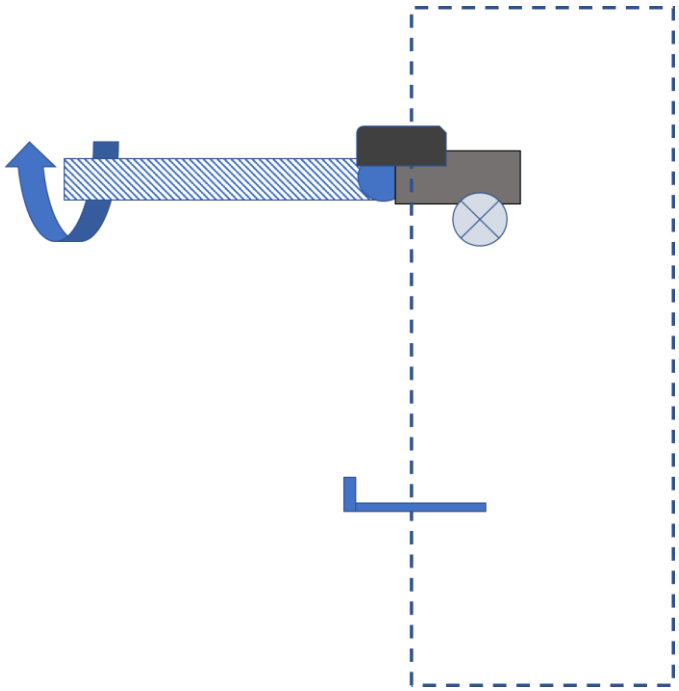


Figure 2-8 Single Grid Fin Overview, Armed

Deployment is triggered by commanding the servo motor to turn slightly out of the way of the passive latch. When the grid fin is out of the way of the passive latch, the hinge moment of the torsional spring will force the grid fins to swing open into the armed position shown in Figure 2-8. The rotation arrow is shown to illustrate that the grid fin is capable of rotating about its long axis in the plane of the page, which is the motion it makes during actuation of the grid fins.

The actual shape of the individual grid fin is a complex design that relies on the tradeoffs of different grid densities and web thicknesses as discussed in Section 1.1.1.5 The dimensions of the grid fin overall, the size of the grid and the thickness of the webs of the grid fin were based off of the G13 and G16 styles characterized by P. Theerthamalai [21]. The ARS subteam designed the grid fin geometry in SOLIDWORKS shown in Figure 2-9, with dimensions shown in inches in Figure 2-10 to Figure 2-12.

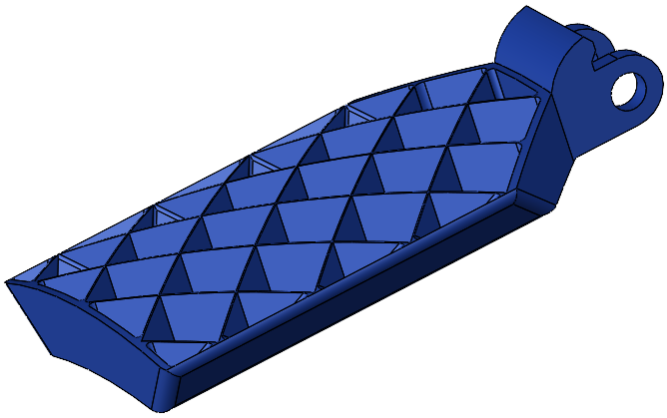


Figure 2-9 Grid Fin Design

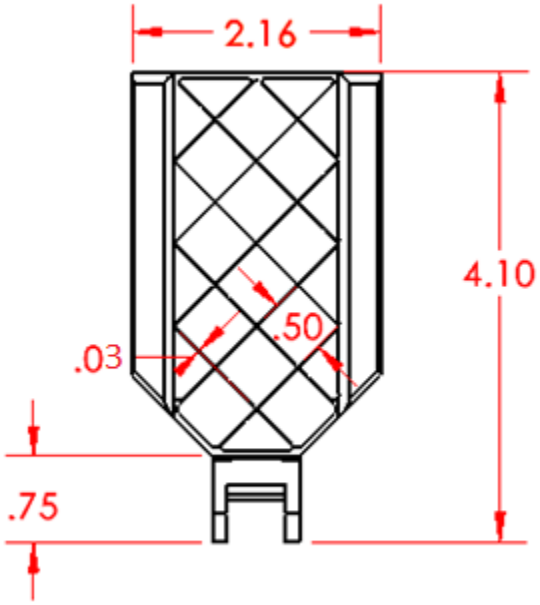


Figure 2-10 Grid Fin Dimensions in Inches, Bottom View

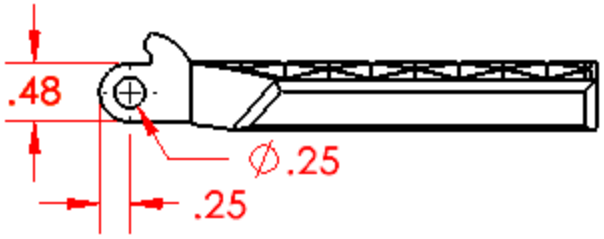


Figure 2-11 Grid Fin Dimensions in Inches, Side View

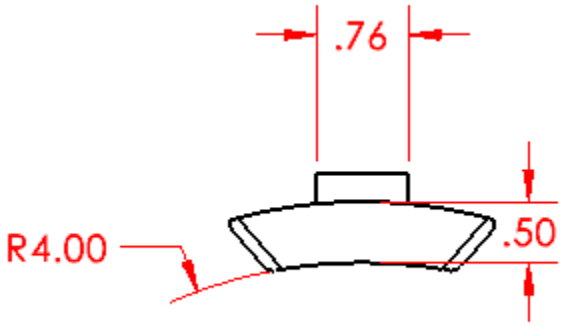


Figure 2-12 Grid Fin Dimensions in Inches, Front View

The geometry we settled on was informed by the literature review, but was modified through multiple iterations of design. When laying out the framework for the computation fluid dynamics (CFD) simulations discussed in Section 2.2.4, it became clear that the grid fin would need to produce a higher drag force on descent and have a lower drag force on ascent. The grid fin geometry was modified such that it is curved to twice the radius of the rocket body to sit more flush during ascent. The sides of the grid fin are canted inwards such that the windward projected area of the grid fin when deployed was increased by approximatley 15%, which would create more drag.

The grid fins are attached to servomotors (used to control each fin's deflection angle) by a custom shaft. The shaft acts as the connection to the motor as well as the hinge for spring deployment. The grid fin shaft is connected directly to the servomotor 25-tooth spline output shaft. The grid fin shaft was 3D printed using the Ultimaker 3 printer, as it was not as complicated

of a part as the grid fins. The Ultimater 3 prints in Polylactic Acid (PLA). The servomotor chosen by the ARS subteam is the HiTec D89MW microservo [70] shown in Figure 2-13. This servo is the smallest motor available that has a high enough torque range to successfully turn the grid fins.

Figure 2-13 HiTec D89MW microservo © HiTec 2020

With dimensions of 1.14 x 0.51 x 1.18 in. (29.0 x 13.0 x 30.0 cm), this servo is small enough to fit four of them inside the 4 in. airframe diameter, while still being able to rotate the grid fins with an operating torque range of 4.625 – 7.375 lbf-in (0.523 – 0.833 N-m). The servos are fastened directly to the bulkheads and a top bracket which slides over all of the servos and into the recessed seating areas in the bulkhead, as shown in the exploded view in Figure 2-14.

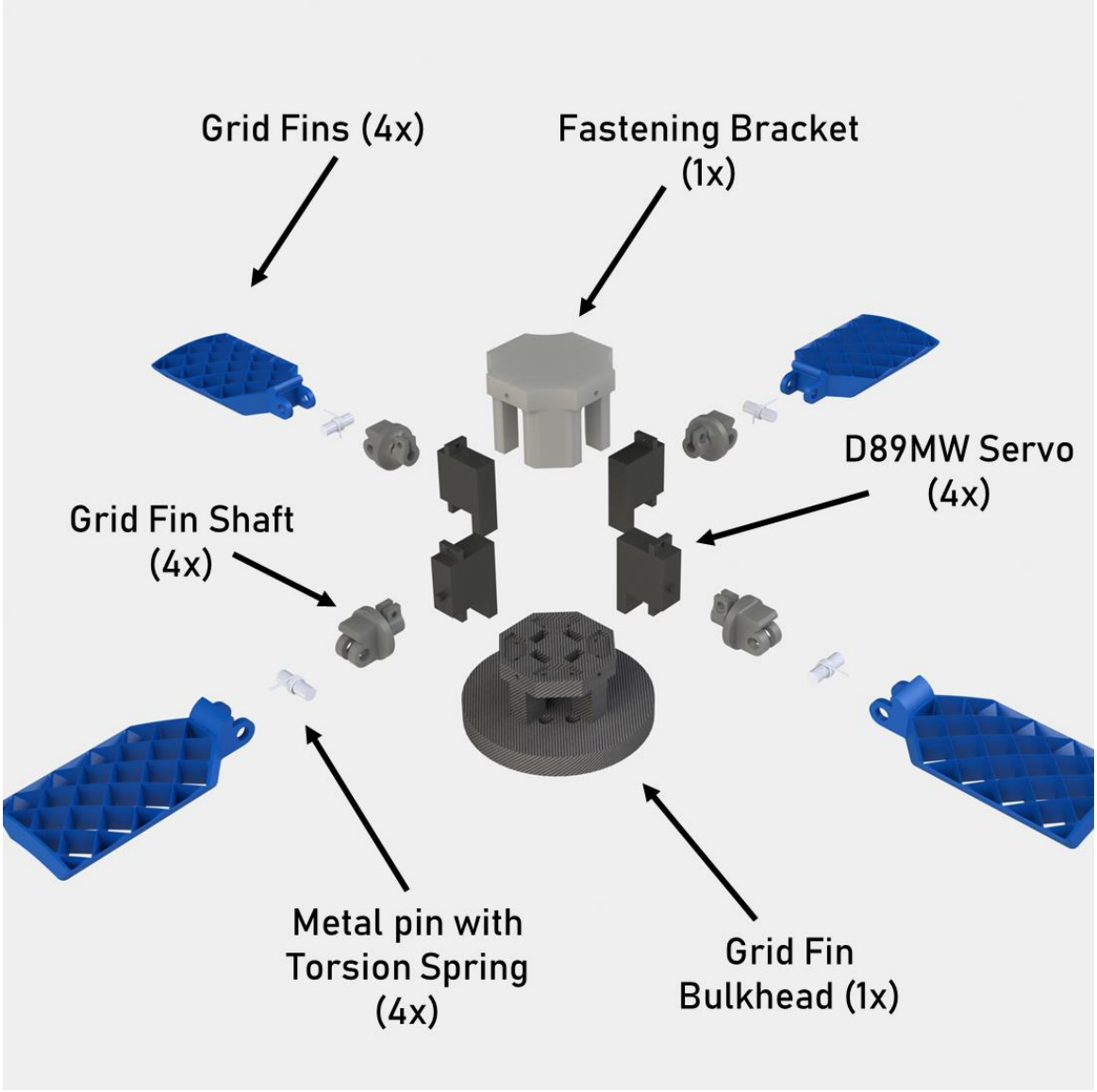


Figure 2-14 Exploded View, Grid Fin System

The grid fins were constructed using a 3D printer. The WPI Formlabs Form 2 printer was selected, as it had the resolution needed to print the grid walls. The material was chosen based on the recommendation of WPI's Rapid Prototyping Lab Director, Dr. Erica Stults, who believed the Formlabs "Tough Resin" material would stand the best chance of printing a complicated part successfully. "Tough Resin" simulates the strength and stiffness of polypropylene [71]. The grid fin system prototype was fully assembled, as shown in Figure 2-15. The grid fins were connected to an Arduino Teensy microcontroller [72] with power to the servos and a serial communication

over USB to a computer that sends commands to the Arduino to command the servos. The wiring diagram for servo power in launch configuration is shown in in Figure 4-2.

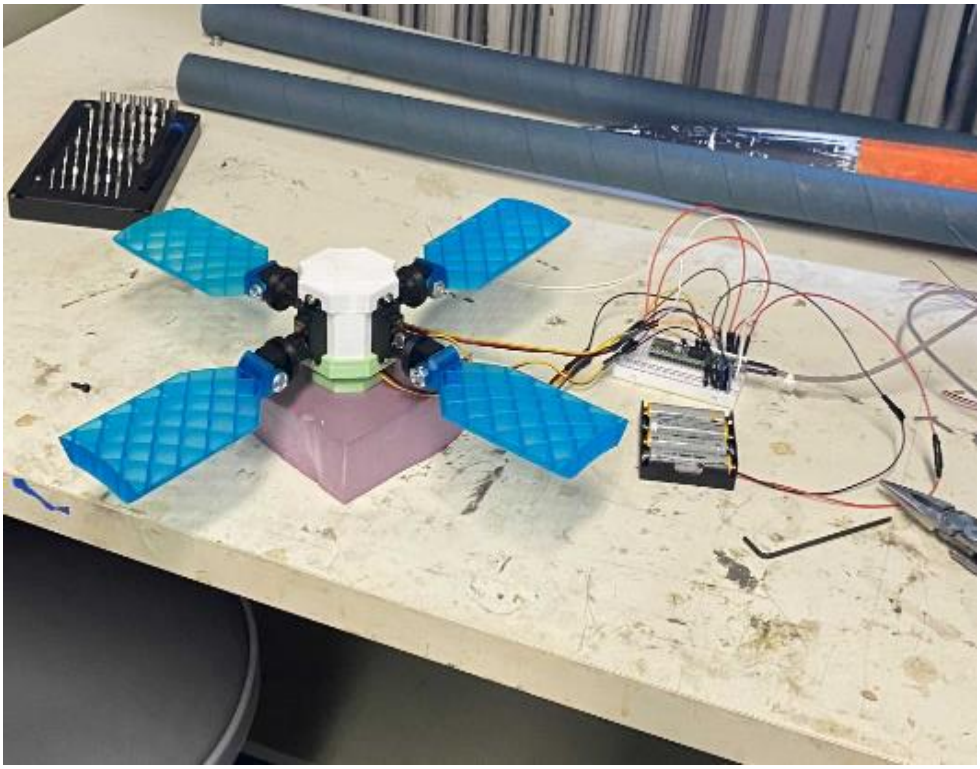


Figure 2-15 Grid Fin System, Assembled

2.1.3 Flight Camera

In order to view the flight, specifically the grid fin actuation, a small camera was to be placed above the grid fins on the outside of the airframe. The camera was chosen to be as small as possible, while also being battery operated and have a wide viewing angle. The ARS subteam selected the Sir Gawain G007 mini camera [73] shown in Figure 2-16, with dimensions of 0.87x0.87x0.87 in. (2.21 x 2.21 x 2.21 cm) and a wide-angle viewing lens of 140°.

Figure 2-16 Sir Gawain G007 mini camera © Sir Gawain 2020

A small fairing was designed to hold the camera in place, as well as reduce its drag during ascent. The camera fairing is epoxied directly to the airframe and features a radius at the base for a clean fit against the Blue Tube. The camera is epoxied to the fairing on two sides for a strong connection while leaving the micro-SD card port, mini-USB port, and power button exposed. The camera fit is shown in Figure 2-17, while the fairing dimensions is shown in Figure 2-18.

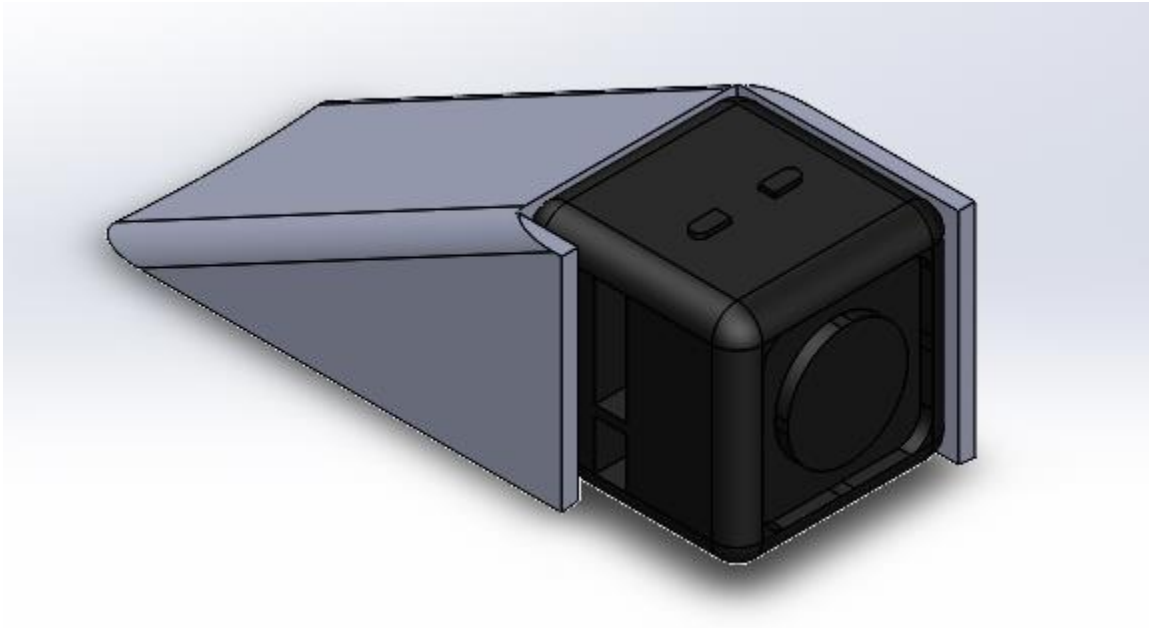


Figure 2-17 Camera and Fairing Assembly

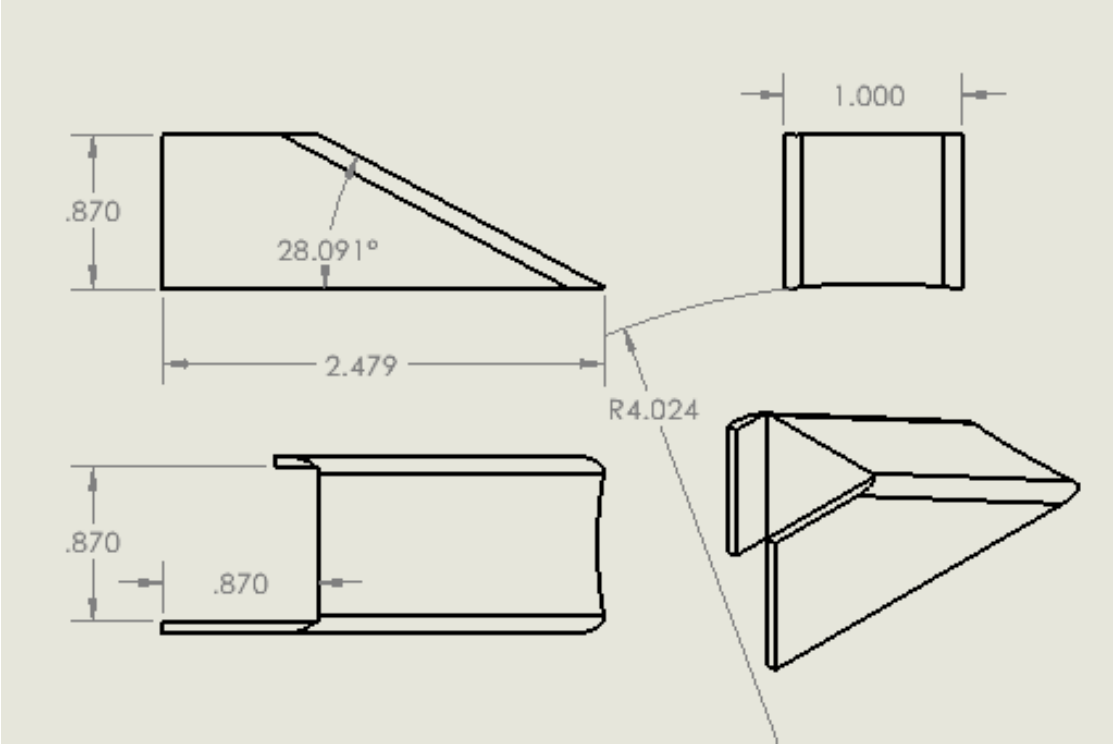


Figure 2-18 Camera Fairing Dimensions (in inches)

2.1.4 Nosecone

Using the variety of nosecone shapes discussed in Section 1.1.1.4, The nosecone of the rocket was decided between multiple options of COTS high-powered model rocket nosecones, and a custom-made 3D printed nosecone. The custom nosecone designs were all relatively large and would have had to be 3D printed in separate pieces so that they can fit inside a standard 3D printing stage. Due to complexity and cost, we decided that a COTS nosecone was the best option for the Project Andromeda rocket.

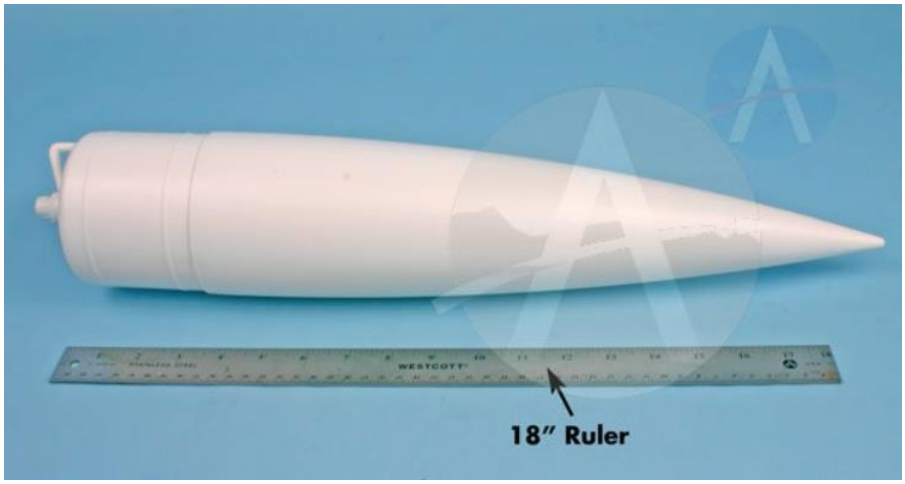


Figure 2-19 Selected COTS Nosecone [74] © Apogee Components 2020

The Madcow Rocketry 20148 nosecone was selected for its availability for purchase and its smaller size and mass relative to other choices [74]. The nosecone will have a small parachute that will attach to it past its shoulder and will be ejected with a black powder charge. The fit in the rocket can a friction fit, where it is sanded to alter the fit. Instead, the nosecone will be attached to the rocket with shear pins which can be sheared and separated with a black powder charge as discussed in Section 3.1.6.

2.2 Analysis

2.2.1 Finite Element Analysis with Ansys Airframe Stress Distribution

Throughout Project Andromeda, Ansys Workbench (and therefore Finite Element Analysis – FEA) was used for multiple analysis tasks. For the analysis tasks that involved finding the stress and deformation of both the grid fins and airframe, the deformation and equivalent stress features of Workbench were used. In order to get the results in terms of deformation and stress a pressure load must be applied to the part.

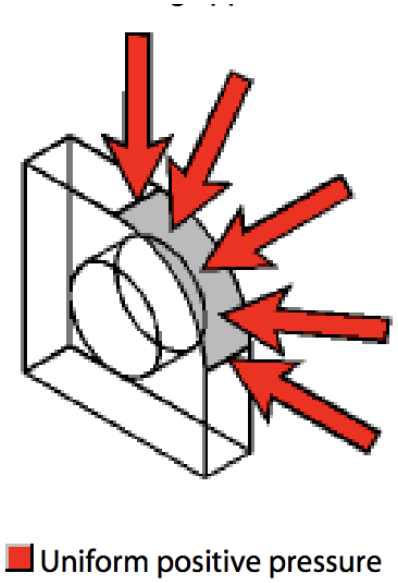


Figure 2-20: Workbench Pressure Load example © Ansys 2020

As can be seen in Figure 2-20, the pressure load is applied normally and uniformly to the surface that is selected. Once a pressure load is applied, a solution can be generated. For our project, we were able to export these pressure loads from Fluent into Workbench.

The first calculation performed was an evaluation of deformation. This can be modeled throughout an entire part and is modeled with the formula in and referenced in Figure 2-21. In this equation Ux, Uy, and Uz each represent the deformation in that given direction, and U represents the total deformation

$$U = \sqrt{U_x^2 - U_y^2 - U_z^2} \qquad 12$$

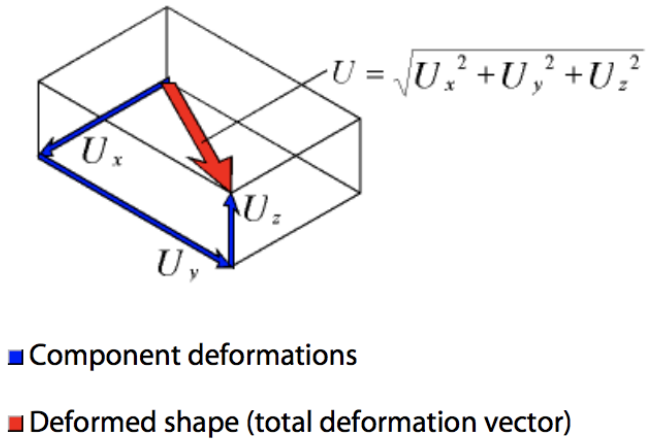


Figure 2-21: Workbench Deformation example © Ansys 2020

Figure 2-21, shows that deformation is calculated relative to the part's coordinate system for more accurate results. The deformation tool takes each component deformation from this coordinate system in order to create a total deformation vector.

The next calculation was equivalent stress, also known as (*von Mises* stress). Equivalent stress is useful to use because it can show the stresses from a three-axis system as one stress that is easier to understand [75]. Equivalent stress is calculated using the principal stresses with the formula in Eq. 13.

$$\sigma_e = [\frac{(\sigma_1 - \sigma_2)^2 + (\sigma_2 - \sigma_3)^2 + (\sigma_3 - \sigma_1)^2}{2}]^{1/2} \quad 13$$

Where each sigma represents a principal stress in X,Y,Z directions respectively. Equivalent stress is mostly used to compare with the yield strength of a material, which is important in this project due to the high forces and pressures that are applied. Specifically, in this project it was used to see if the grid fins could withstand the stress at terminal velocity, and if the airframe and bulkheads could withstand the stress at maximum acceleration.

2.2.2 Computational Fluid Dynamics Analysis with Ansys

Various analysis tasks for Project Andromeda require the use of Computational Fluid Dynamics (CFD) to simulate the flow of fluid numerically, due to the complex geometries and systems involved with the flight of a rocket. Ansys Workbench contains various fluid flow solvers for many different applications. For the purposes of this project, Ansys Fluent was chosen as the Flow Solver CFD program. Fluent allows the choice of numerical method between a pressure-based solver or a density-based solver. Pressure-based solvers are better suited for low-speed incompressible flows than density-based solvers, which are primarily used for high-speed compressible flows [76]. Flows that have a low Mach number $M < 0.5$ have a very small change in density from $M=0$, and compressibility can be ignored [77]. Simulations made using OpenRocket, discussed in Section 2.1.1, showed that the rocket would not reach M = 0.5. So, for the purposes of this project air is considered incompressible, and simulations will utilize the pressure-based solver for fluid flow. Ansys Fluent's pressure-based solver uses either segregated or coupled algorithms for solving governing equations, which are outlined in Figure 2-22.

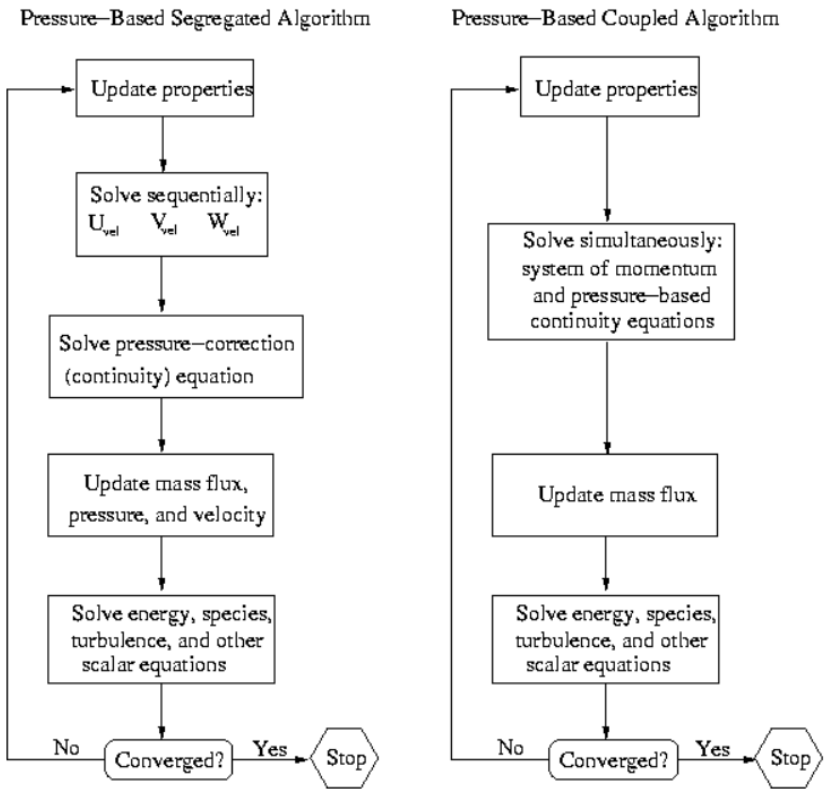


Figure 2-22 Overview of the Pressure-Based Method [76] © Ansys 2020

The coupled algorithm iterates through the governing equations used, which depend on the model being used, as a system of coupled equations, continually iterating and resolving the equations until the solution is converged. Coupling the governing equations, generally pressure and velocity, is less memory efficient but is more robust and increases the rate of solution convergence. The coupled algorithm method was used for this project. The algorithm is considered converged if the residuals of all of the solver variables are less than the user or program specified threshold. A residual is the fractional error of a variable between iterations, defined as the ratio of the change in value of a variable between iterations to the current value [76]. A user can manually set the threshold for the residual convergence value, which is 0.001 by default for most solution variables, depending on the chosen model. Figure 2-23 below shows residuals of solution variables converging with increasing iteration.

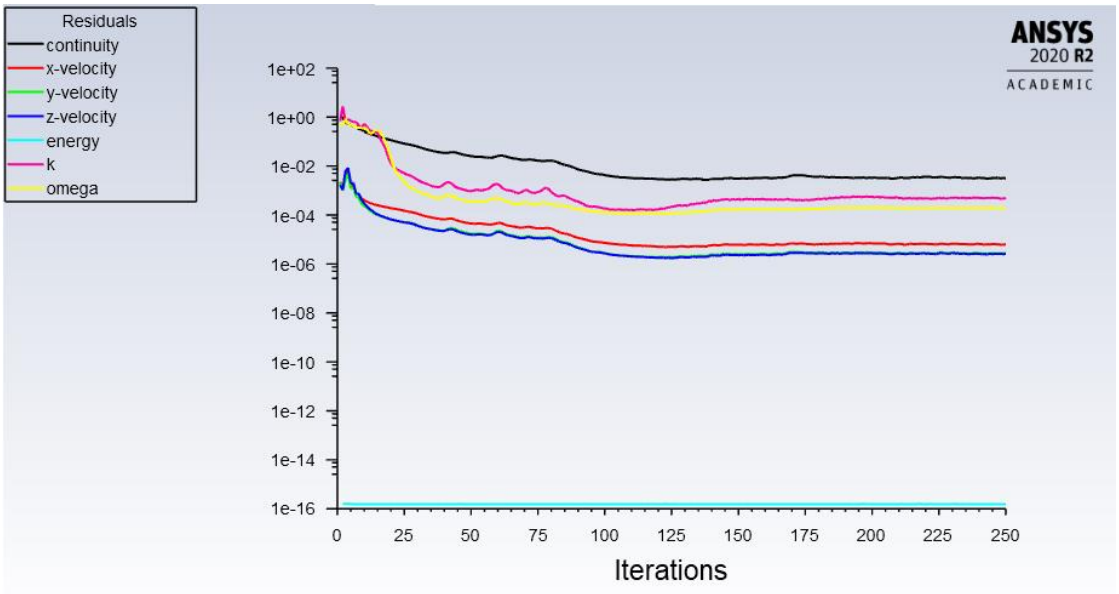


Figure 2-23 Plotted Residuals in Fluent

There are a multitude of pressure-based models of fluid flow that Fluent can use. The models differ in their assumptions and solver terms. For higher accuracy, models that can calculate turbulent viscosity can be used. The Spalart-Allmares Model [76] was considered first for this work, as it is a relatively simple, single-equation model designed specifically for aerospace applications with wall-bounded flows, such as those across a complex geometry surface like grid fins or the rocket fuselage. However, after some experimentation, the Spalart-Allmares model

generated anomalous results with non-converging residuals in complex and finely detailed meshes used on several subsystems of the rocket. This may be due to the fact that one-equation models are notorious for being unable to rapidly shift between length scales where the flow changes from wall-bounded to free shear flow [78]. The k-ε and k-ω models are also two commonly used turbulent viscosity models. The k-ε model has accurate free-stream independence and is apt for calculating flow not closely bound by walls, called far field flow. The k-ω more accurately models wall-bounded flow but isn't as accurate for flow in the far field. The Shear-Stress Transport (SST) k-ω model blends both k-ε and k-ω models, for a more all-around robust model that is capable of a wide variety of flows that remains accurate in near-wall and far-field flows, so it was the best choice for this project [78]. The model is based on the turbulent kinetic energy term, k, and the specific dissipation rate, ω, which are found from the model's transport equations. The equations are described in detail in the Ansys Fluent user guide [78].

In order to simulate flow around a specific 3D geometry, a 3D CAD file of the geometry must be imported into Ansys Fluent. Since Fluent is solving for the conditions of a fluid, the solid geometry must be cut out of a flow domain in which the flow will be simulated. Physically, this means that the solid geometry in question must be surrounded by a flow domain geometry, and then the solid geometry cuts an empty area in the flow domain using a Boolean function. This empty area is where the program is told not to simulate fluids, and the faces of this empty area must be defined as wall type boundary conditions. Wall type boundaries tell Fluent to solve for the no-slip and near wall conditions in these areas. With this set up properly, Fluent cannot solve for fluid flow within the empty space. The specific shape and setup of the fluid domain is heavily dependent on the problem being solved. See Section 2.2.4 and Section 4.2.2 for a discussion of the setup of Fluent domains in specific scenarios.

After these fluid domain geometries are prepared, they must be meshed into fine cells or finite elements, similar to what is used in finite element analysis (FEA). These elements are then recognized by Fluent as control volumes to which the equations of the fluid model are applied in each solver iteration. Meshing is often the most complicated process in preparing a CFD simulation for analysis because it requires taking into account many details such as layer thickness of the cells, mesh detail near walls, mesh detail at faces, and mesh detail in the far field. The meshing

strategy for a particular problem is often highly specific to the scenario being simulated and requires iteration to produce a fruitful result [78].

Defining boundary conditions, as mentioned with walls, is extremely important for running the Fluent simulation. Without correctly defining boundary conditions, the solver may not be able to initialize the calculation or produce inaccurate results. Fluent offers a large variety of boundary condition types, which are heavily customizable and establish the flow conditions for the simulation to be run. The boundary conditions that are applicable to the simulations used for analysis in Project Andromeda are summarized in Table 2-1 Fluent Boundary Conditions, Summary [75].

Table 2-1 Fluent Boundary Conditions, Summary [75]

Boundary Condition	Description
Pressure Inlet	Used to define fluid pressure at flow inlets
Velocity Inlet	Used to define the flow velocity, along with all relevant scalar properties of the flow, at flow inlets
Mass-Flow Inlet	Used to provide a prescribed mass flow rate or mass flux distribution at an inlet
Pressure Far Field	Pressure far-field conditions are used to model a free-stream condition at infinity, with free-stream Mach number and static conditions being specified
Pressure Outlet	Pressure outlet boundary conditions require the specification of a static (gauge) pressure at the outlet boundary

Wall boundary	Wall boundary conditions are used to bound fluid and solid regions. In viscous flows, the no-slip boundary condition is enforced at walls by default

For each of the boundary conditions used in Fluent simulation, various relevant scalar properties of the flow can be specified. In order to remain consistent, the Fluent simulations run for all subsystems of the Project Andromeda rocket share the same basic set of assumptions listed in Table 2-2.

Table 2-2 Fluent Simulation Assumptions

Assumption	Associated Properties/Settings
Constant density (incompressible) air flow at sea level density	Fluid Material = air $\rho = 1.225$ kg/m^3 = 0.076474 lbm/ft^3
Gauge pressure at pressure outlet is zero	P = 0 atm
Simple wall conditions	Wall Boundary Condition (default): Stationary Walls, No-slip condition, Standard Roughness

2.2.3 Airframe Stress Distribution (Analysis Task 1)

The first analysis task for the Airframe and Recovery Systems subteam was to identify critical locations of high stress throughout the airframe and internal structures during a high stress maneuver. Since the objective of the Project Andromeda rocket is in part to demonstrate propulsive landing, a landing impact was chosen as the scenario to simulate. In order to obtain a stress distribution on impact using Ansys Explicit Dynamics in Workbench, ARS decided to focus on the airframe and bulkhead stress.

A model of the rocket with all internal systems removed, except for the bulkheads and airframe was used. As was mentioned in Section 2.1, ARS was responsible for designing the bulkheads that secure the grid fins, cold gas thruster, and stage separation systems. We decided that the use of a composite material that could be 3D printed would be much more effective than using aluminum. In terms of choosing a composite material, carbon was the obvious choice due to its strength, as can be seen in Figure 2-24.

Fiber Reinforcement	Test (ASTM)	Carbon	Kevlar®	Fiberglass	HSHT FG
Tensile Strength (MPa)	D3039	800	610	590	600
Tensile Modulus (GPa)	D3039	60	27	21	21
Tensile Strain at Break (%)	D3039	1.5	2.7	3.8	3.9
Flexural Strength (MPa)	D790[1]	540	240	200	420
Flexural Modulus (GPa)	D790[1]	51	26	22	21
Flexural Strain at Break (%)	D790[1]	1.2	2.1	1.1	2.2
Compressive Strength (MPa)	D6641	320	97	140	192
Compressive Modulus (MPa)	D6641	54	28	21	21
Compressive Strain at Break (%)	D6641	0.7	1.5	—	—
Heat Deflection Temp (°C)	D648 B	105	105	105	150
Izod Impact - notched (J/m)	D256-10 A	960	2000	2600	3100
Density (g/cm³)	—	1.4	1.2	1.5	1.5

Figure 2-24 Composite Data Sheet © Markforged 2020

Figure 2-24 shows that carbon outperforms all other composites in terms of strength and is also readily available. With the bulkheads completed, it was clear that the assembly was too complex to be fully simulated without extremely powerful computation clusters and long simulation times. A simplified airframe with simplified bulkheads was set up for Ansys impact simulations, shown in Figure 2-25.

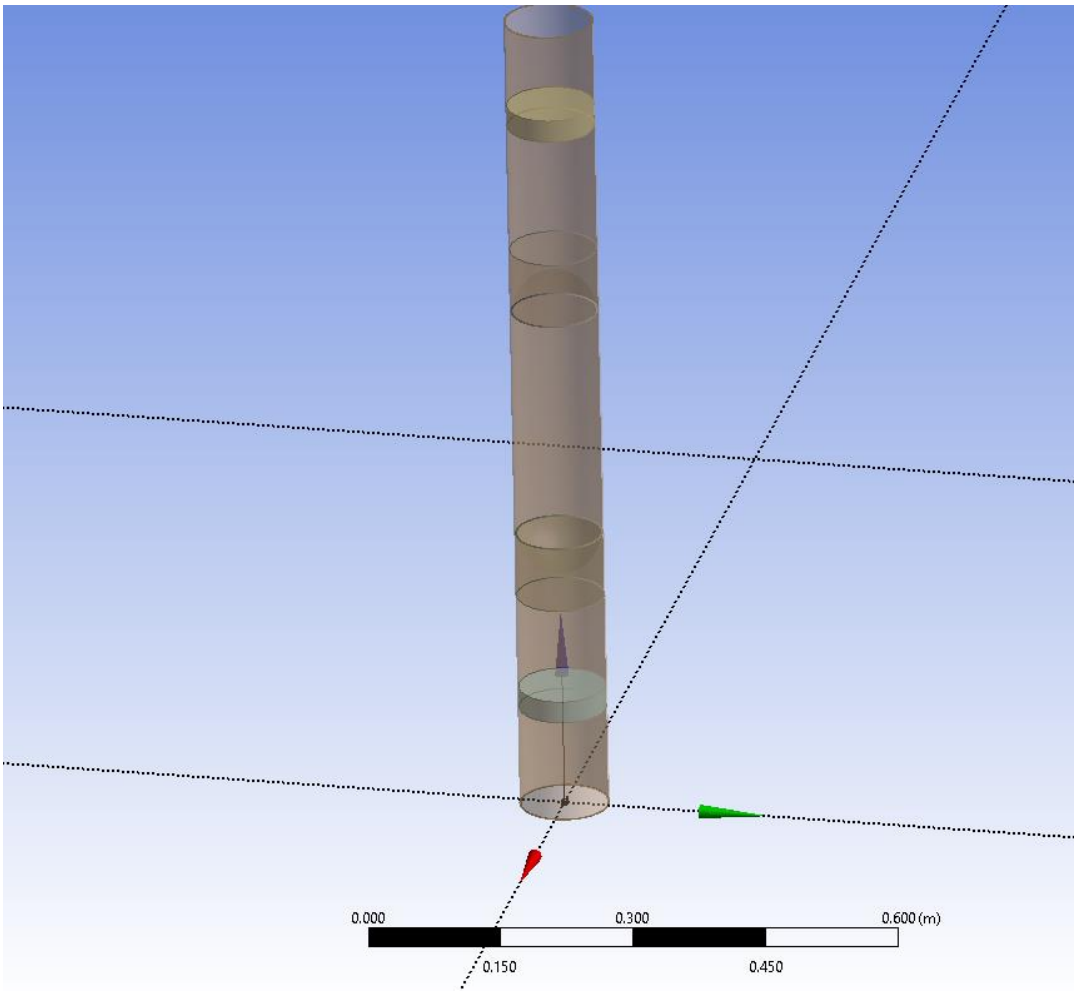


Figure 2-25 Ansys Explicit Dynamics Impact Geometry

The bulkheads were defined in Ansys as “bonded” to the interior of the airframe. In Ansys, specification of material for a part is accomplished by “assigning” a material to a specific part when the simulation is set up. The airframe material assigned was epoxy fiberglass, which was the most similar material to BlueTube available in the Ansys basic material library. The bulkheads were assigned a biaxial woven carbon fiber epoxy composite material. A standard universal gravity vector was defined in the downwards direction, and 3 initial condition cases of downward velocity were assigned to the rocket assembly to simulate impact: 5 m/s impact, 10 m/s impact and 58 m/s. The simplified simulation assembly is much less massive than the actual assembly since most heavy components, especially the cold gas system, are omitted which means the simulation results would underestimate the stresses experienced by the airframe. However, Ansys allows for initial

conditions to be defined, which allows specification of realistic landing velocities despite not having the correct mass for the real assembly.

First, we wanted to demonstrate the safety of landing at 5 m/s which is the safe landing velocity with a parachute or with the cold gas thruster system discussed in Section 1.1.1.5, Section 1.1.2.7, and Section 3.2.4. The results of the Ansys simulation show a peak stress on the order of 85 MPa during impact for a landing velocity of 5 m/s is shown in Figure 2-26 and Figure 2-27. This maximum stress does not exceed the ultimate stress of the airframe at any point, but it does slightly deform the airframe on the order of 1mm near a bonded area of the airframe. There is reason to suspect that there is some anomalous result from Ansys with the high stresses seen around the bulkhead, which could be arising due to an improperly defined interaction between the bulkhead and the airframe.

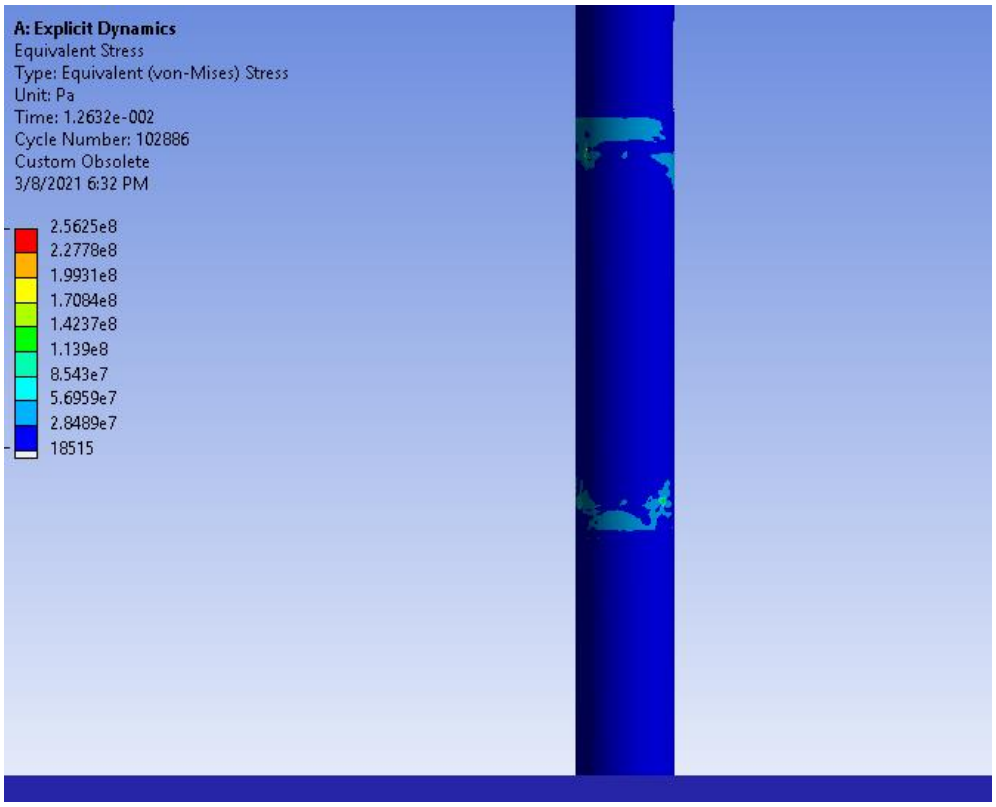


Figure 2-26 von-Mises Stress, 5m/s Impact Landing

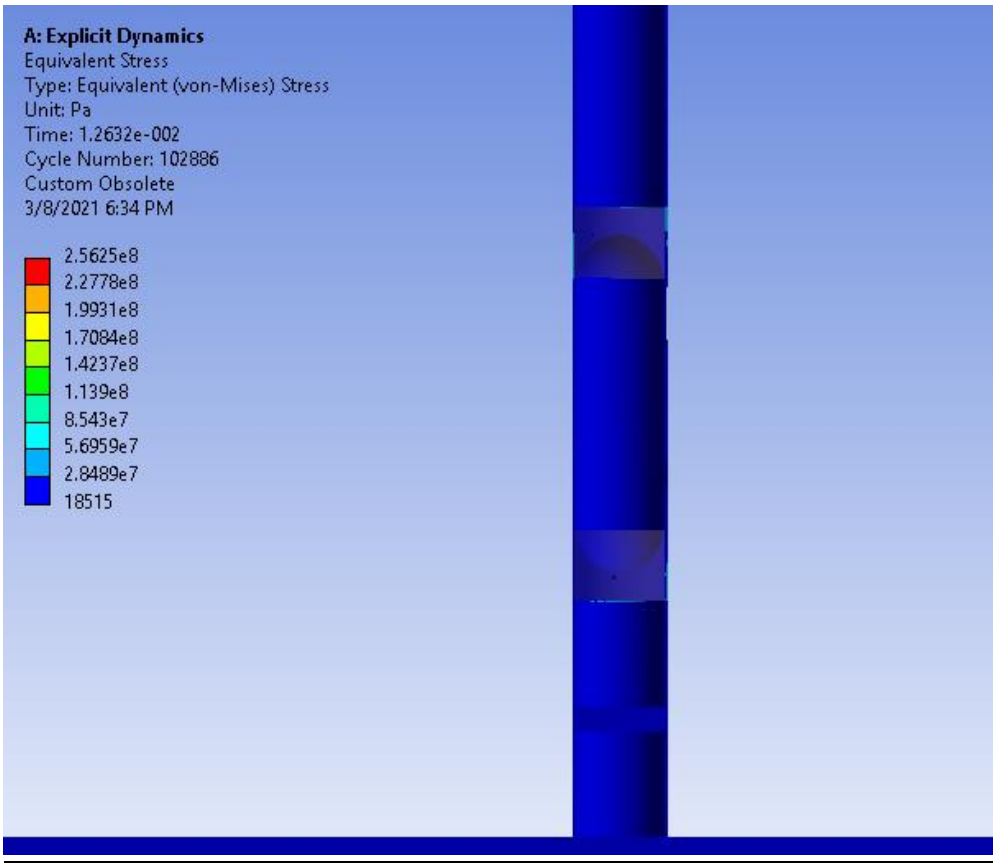


Figure 2-27 von-Mises Stress, 5m/s Impact Landing, Cross Section

This impact simulation also shows that the cold gas system would be safe in a 5m/s landing, with zero plastic deformation around the tank and no part of the bulkheads securing the tank experience yield stress.

The next simulated scenario is a higher 10m/s impact velocity that would represent a partial failure of the cold gas system, either firing too late or too early, resulting in a higher landing velocity. In this scenario, the simulations showed that there would be extreme deformation experienced by the lower part of the airframe below the bulkheads, shown in Figure 2-28 and Figure 2-29. The deformation occurs around the bulkheads in the very thin airframe and would likely result in interference with the internal structure of the stage separation subsystem. The lower section of airframe that experiences this damage would not be reusable. However, the peak stress still does not exceed ultimate stress of the Blue Tube, and the bulkheads themselves have no damage or plastic deformation. Furthermore, the cold gas system is not likely to be deformed or

impinged on at all by the damaged airframe. The section of airframe protected by the cold gas section bulkheads was entirely undamaged. All stress was diverted away from that section of the airframe and absorbed by the lower airframe. This landing likely would break some components, but most components of the rocket would survive and be reusable after a 10m/s landing.

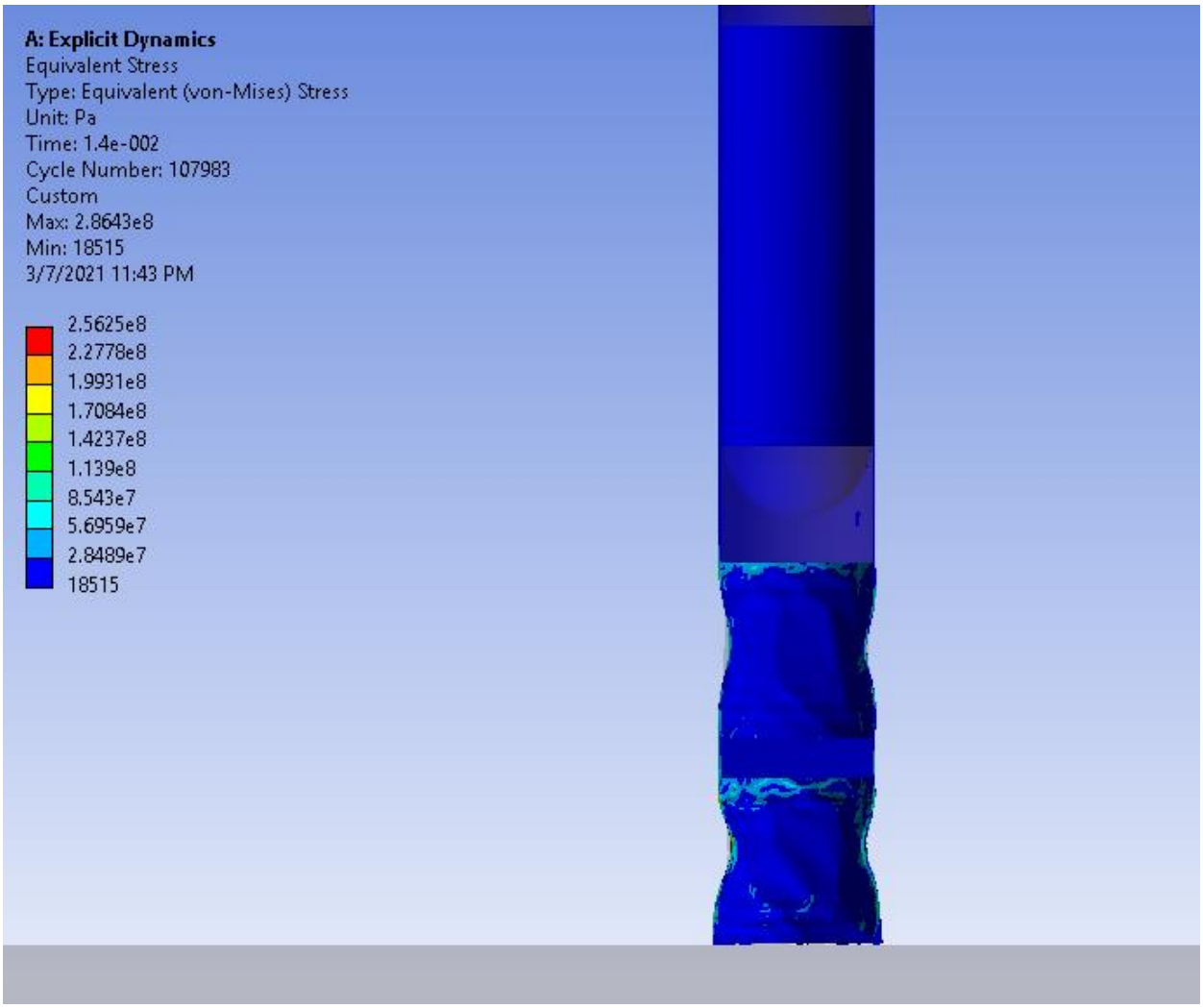


Figure 2-28 von-Mises Stress, 10m/s Impact Landing, Cross Section

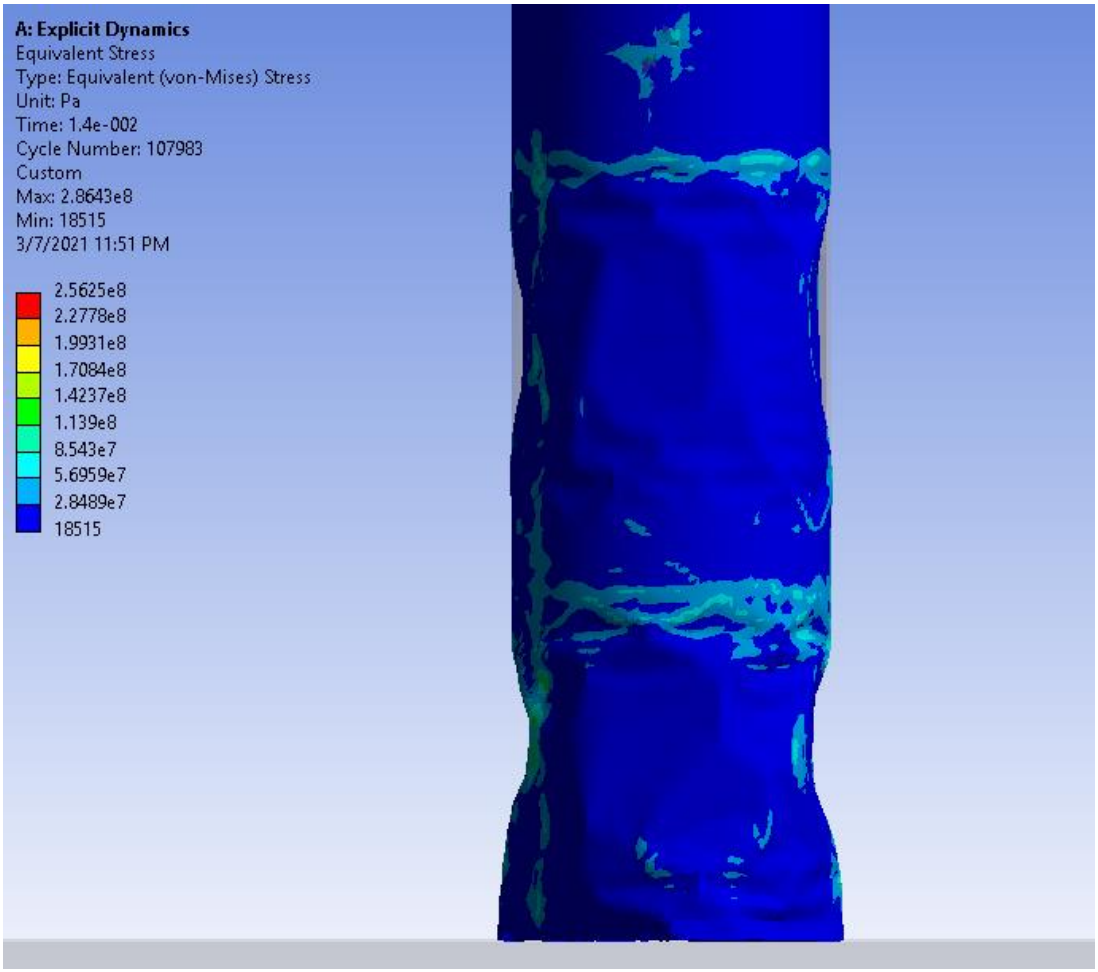


Figure 2-29 von-Mises Stress, 10m/s Impact Landing

ARS attempted to simulate one final case of total propulsive landing failure with a landing velocity equal to 58 m/s, which is approximately the terminal velocity of the rocket. While running this simulation, the energy error derived from Ansys solver equations would dramatically increase and halt the simulation after many attempts. This is likely because the stresses experienced during impact are large enough that the airframe begins shattering since it is made of fiberglass, and as that begins happening the energy error in Ansys becomes too large to proceed in time iteration. Then, the solver becomes unable to resolve meaningful stress and deformation calculations, halting the simulation and reporting an error. However, Ansys still output the equivalent von-Mises strain at time of impact before erroring out, which is shown in Figure 2-30 and Figure 2-31. This is not the maximum strain experienced, as this is just a few milliseconds into impact and does not represent the final load or damage experienced by the rocket in this impact scenario. In fact, it is

clear that there is extreme deformation occurring all over the entire airframe milliseconds into the impact, and the maximum stress error was reached by Ansys. In an actual hard landing, this situation would likely result in the entire airframe being peeled apart and shattered past ultimate failure very rapidly. This is indicative that a landing at this velocity would present the very expected result of destruction of the rocket.

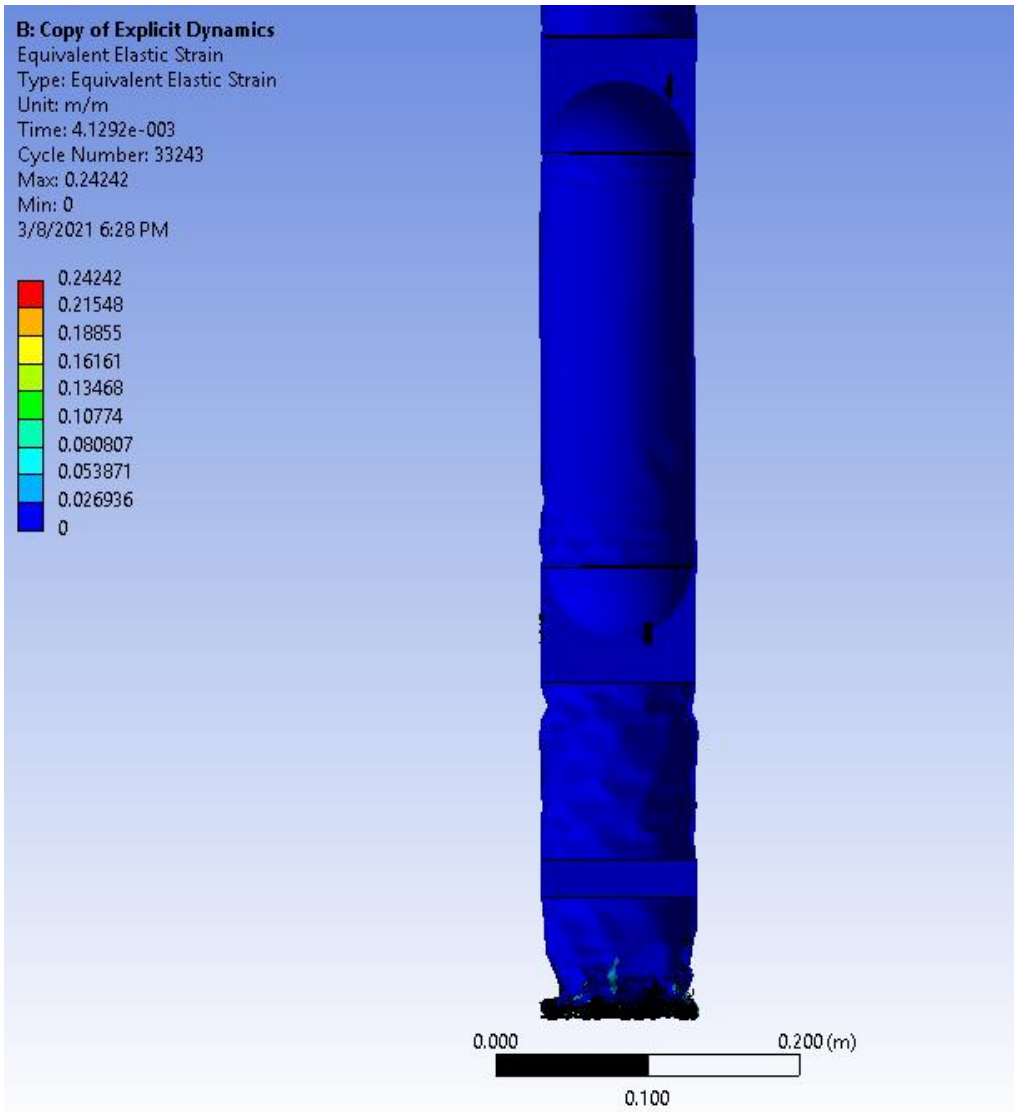


Figure 2-30 von-Mises Strain, 58m/s Impact Landing, Cross Section

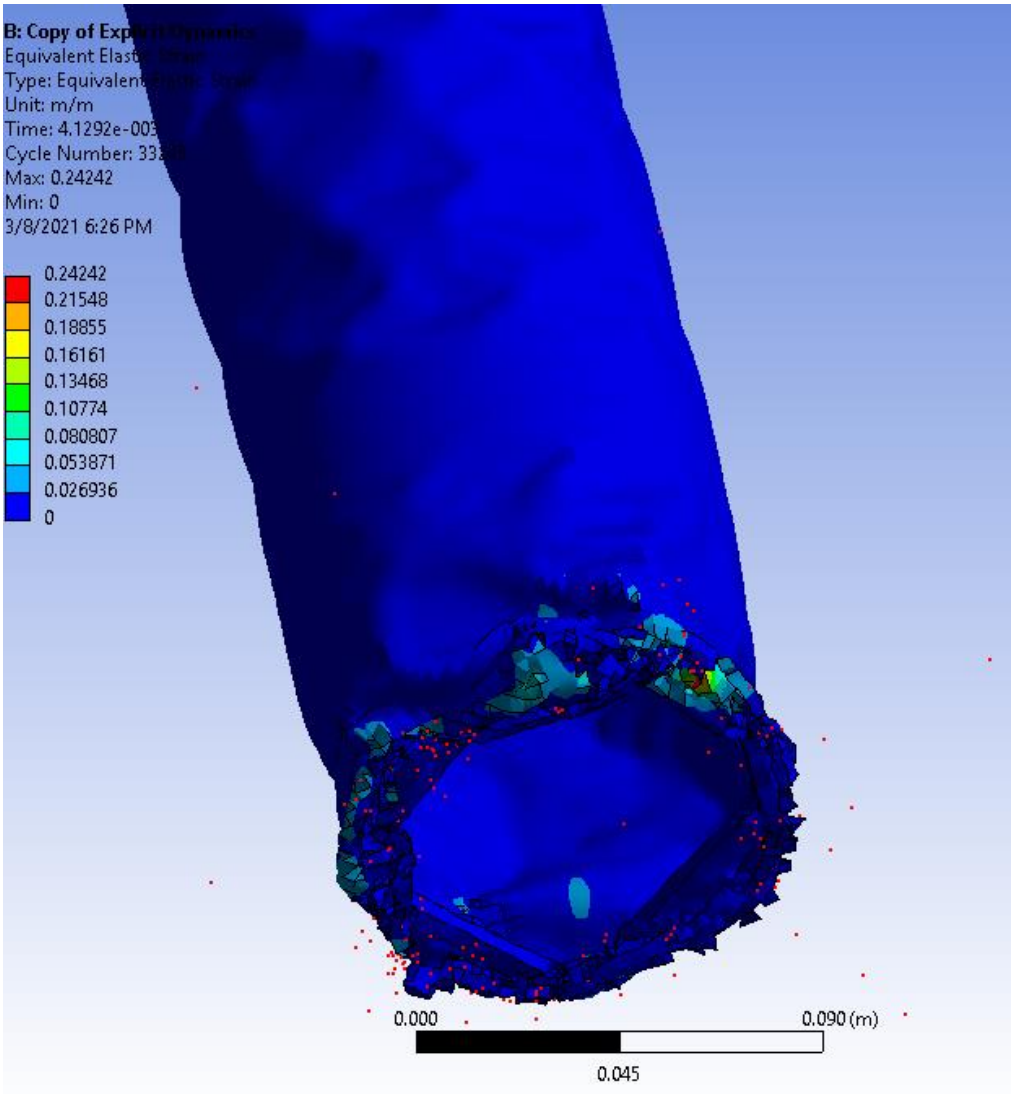


Figure 2-31 von-Mises Strain, 58m/s Impact Landing

2.2.4 Grid Fin Aerodynamic Loads (ARS Analysis Task 2)

The second analysis task for the ARS subteam was to analyze the aerodynamic loads on the grid fins. The grid fins of the Project Andromeda rocket are intended to act as drag brakes for the rocket during descent, and to stabilize the attitude of the rocket such that it is pointed vertically, just as it is in launch configuration. To produce accurate and functional control logic for actuating the grid fins and controlling the flight dynamics of the rocket on descent, the forces, and moments that the fins exert on the airframe must be understood. Figure 1-16 and Figure 1-17 depict the roll moment and pitching moment induced by the grid fin. The induced rolling moment occurs about

the rocket's central X-axis, and the induced pitching moment occurs about the center of gravity of the rocket, parallel to the Y-Z plane, which results in pitch control from the grid fins aligned with the rocket Z axis and yaw control from the grid fins aligned with the rocket Y axis. The grid fin actuates by rotating about its own central axis through a deflection angle δ, as shown below in Figure 2-32 and Figure 2-33.

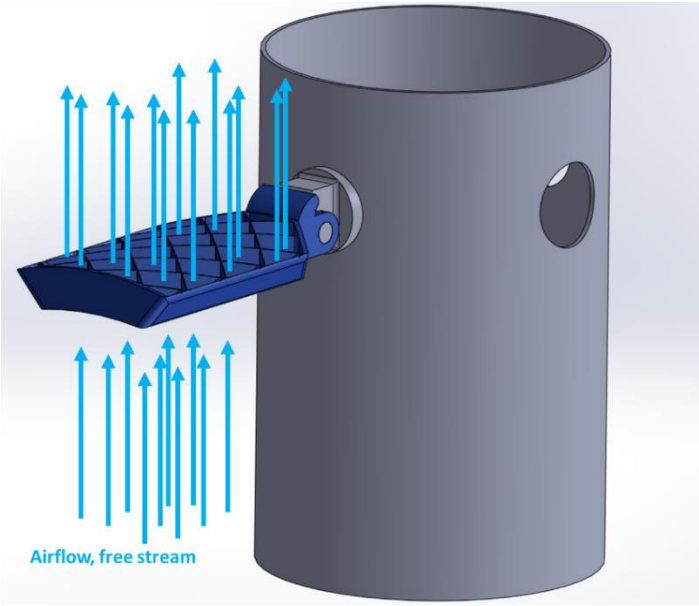


Figure 2-32 Grid Fin Airflow Diagram

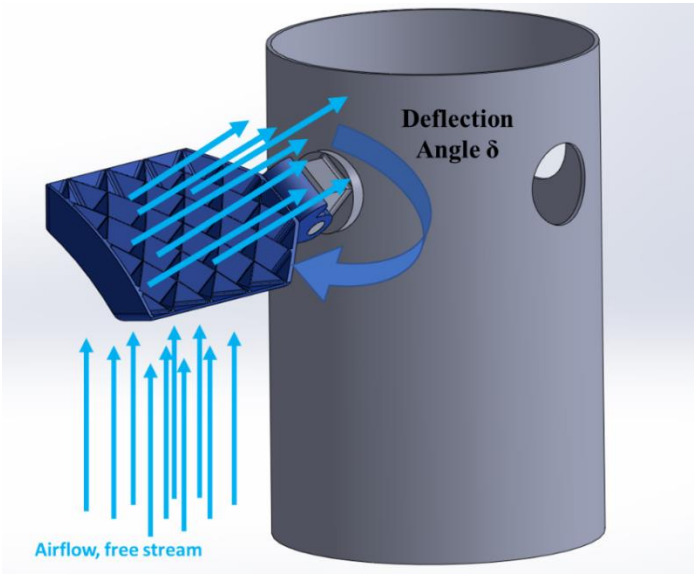


Figure 2-33 Grid Fin Airflow Diagram, Deflection Angle

Combining the rocket and the grid fin into one Fluent model becomes more difficult to mesh and run due to the complex geometry, so we only considered simulation of a single grid fin. Functionally, these fins can be treated and analyzed in Fluent a similar fashion to airfoils or wings with traditional drag and lift vectors that are fixed to the lifting surface coordinates. A diagram illustrating the grid fin lift and drag components similar to that of a wing is shown in Figure 2-34.

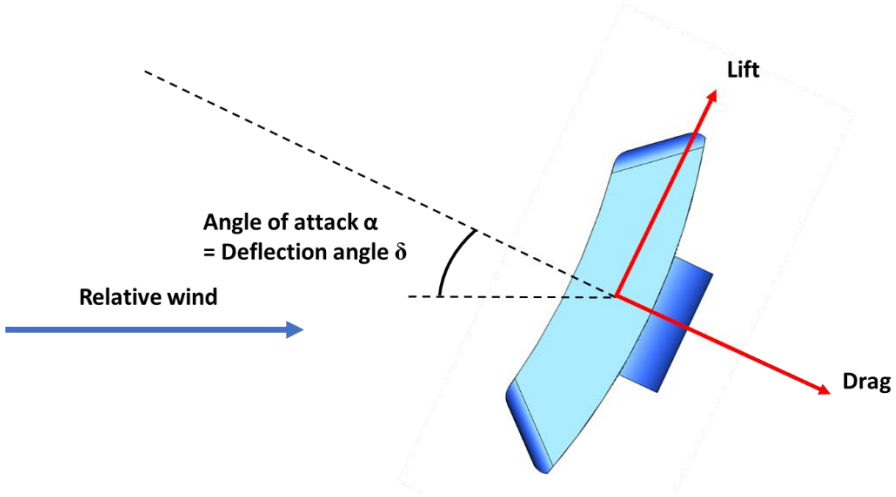


Figure 2-34 Grid Fin Lift and Drag Vectors

Because the Fluent lift and drag vector coordinates are fixed directions on the grid fin, they do not line up with the principal axes of the rocket. Therefore, the results of any simulation with this coordinate system need to be transformed to find the components of drag and lift in the principal rocket axes so that the effects of the grid fins during flight can be quantified. These assumptions require a rocket body angle of attack (AOA) of zero to evaluate the drag contribution resulting from the relative air velocity along the rocket's X-axis, as shown in Figure 2-35. The transformed drag and lift components of the grid fin in the rocket coordinates is illustrated in Figure 2-35 and Figure 2-36, with the Z-axis pointed into the page.

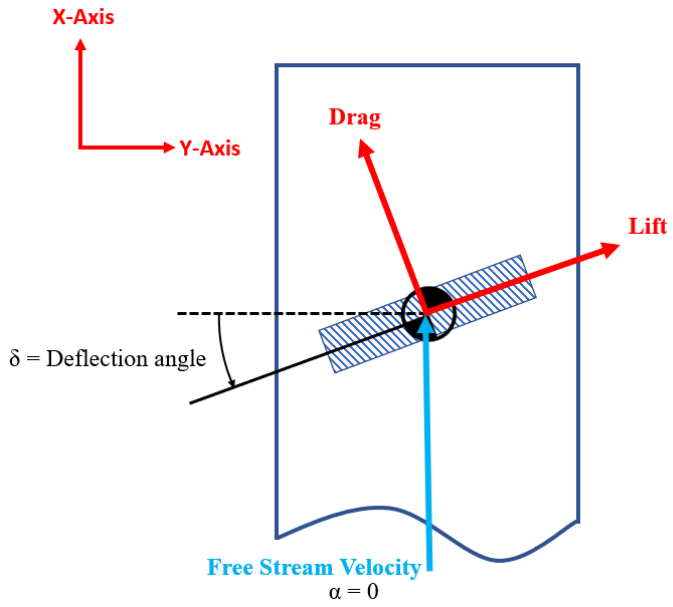


Figure 2-35 Grid Fin Components, Rocket Coordinates

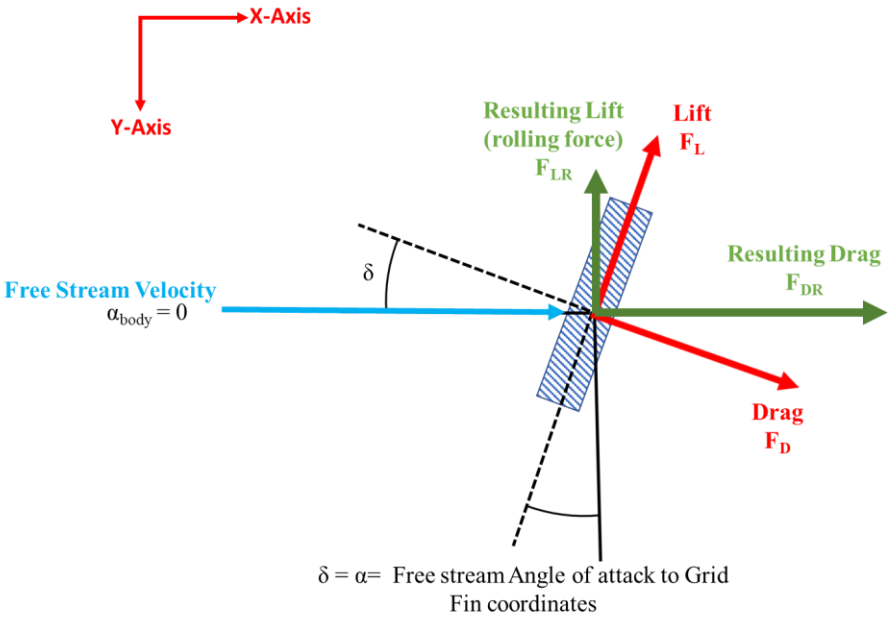


Figure 2-36 Grid Fin Resultants, Rocket Coordinates

With the AOA of the body assumed to be zero, we can consider the deflection angle δ to be equivalent to our simulation's angle α. Therefore the resulting drag and lift forces in the

coordinates of the rocket, F_{DR} and F_{LR} respectively, can be found from calculated drag and lift components F_D and F_L with Eq. 14 and Eq. 15.

$$F_{DR} = F_D\, cos(\alpha) + F_L\, cos(90 - \alpha) \quad 14$$

$$F_{LR} = F_D \sin(\alpha) - F_L \sin(90 - \alpha) \quad 15$$

With these equations, the system has been described with respect to the values that need be calculated in Fluent: F_D and F_L. The next step is to begin creating and meshing the geometry required for the simulation. Since this is the most difficult step, the best way of approaching this problem is to have one specified geometry and mesh and modify the inlet boundary conditions to vary the AOA over multiple simulation runs instead of rotating the grid fin and recreating the mesh. This can be visualized by referring to Figure 2-37.

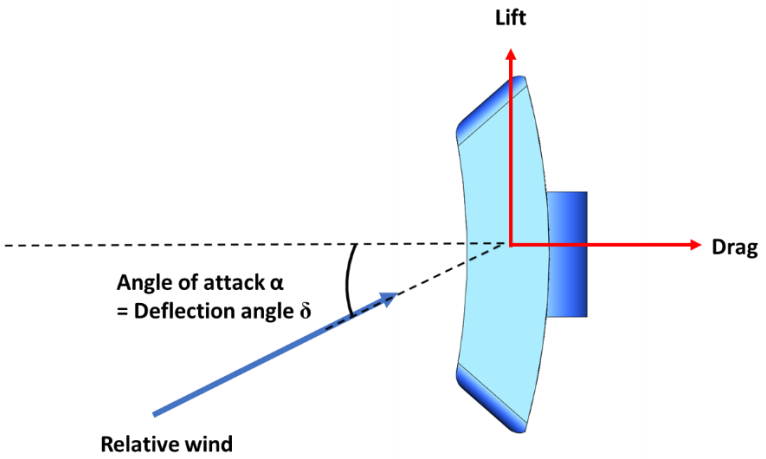


Figure 2-37 Grid Fin Fixed Geometry Approach

The grid fin has a very complex geometry, with a large range of length scales associated with it. For example, the grid fin web wall is just 0.01”, over 100 times smaller than the 4.10” length of the grid fin. As a result, the meshing process for the geometry of the simulation is very complicated. Multiple iterations of meshes and multiple meshing approaches were used when first experimenting with this simulation. We created a slightly simplified version of the grid fin geometry by reducing number of curved faces and removing the hinge geometry. Because the grid fin geometry being simulated is not exactly the same as the grid fin that will be used, the results

will not exactly reflect the grid fin aerodynamic profile in reality. However, the changes are minor, changing the projected area of the grid fin by less than 0.00001 inches^2 (6.4516E-9m^2) according to an estimate made in SOLIDWORKS. Part of the geometry simplification includes removing the hinge, but this is in the wake of the grid fin incoming velocity, and so is considered to have a negligible effect on the aerodynamics. As described in Section 2.2.2, when creating the flow or domain geometry, the grid fin solid geometry has to be cut out of a larger flow domain. In Figure 2-38, the flow domain geometry with the grid fin cut out is shown shaded on the left and wireframe view on the right.

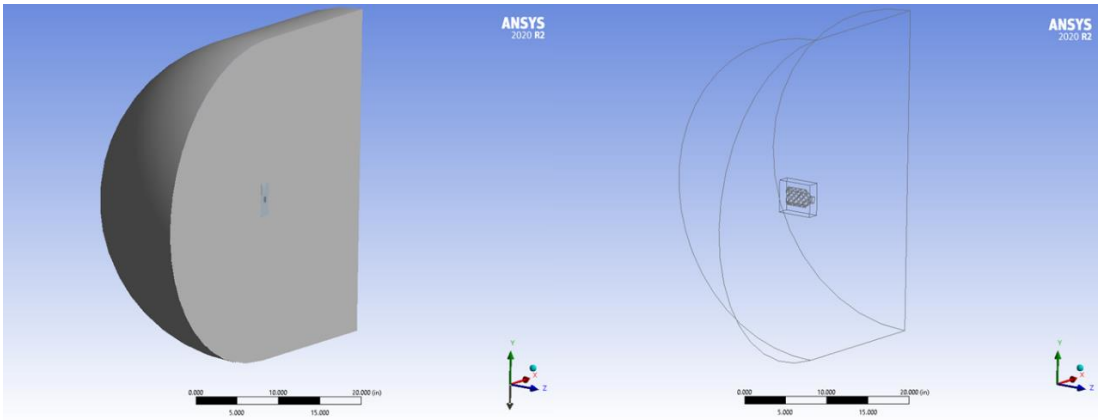


Figure 2-38 Fluent Simulation Domain Geometry, Left Shaded, Right Wireframe

The small bounding box that is surrounding the grid fin shape on the right is called the body of influence. A body of influence volume is used later when meshing to alert the meshing tool that the region within the box contains the geometric body which affects the flow, i.e. the grid fin, which needs a high density of mesh detail. The area outside of the box can be considered free stream or far field flow, and therefore doesn't need nearly as much mesh detail. Following the creation of geometry, all of the various faces of the grid fin area were named such that faces may be selected in the meshing process. Ansys allows the user to specify the sizing of the mesh on any of the faces of the simulation geometry, called face sizing. In the Ansys meshing tool, we fine-tuned the face sizing for the grid fin to appropriately match the dimensions of the grid fin, with mesh cells significantly smaller than the faces of the grid fin. Finally, an inflation layer is added. This tells the Fluent meshing tool to gradually increase the resolution of the mesh over a set number

of layers by decreasing the finite cell size in each layer as it gets closer to a defined boundary [79] [80]. In this case, the inflation layer starts at the bounds of the domain and ends at the surface of the fin, where the mesh progressively gets more detailed as it gets closer to the fin. The resulting mesh for the grid fin simulation is shown in Figure 2-39 and Figure 2-40.

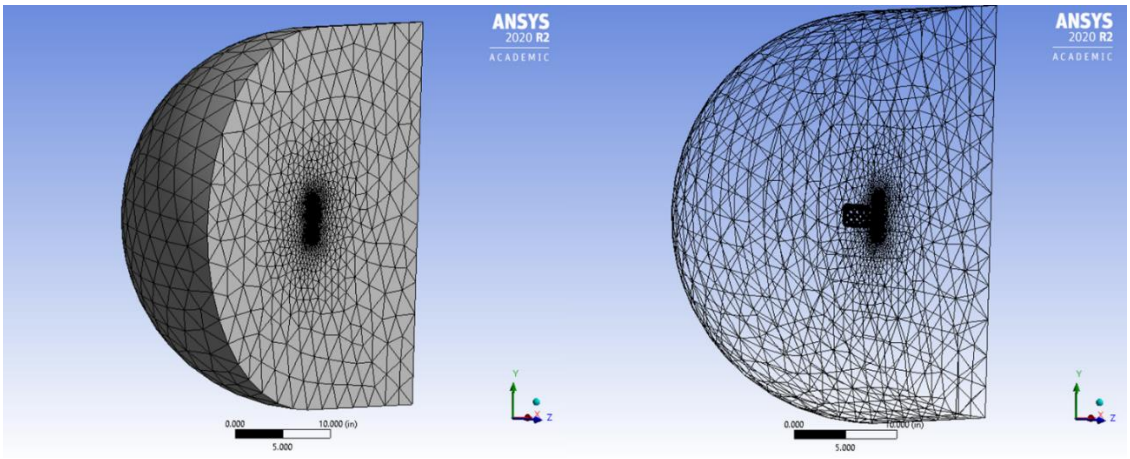


Figure 2-39 Grid Fin Domain Geometry Mesh, Overview

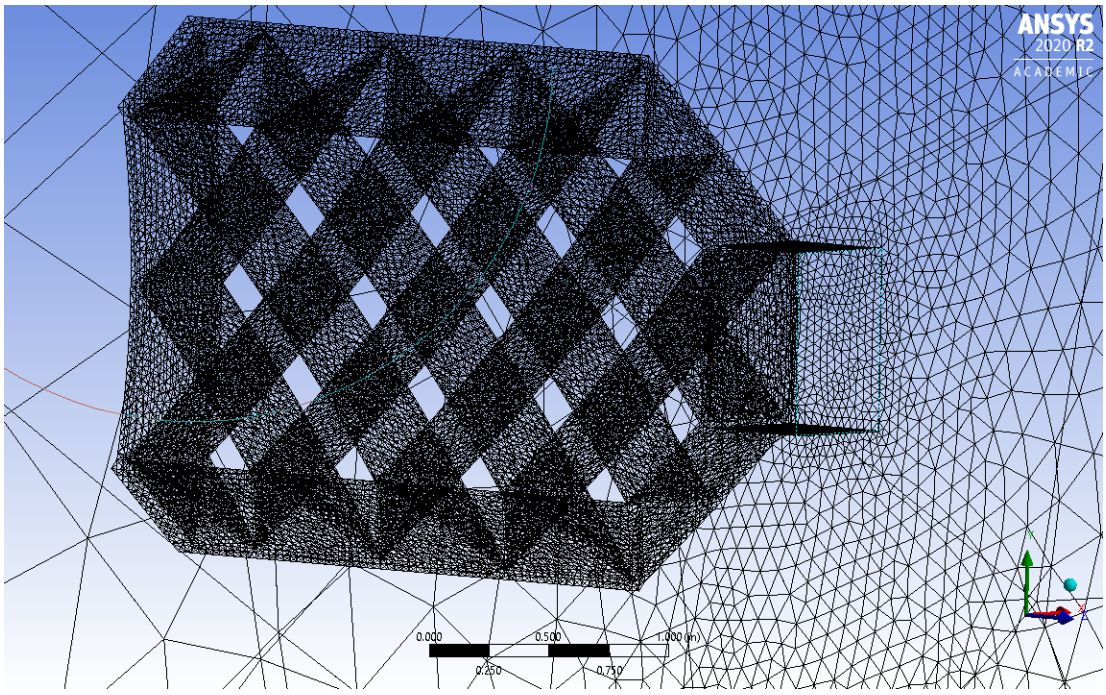


Figure 2-40 Grid Fin Domain Geometry Mesh, Detailed

With the mesh successfully sized, there are a total of approximately 1.3 million elements in the mesh. This is a very finely detailed mesh, which is necessary to model the flow over the thin webs of the grid fins as described by Manuwar [20]. The smallest element edge length is approximately 0.005”, which means that the mesh is sufficiently detailed for the 0.01” (0.254mm) thickness of the grid fin webs.

A set of boundary conditions must be defined to import this mesh properly into Fluent and initialize the simulation conditions. The faces that would act as the boundary conditions of the domain were named in Design Modeler and moved into Fluent. The boundary conditions are shown in Figure 2-41 and are described in

Table 2-3.

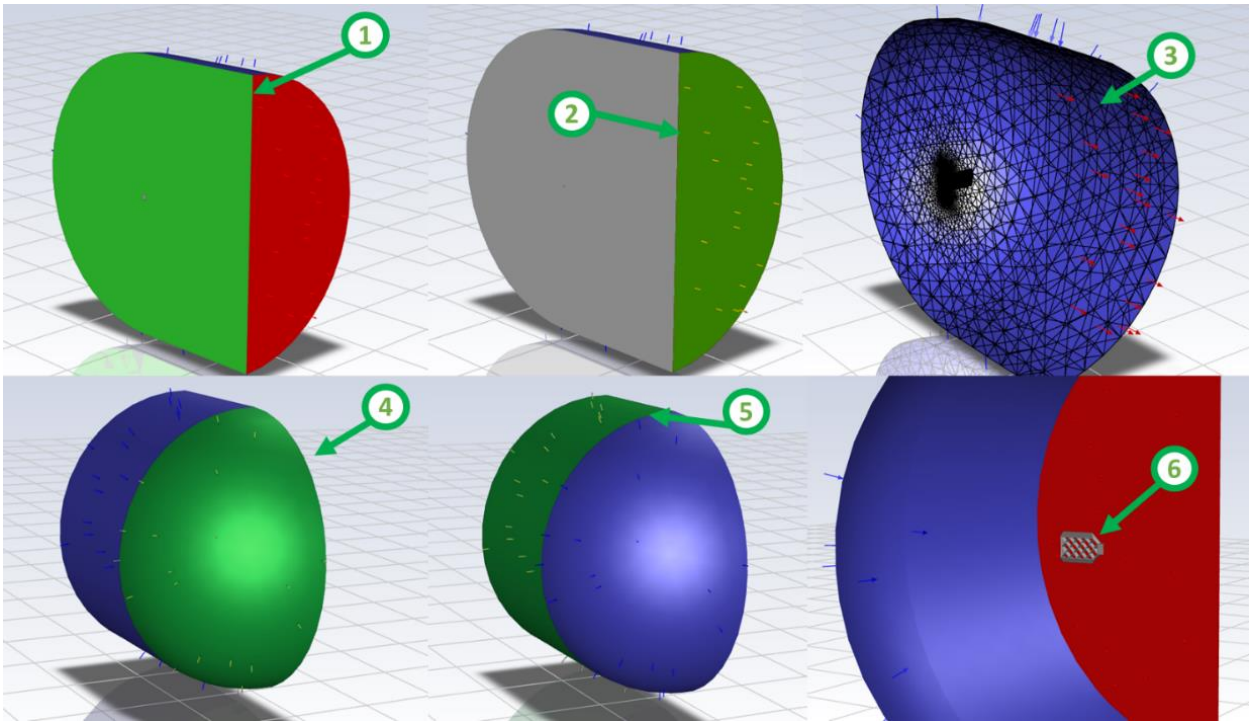


Figure 2-41 Grid Fin Boundary Conditions, Fluent

Table 2-3 Boundary Conditions, Grid Fin Fluent Simulation

Boundary Condition (Figure 2-41)	**Boundary Condition Type**	**Associated Values**
1	Wall	See Table 2-2
2	Pressure Outlet	See Table 2-2
3	Interior	Considered a zone, the fluid interior, rather than a boundary condition
4	Velocity Inlet	V = Terminal Velocity = 238.2185 ft/s (72.60899 m/s) Flow X Component = cos (α) Flow Y Component = sin (α) Flow Z Component = 0
5	Velocity Inlet	V = Terminal Velocity = 238.2185 ft/s (72.60899 m/s) Flow X Component = cos (α)

		Flow Y Component = sin (α) Flow Z Component = 0
6	Wall (Grid Fin)	See Table 2-2

The velocity inlet boundary condition was chosen for its simplicity in changing the flow direction, which is necessary for finding the drag and lift on the grid fin for various angles of attack (α). By specifying the flow component, Fluent takes care of assigning and initializing the correct fluid properties to the inlet boundaries based on the pressure outlet conditions and other assumptions made.

The velocity that is needed to test the effectiveness of the grid fins is the terminal velocity, since this is the freestream velocity that the grid fins will encounter during descent. In order to find the terminal velocity of the simplified descent body pictured in Figure 2-42, we assumed that terminal velocity from the grid fins at $\alpha = 0$ will be the freestream used for each simulation of the grid fins. Several Fluent simulations were run to find the $\alpha = 0$ drag of an individual grid fin, with an initial velocity of 200 ft/s (60.96 m/s) as a first guess. The dynamic pressure contours of the initial Fluent simulation run are shown in Figure 2-43 with pressure contours plotted on the grid fin surface and on the XY plane at the grid fin hinge.

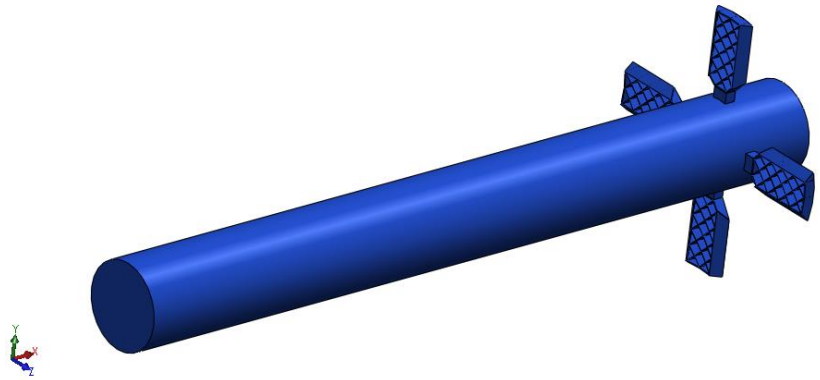


Figure 2-42 Simplified Descent Body Geometry

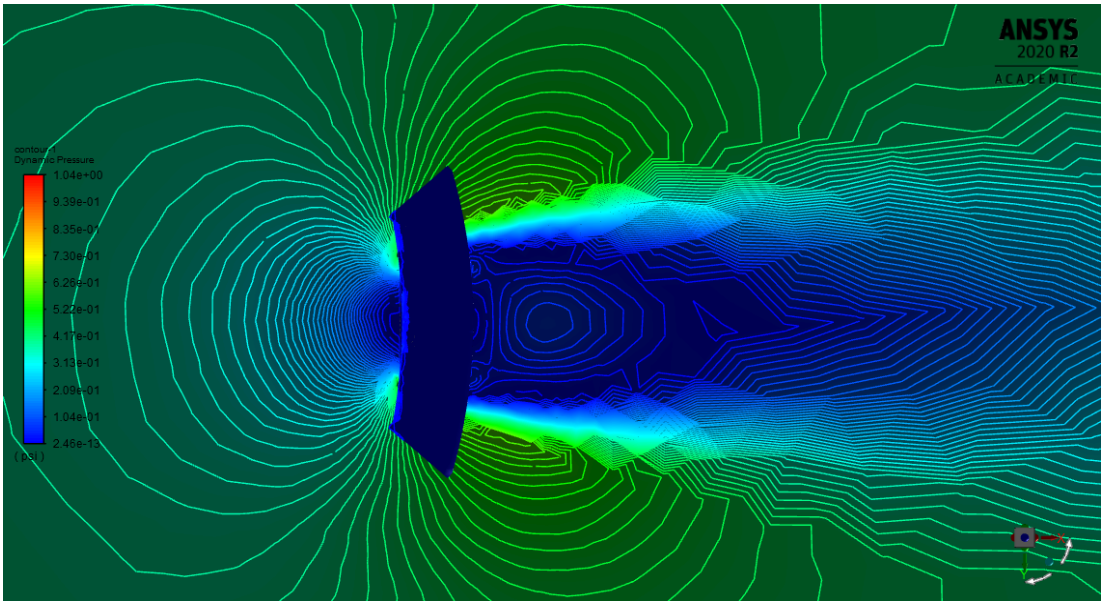


Figure 2-43 Grid Fin Dynamic Pressure Contours, Initial Fluent Run, α = 0

Further, ARS had to determine the drag on the body of the descent stage of the rocket to accurately calculate the terminal velocity. The descent stage was assumed to be a simple cylinder of length 36” in the X-axis, consistent with the OpenRocket model. We isolated the simple cylinder geometry from Figure 2-42 in Fluent and ran a simulation with a very similar set up as described above with the grid fin, at velocity of 200 ft/s (60.96 m/s) and angle α at 0 degrees along the X axis shown in Figure 2-42. Dynamic pressure contours for the simple descent body are plotted on the cylinder or airframe body in Figure 2-44 with wind along the X axis.

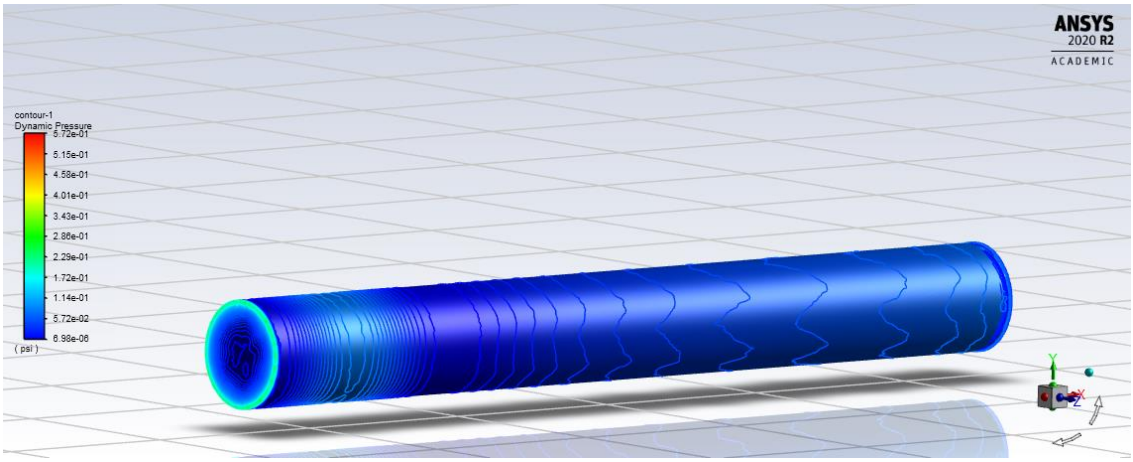


Figure 2-44 Descent Stage Simple Geometry Dynamic Pressure Contours

Fluent was primarily used to find force of drag in this case. The force of drag equation, Eq. 16, is used the drag coefficient of the descent body:

$$F_D = \frac{1}{2}\rho V^2 C_d A \quad (16)$$

We found the coefficient of drag of the descent body C_d to be 1.0664, with area A being the projected windward areas of the geometries found from SOLIDWORKS. At the time of the simulations, the bill of materials (BOM) showed that the estimated mass of the rocket was 5.59 kg. By setting the force of gravity equal to drag in Eq 16, the equation for terminal velocity can be found as Eq. 17.

$$V_{terminal} = \sqrt{\frac{2mg}{\rho A C_d}} \quad (17)$$

We found terminal velocity of the descent stage to be 238.2 ft/s (72.6 m/s). Using this value as the input for the inlet conditions of the simulation, 16 Fluent simulations were run for angles of attack ranging from 0 to 70 degrees. At each angle of attack, Fluent was used to output the center of pressure, drag force, lift force, and hinge moment. From these values, resultant forces and moments on the rocket were found using Eq. 14 and Eq. 15. The moments induced on the rocket were found by multiplying the resultant aerodynamic force by its respective moment arm. The equations formulated for finding the moments are expressed with Eq. 18 and Eq. 19.

$$M_X = F_{LR}(z_{cp} + R) \quad (18)$$
$$M_y = F_{DR}(y_{cp} + d_{cm}) \quad (19)$$

Here, M_x is rolling moment, M_y is pitching moment, z_{cp} is the horizontal distance from the center of pressure to the fin hinge, R is the outer radius of the rocket, y_{cp} is the vertical distance from the center of pressure to the grid fin hinge, and d_{cm} is the distance from the grid fin hinge to the rocket's center of mass found in SOLIDWORKS to be equal to approximately 0.41 meters or 1.35 feet. The values obtained for individual grid fins can be used to develop a control model for the descent of the rocket. It is important to note that pitching moment becomes coupled with yawing moment if the grid fin is not lined up with the Z-axis. Grid fins lined up along the Z-axis

induce a pitching moment, and grid fins lined up with the Y-axis induce a yawing moment. This means that the roll angle will affect the resultant moments and is accounted for in the descent control scheme. The results of these calculations and the simulations are considered the aerodynamic model for the grid fin and are plotted in Figure 2-45 through Figure 2-47.

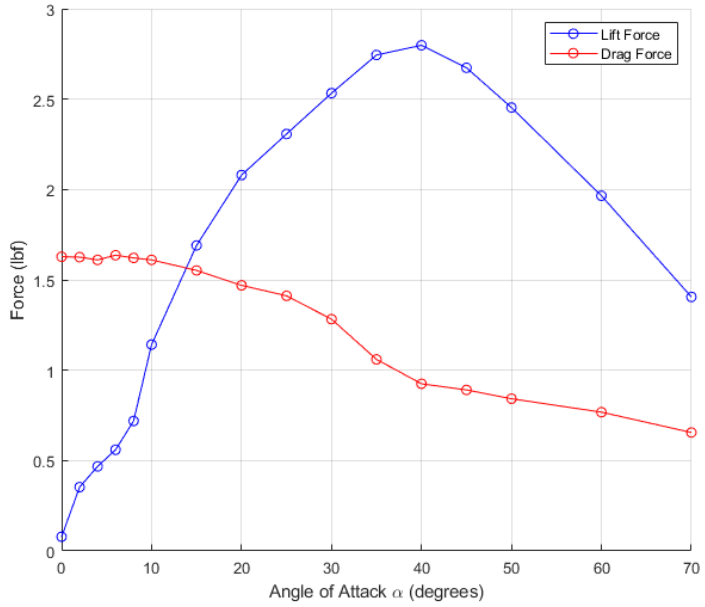


Figure 2-45 Lift and Drag Forces on Grid Fin, Fluent Results

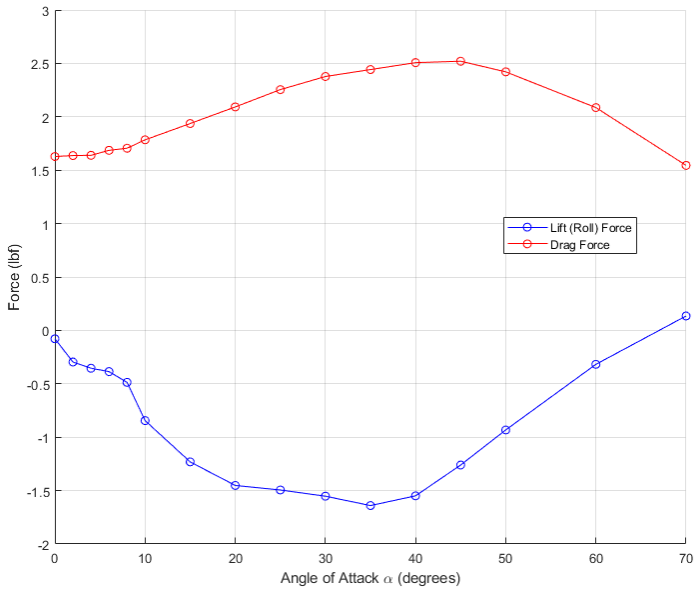


Figure 2-46 Resulting Forces on Rocket, Single Grid Fin

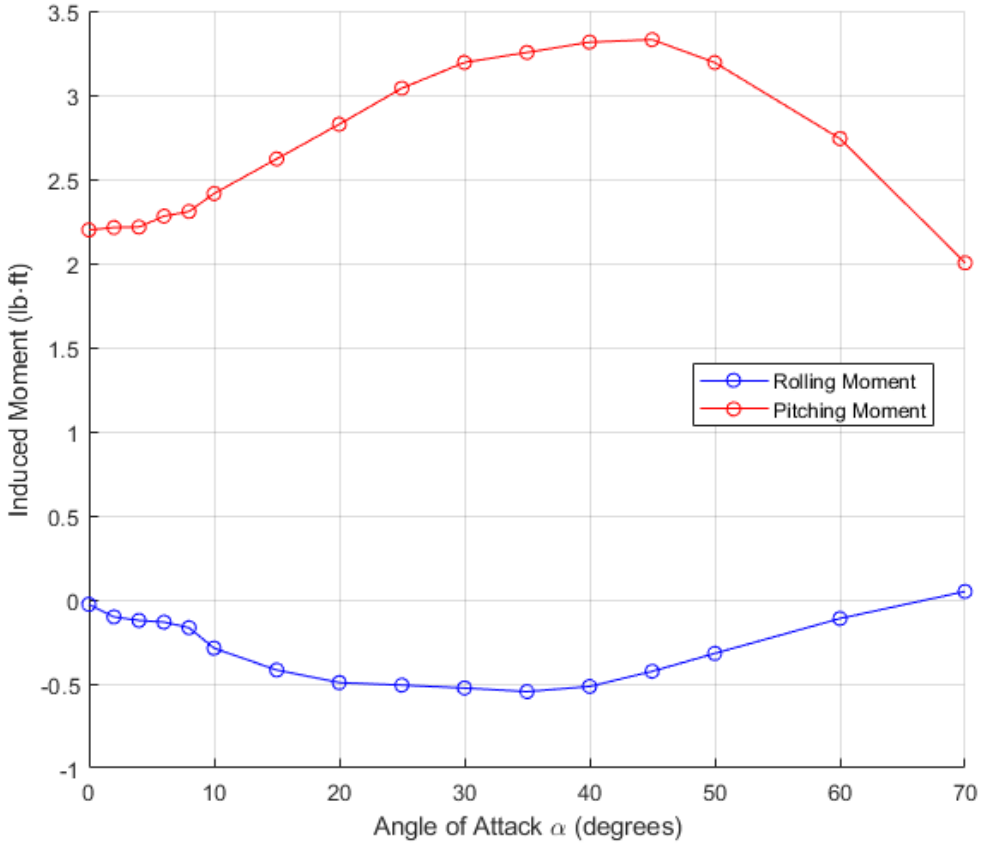


Figure 2-47 Resulting Moments on Rocket, Single Grid Fin

The results of the Fluent simulations show that the maximum drag on the descent stage occurs with all grid fins deflected to 45 degrees. Maximum roll authority occurs when the grid fins are deflected to 35 degrees. The aerodynamic model produced many pressure contours plots, with dynamic pressure being plotted on the grid fin surface along the X-Y plane at the hinge (Z = 0). Some of the pressure contour plots are omitted for brevity, and the rest are shown in Figure 2-48 through Figure 2-52.

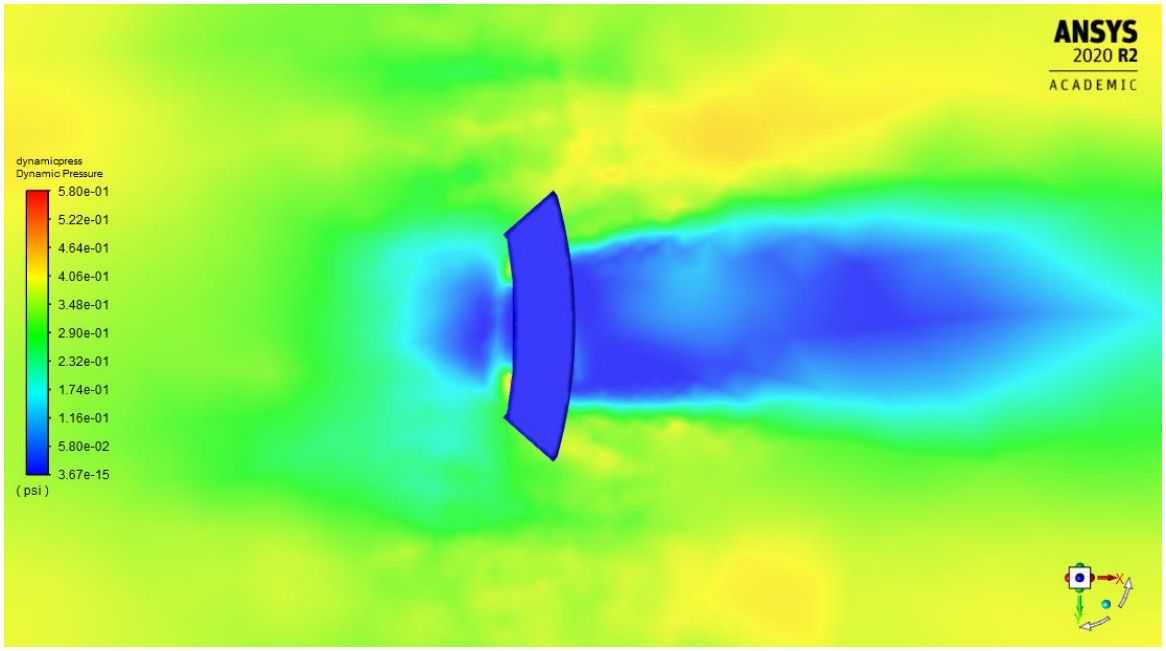


Figure 2-48 Dynamic Pressure Contour, Grid Fin α = 0 degrees at hinge (Z=0) plane

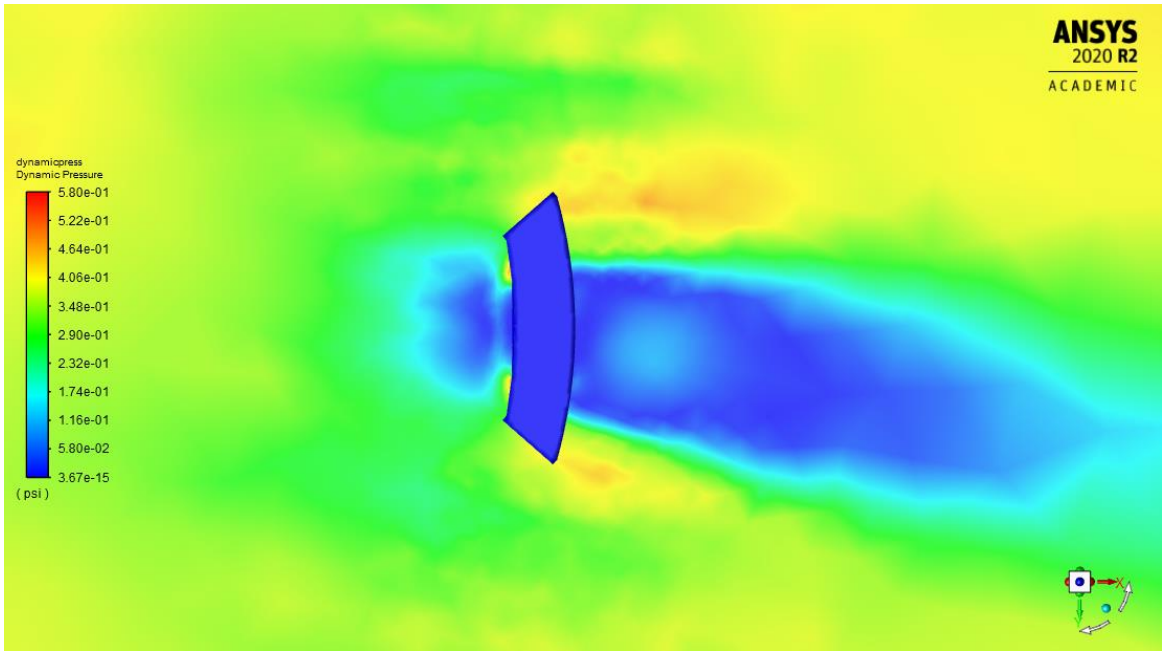


Figure 2-49 Dynamic Pressure Contour, Grid Fin α = 6 degrees at hinge (Z=0) plane

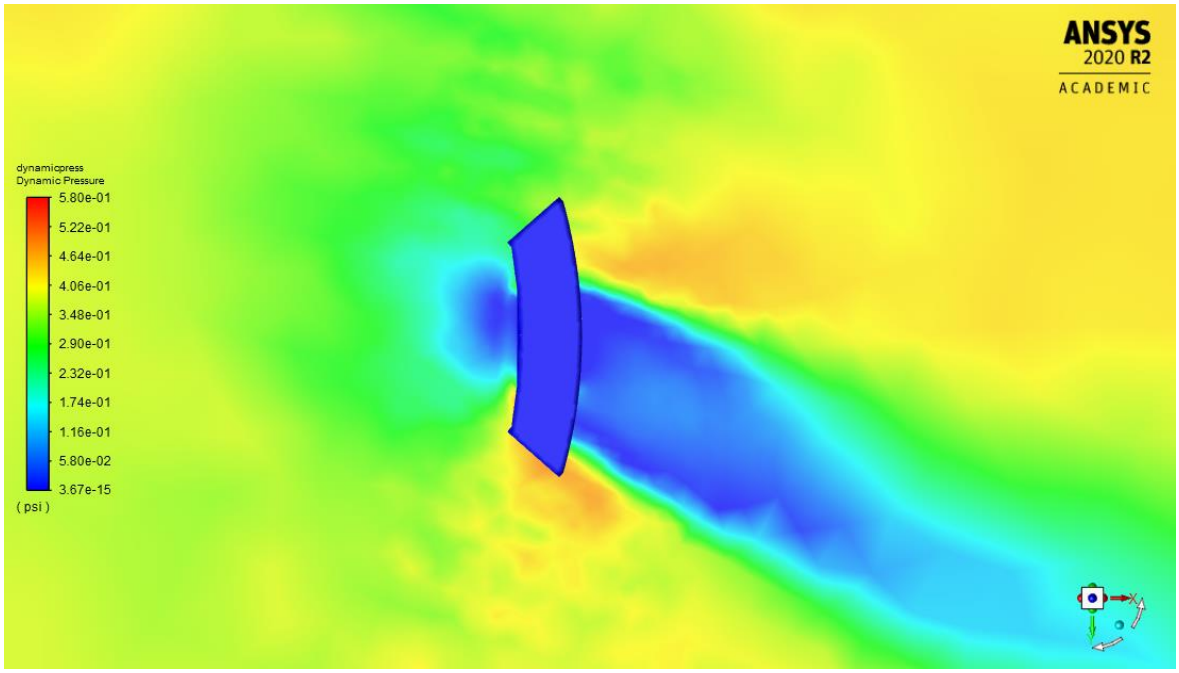


Figure 2-50 Dynamic Pressure Contour, Grid Fin α = 20 degrees at hinge (Z=0) plane

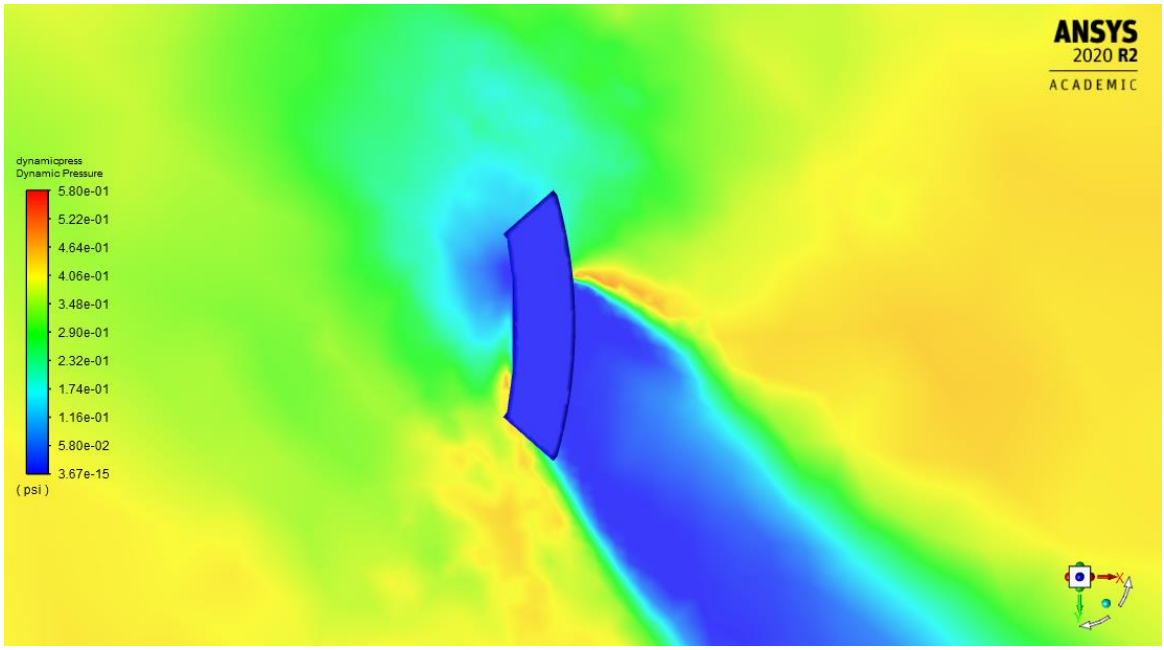


Figure 2-51 Dynamic Pressure Contour, Grid Fin α = 45 degrees at hinge (Z=0) plane

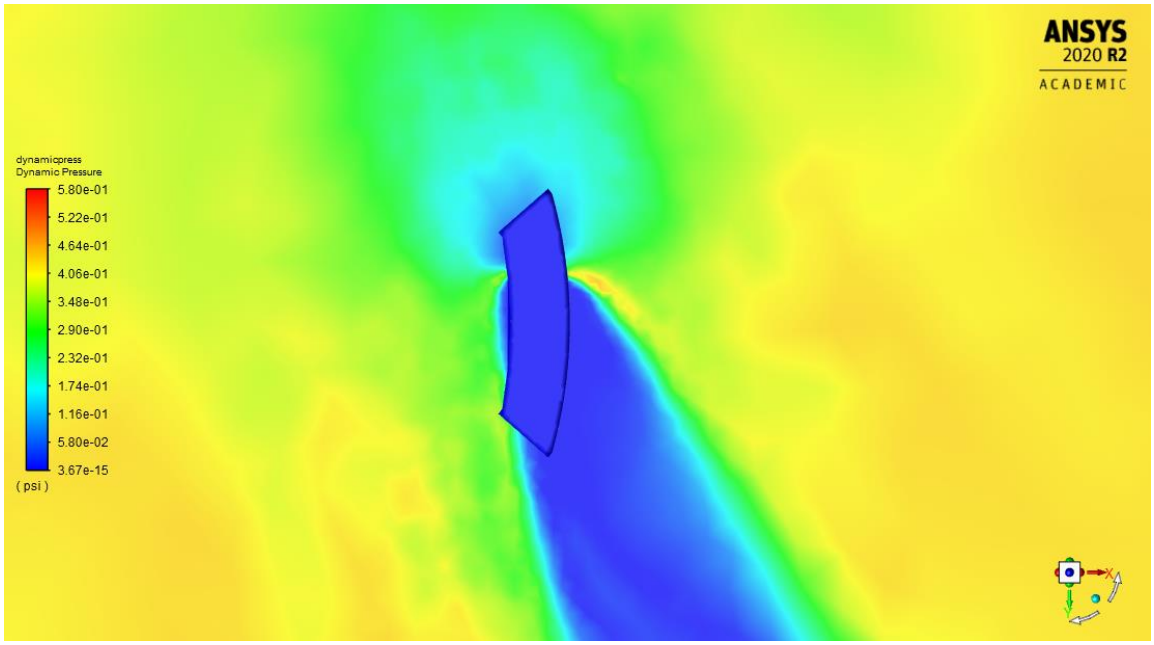


Figure 2-52 Dynamic Pressure Contour, Grid Fin α = 70 degrees at hinge (Z=0) plane

2.2.5 Grid Fin Stress Distribution (Analysis Task 3)

Ansys allows CFD solutions from Fluent to be imported directly into the Ansys Structural software for finding stresses and loads on the grid fins. The aerodynamic model created in Fluent in Section 2.2.4 was imported directly onto the grid fin geometry, which was meshed separately in another instance of an Ansys workflow, as shown in Figure 2-53.

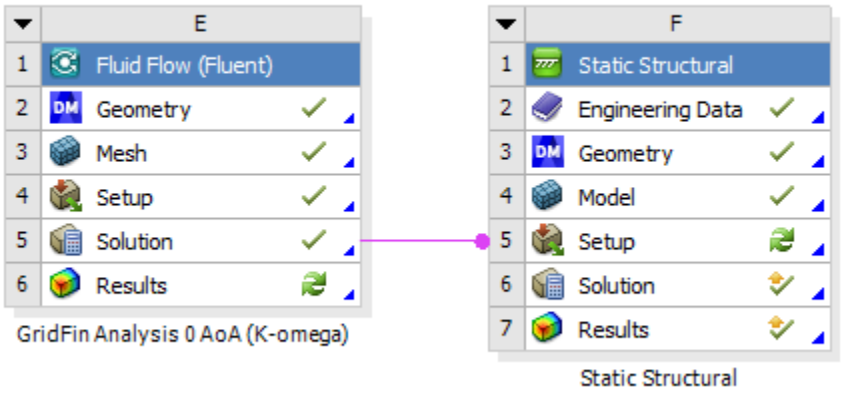


Figure 2-53 Fluent Aerodynamic Model Import to Ansys Structural

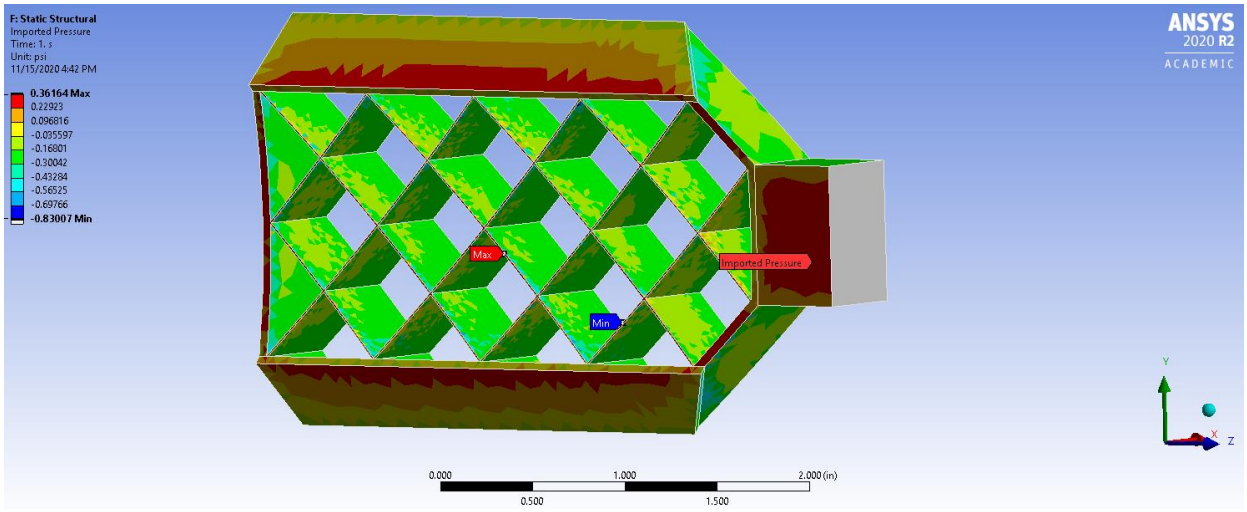


Figure 2-54 Fluent Grid Fin Imported Pressure Load

The pressure load was then accessible during the meshing of the grid fin in Ansys Structural, where the pressure load was designated by Fluent to be specifically on the CFD surface of choice, which for this analysis is the pressure load on the surface of the grid fin wall boundary discussed in Section 2.2.4. Once the pressure load was applied and the mesh was created in Workbench, solutions could be found. The first step in creating the mesh is to import the grid fin from SOLIDWORKS. Next a fine mesh had to be selected in order to best fit the size of the grid fin for maximum accuracy. Once the mesh was created, the hinge was fixed creating the only fixed point of the entire part. The first model, as shown in Figure 2-55, was a model of the deformation of the grid fin at terminal velocity.

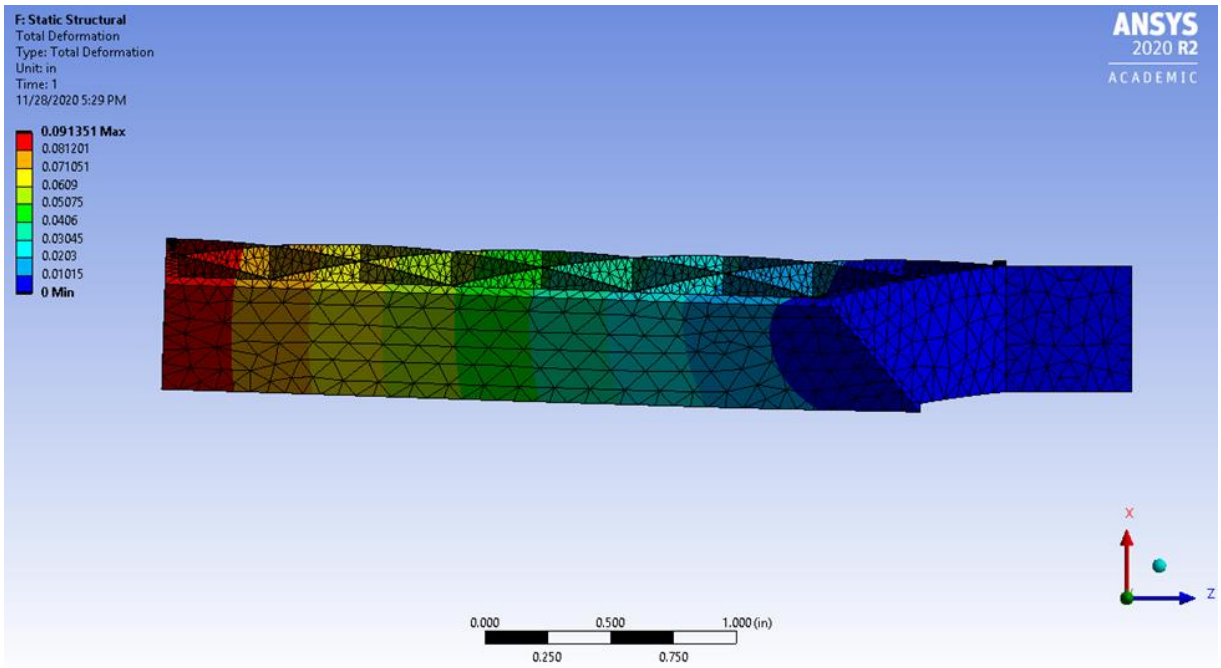


Figure 2-55 Deformation on Grid Fin (Side View)

The deformation that was calculated by Ansys was expected as the grid fin has one support and is therefore similar to a cantilever beam. This translates to the deformation increasing along the z-axis. In the case of the grid fin, it is fixed to the rocket so the part of the grid fin closest to the rocket would experience the least deformation. Table 2-4 summarizes the deformation throughout the grid fin.

Table 2-4 Results of Deformation Model

Minimum Deformation	0 in. (0 m)
Maximum Deformation	9.1351 E-2 in. (.00232 m)
Average Deformation	4.0116 E-002 in (.0010 m)

As can be seen in Table 2-4, the deflection at the hinge (minimum deformation labeled in table) was extremely low, while it continued to increase along the z-ais. This deformation is far within the maximum yield for the material, which means the grid fin could withstand this deformation during descent.

Figure 2-56 shows the equivalent stress throughout the grid fin at terminal velocity. The areas closest to the rocket in the grid fin experienced the highest amounts of stress. The stress model was important as it allowed our group to see whether or not the grid fins would fail.

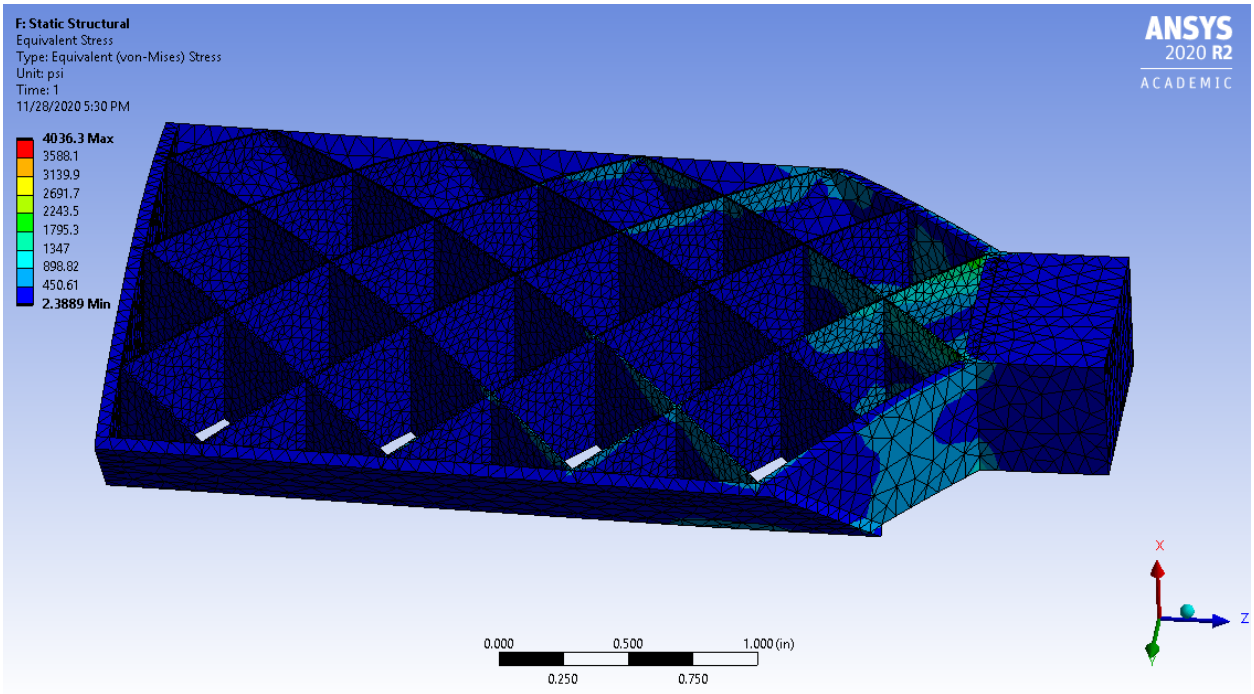


Figure 2-56 von-Mises Equivalent Stress on Grid Fin

Table 2-5 Equivalent Stress Results

Minimum Stress	2.3889 psi (.01647 MPa)
Maximum Stress	4036.3 psi (27.83 MPa)

Average Stress	218.5 psi (1.506 MPa)

	METRIC[1]		IMPERIAL[1]		METHOD
	Green[2]	Post-Cured[3]	Green[2]	Post-Cured[3]	
Tensile Properties					
Ultimate Tensile Strength	35 MPa	61 MPa	5076 psi	8876 psi	ASTM D 638-14
Tensile Modulus	1.4 GPa	2.6 GPa	203 ksi	377 ksi	ASTM D 638-14
Elongation	32.5 %	13 %	32.5 %	13 %	ASTM D 638-14
Flexural Properties					
Flexural Stress at 5% Strain	39 MPa	86 MPa	5598 psi	12400 psi	ASTM D 790-15
Flexural Modulus	0.94 GPa	2.2 GPa	136 ksi	319 ksi	ASTM D 790-15
Impact Properties					
Notched IZOD	not tested	18.7 J/m	not tested	0.351 ft-lbf/in	ASTM D256-10
Temperature Properties					
Head Deflection Temp. @ 1.8 MPa	not tested	62.4 C	not tested	144.3 °F	ASTM D 648-16
Heat Deflection Temp. @ 0.45 MPa	not tested	77.5 C	not tested	171.5 °F	ASTM D 648-16
Thermal Expansion (-30 to 30° C)	not tested	78.5 um/m/C	not tested	43.4 µin/in/°F	ASTM E 831-13

[1] Material properties can vary with part geometry, print orientation, print settings, and temperature.

[2] Data was obtained from green parts, printed using Form 2, 100 µm, Grey Pro settings, without additional treatments.

[2] Data was obtained from parts printed using Form 2, 100 µm, Grey Pro settings and post-cured with a Formcure for 120 minutes at 80 C.

Figure 2-57: Grey Pro Data Sheet [81] © FormLabs 2020

The results in Table 2-5 and Figure 2-57 show that the maximum stress on the grid fin is 4036 psi (27.83 MPa), while the ultimate tensile strength of the material being used—Post-Cured resin—is 8876 psi (61.0 MPa). This means that the grid fins will be able to handle stress at terminal velocity.

By using Fluent and Workbench, our group was also able to retrieve the hinge moment at different angles of attack which can be seen in Figure 2-58. The figure illustrates a parabola-type graph with a peak hinge moment at an AOA of about 40 degrees.

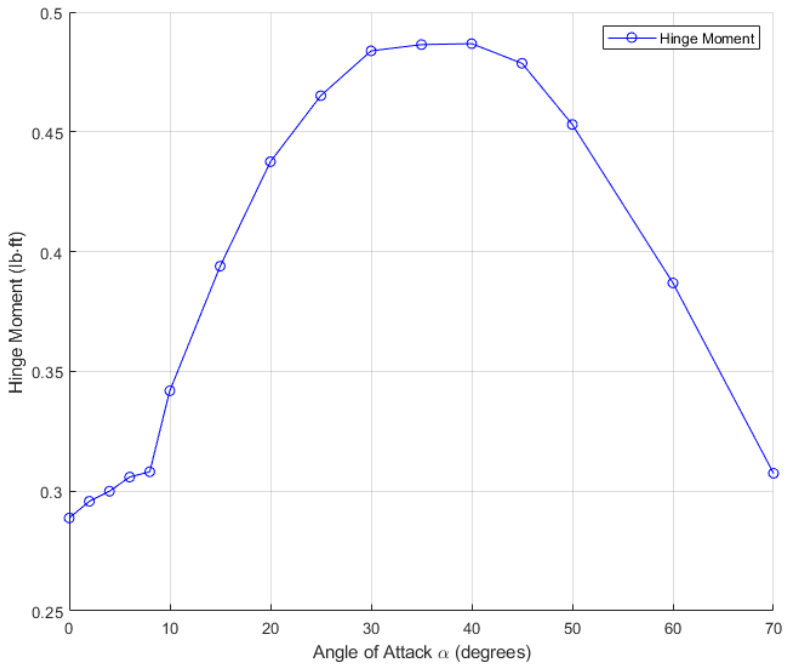


Figure 2-58 AOA v. Hinge Moment

3 Propulsion, Thermal, and Separation Systems – Design and Analysis

The Propulsion, Thermal, and Separation Systems subteams goal was to model the combustion physics and performance of a selected solid rocket motor (SRM), simulate the temperature distribution and fluid flow distribution within the SRM, design a mechanical stage separation, and develop a novel compressed gas propulsive landing system. Cantera was used to model the combustion reaction of the SRM and provided material inputs to simulate the temperature distribution and fluid flow inside of the motor. SOLIDWORKS was used to design clasps for the mechanical stage separation system which were then printed in the Higgins Laboratories. MATLAB was used to find the total impulse needed to land the rocket and required tank size based on a chosen nozzle throat diameter of the cold gas thruster.

3.1 Methodology

In this section, the methodologies of the PTSS subteam's analysis tasks are presented.

3.1.1 Engine Systems

The PTSS subteam decided to use a single motor in the launch system for our rocket. While we did not intend to attempt to launch this rocket, we preceded with our motor selection under the assumption that we would be launching. We did however include two stages in the rocket design so that the motor used for launch and stability fins could be released at apogee and recovered by a parachute. This would allow for a significant reduction in the dry mass of the rocket when attempting to land using the cold gas thruster. Based on advice from other members of the MQP that had previous experience in model rocketry we decided to explore motors from Cesaroni Technologies Inc. (CTI) due to their reliability and ease of use compared to Aerotech. We originally estimated the mass of our rocket to be 16 lbm (7.26 kg) and wanted to have a minimum thrust to weight ratio of 5. We wanted to keep the first stage of our rocket as short as possible so we began looking first at 3-grain motors from CTI. Eventually, we selected the 2.126 in (54 mm), 3-grain, K360 motor from CTI. This provided us with an average thrust of 80.331 lbf (357.333 N) resulting in a thrust of weight ratio of around 4.9 which is sufficient for our application. A summary of the rocket motor characteristics is provided in Table 3-1 [82].

Table 3-1 Properties of K360 CTI Rocket Motor [79]

Diameter	2.126 in (54 mm)
Length	9.291 in (236 mm)
Peak Thrust	91.081 lbf (405.151 N)
Average Thrust	80.331 lbf (357.333 N)
Thrust Duration	3.5 s
Total Impulse	281.16 lbf-s (1250.6686 N-s)
Specific Impulse	184.4 s
Total Weight	2.716 lbm (1232 g)

3.1.2 Ignition System

The commercial motor kit for the K360 motor used in our rocket design includes an e-match igniter. We plan to use this e-match igniter for ignition of our first stage rocket motor. The provided e-match igniter is a coiled wire dipped in a pyrogenic substance with a composition of 31-32% of barium chromate, 42-43% magnesium powder, and 26-27% viton

fluoroeslastomer [83]. The igniter used to launch the rocket will be remotely powered and controlled by the controller provided at the selected launch site.

3.1.3 Motor Performance Analysis (Analysis Task 1)

Model rocketry websites provide access to databases of motor performance test data. This provides thrust curve for the motor and other parameters such as total impulse, specific impulse, average thrust, maximum thrust, and burn time [84]. The PTSS subteam developed a motor performance model using Cantera to compare calculated motor thrust and specific impulse to the test data. Cantera is an open-source software that allows users to perform fundamental thermodynamic, chemical kinetic, and species transport calculation [85]. The purpose of this model was to estimate the motor performance and estimate the equilibrium conditions inside the motor. The equilibrium temperature and pressure, and fluid properties, such as density, viscosity, specific heat at constant pressure, and thermal conductivity, will be used in as inputs to the thermal model in Section 3.1.4.

The PTSS subteam developed an equilibrium model in Cantera for the combustion of the motor propellant. Due to the complexity of the combustion of a solid rocket motor several assumptions were made in this model starting with the propellant formulation. CTI publishes vague information on the exact ingredients and composition used in their model rocket motor propellant grains as described in Table 3-2 [86]. For the purpose of this model it was assumed that the propellant composition was 90% ammonium perchlorate (AP) and 10% aluminum on a molar basis. The combustion of AP and aluminum produces a few main product species and many traces species. To simplify the analysis we decided to use only the main product species in the Cantera model. The dominant product species were determined to be oxygen, water, hydroxyl, nitrogen, hydrogen chloride, chlorine, and alumina using NASA's online Chemical Equilibrium with Applications software [87]. This results in the chemical reaction being described as shown in Eq. 20 where the variables, a – g, are the product species mole fractions to be determined from the Cantera equilibrium model.

$$0.9\,NH_4ClO_4 + 0.1\,Al \rightarrow aO_2 + bH_2O + cOH + dN_2 + eHCl + fCl + gAl_2O_3 \qquad 20$$

Other major assumptions included: chemical equilibrium in the combustion chamber, steady-state and frozen flow, isentropic flow in the nozzle, no chemical reactions

amongst product species, no nozzle throat erosion, single-phase flow, no frictional or boundary layer losses, no heat losses, and constant volume. To run an equilibrium condition in Cantera, two of the following properties must be held constant: temperature, pressure, internal energy, enthalpy, entropy, and volume. For our model we chose to keep specific internal energy and specific volume constant. However, it is necessary to note that the internal energy would not be constant as heat would be lost to the surroundings.

Table 3-2 Published Rocket Motor Ingredients from CTI [83]

Ingredient	Percentage
Ammonium Perchlorate	40-85%
Metal Powders	1-40%
Synthetic Rubber	10-30%

With all necessary assumptions identified, the PTSS subteam developed an input file for Cantera where an ideal gas mixture was defined. Then, a species was defined for each of the reactants and products in the defined chemical reaction using thermodynamic data from NASA's ThermoBuild database [88]. Transport data was also included for species, collected from several databases and reports for all species except alumina as data could not be found for alumina [89] [90] [91]. We installed Cantera and used the MATLAB interface to write a script which would read our input file. Initial conditions were provided and then we ran an equilibrium condition calculation using Cantera MATLAB functions while holding internal energy and specific volume constant. We decided to use an initial pressure of 14.7 psia (101,325 Pa) and an initial temperature of 1,004.67 R (558.15 K) since this is the minimum temperature required to initiate combustion of the igniter provided by CTI for the rocket motor [83].

After running the equilibrium calculation in Cantera, values for the temperature, pressure, density, specific heat at constant pressure and volume, and mean molecular weight of the combustion gas are retrieved using Cantera's built-in functions. Other necessary inputs are the nozzle throat area and expansion ratio which were determined to be 0.454 in. (11.55 mm) and 7.54 respectively, from commercially available, 2.126 in. (54 mm) rocket motor nozzles [92]. These values were then used to calculate thrust and specific impulse using supersonic isentropic flow equations 3 - 9, Eq. 21 and Eq. 22.

$$\dot{m} = \rho_e u_e A_e \qquad 21$$

$$I_{sp} = \frac{T}{\dot{m} g} \quad (22)$$

The MATLAB script used to evaluate the Cantera model and evaluate the motor performance using isentropic flow equations for this analysis can be found in Appendix D: Cantera Model MATLAB Code

3.1.4 Thermal Distribution Model (Analysis Task 2)

The PTSS subteam developed a thermal and fluid flow model for the K360 rocket motor to estimate the external temperature of the motor case and determine if there are any points of risk for structural degradation due to overheating. A multiphysics simulation software, COMSOL, was used to create 2D axisymmetric model using the heat transfer in solids and fluids, and the laminar flow physics interfaces. This model was implemented with a time-dependent study to provide an accurate representation of the thermal distribution in the motor.

Three domains of interest were selected to simplify this analysis, they are the combustion gas, propellant grain, and aluminum motor case. The geometry of the three domains used in this analysis are shown in Figure 3-1. All domains were 9.291 in (236 mm) long based on dimensions available for the motor grain length [82]. Data was not readily available for the outer diameter of the K360 propellant grain so this was estimated from another 2.126 in (54 mm) rocket motor at 1.875 in (47.625 mm) [93]. This information was then used to estimate the width of the three domains. The first domain is the fluid domain which was estimated to have a width of 0.859 in (21.8125 mm). The second domain is a volumetric heat source which represents the combustion flame zone and was estimated to have a width of 0.393 in (1 mm). The third domain is the propellant grain which was estimated to have a width of 0.0393 in (1 mm). The fourth domain is the motor case which was estimated to have a width of 0.1255 in (3.1875 mm). These dimensions were selected to reflect an estimated condition of the motor towards the end of its 3.5 second burn time.

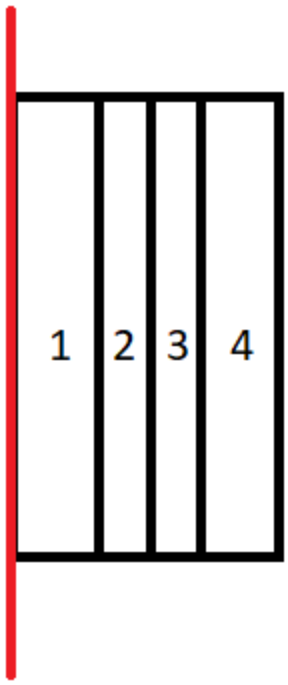


Figure 3-1 Geometry of the Domains in the COMSOL Model. Domains are axisymmetric with red line representing centerline.

The heat source domain was defined as a general heat source which required an input for the heat generated within the flame combustion volume. Using the defined dimensions for domain 2 it was possible to calculate the volume. Eq. 23 and Eq. 24 were used to determine the heat generated per unit volume based on the results from the Cantera model.

$$Q_0 = \frac{\dot{m}_{fuel} h_{RP}}{V} \tag{23}$$

The mass flow rate of the fuel was found using the molar mass of the combustion gas, M_m, the mole fraction of the fuel (aluminum), v_f, and the molar mass of the fuel, M_{mf} as described in 24.

$$\frac{\dot{m}}{M_m} = \frac{\dot{m}_{fuel}}{v_f M_{mf}} \tag{24}$$

The heat of reaction, h_{RP}, was found from the Cantera model by finding the difference in the enthalpy before and after the equilibrium reaction was performed. A summary of these values is provided in Table 3-3.

Table 3-3 Heat source and mass flow rate calculations

$\dot{m}_{fuel}$	0.004715	kg/s
h_{RP}	7034690	J/kg
V	3.308E-5	m^3
Q_0	4.108E8	W/m^3
$\dot{m}$	0.1895	kg/s
M_m	28.2591	kg/kmol
M_{mf}	26.9815	kg/kmol
v_f	0.1	-

After calculating all of these values, each of these domains were assigned a material and initial conditions as described in Table 3-4.

Table 3-4 Description and Initial Conditions of Domains

Domain	Description	Material	Width	Height	Initial Conditions
1	Combustion Gas Fluid	Fluid	0.859 in (21.8125 mm)	9.291 in (236 mm)	P = 14.7 psia (0.101 MPa) **u** = 0 ft/s (0 m/s) T = 527.67 R (293.15 K)
2	Heat Source	Fluid	0.393 in (1 mm)	9.291 in (236 mm)	T = 527.67 R (293.15 K) Q_0 = 3.969E7 Btu/h-ft^3 (Q_0 = 4.108E8 W/m^3)
3	Solid Propellant Grain	Propellant	0.0787 in (1 mm)	9.291 in (236 mm)	T = 527.67 R (293.15 K)
4	Motor Case	Aluminum	0.1255 in (3.1875 mm)	9.291 in (236 mm)	T = 527.67 R (293.15 K)

For the thermal model, we included ambient conditions which were assumed to be a temperature of 527.67 R (293.15 K) and pressure of 14.7 psia (1 atm). Ambient conditions were used for the environment and were kept at their respective default values. The aluminum material properties also utilized data provided in the COMSOL material database. The combustion fluid properties for the combustion products were determined from our Cantera model and are summarized in Table 3-5. We determined the density, heat capacity at constant pressure, and specific heat ratio from the motor performance model results in Section 3.1.3. We estimated the thermal conductivity and dynamic viscosity for the fluid by running a second equilibrium case in Cantera for only the combustion of ammonium perchlorate as described by Eq. 25.

$$NH_4ClO_4 \rightarrow aO_2 + bH_2O + cOH + dN_2 + eHCl + fCl \qquad 25$$

We were unable to identify the necessary transport properties for alumina. Since the exact composition of the solid propellant grain and material properties were unknown, we estimated the thermal conductivity and heat capacity at constant pressure using experimental data collected from a sample of NASA's Space Shuttle solid rocket booster. We used a curve fit from previous research to estimate, thermal conductivity and heat capacity at constant pressure at a temperature of 671.67 R (373.15 K). We estimated the density of the material based on the experimental composition of 70% ammonium perchlorate, 16% aluminum, and 14% polybutadiene acrylonitrile binder [94].

Table 3-5 Material Properties for COMSOL Model Domains

Material	Density	Heat Capacity, C_p	Thermal Conductivity	Dynamic Viscosity	Specific Heat Ratio
Aluminum	168.55 lbm/ft^3 (2700 kg/m^3)	0.215 Btu/lbm-R (900 J/kg-K)	137.6 Btu/hr-ft-R (238 W/m-K)	-	-
Propellant	120.92 lbm/ft^3 (1937 kg/m^3)	0.353 Btu/lbm-R (1476.26 J/kg-K)	0.352 Btu/hr-ft-R (0.6096 W/m-K)	-	-
Combustion Products	0.148 lbm/ft^3 (2.3676 kg/m^3)	0.379 Btu/lbm-R (1587.7 J/kg-K)	0.095 Btu/hr-ft-R (0.1653 W/m-K)	4.882E-5 lbm/ft-s (7.266E-5 Pa-s)	1.2275

Several different boundary conditions were used in the COMSOL models for the heat transfer in solids and fluids physics and in the laminar flow physics. In the heat transfer portion of

the model there were five boundary conditions defined as denoted in Figure 3-2 and Table 3-5. Boundary 1 is defined as a symmetry boundary since this is the axis which the 2D geometry is axisymmetric about. This boundary does not permit any heat flux across the boundary. Boundaries 2 and 4 are defined as thermal insulation boundaries. This is the default boundary condition COMSOL uses and was kept as the exact geometry and conditions at the forward and aft end of the propellant grain were not explored to simplify this model. The thermal insulation boundary does not permit any heat flux across the boundary. Boundary 3 was defined as a convective heat flux boundary. The properties for the convective heat transfer coefficient were selected using the default values for the aluminum material in COMSOL [95]. The external temperature around the motor case was assumed to be the same as the ambient temperature. If the heat flux across this boundary is positive then heat is added to the system otherwise, if it is negative, heat is loss to the surroundings [96]. A summary of the definitions and inputs for the boundary conditions in the heat transfer is provided in Table 3-6.

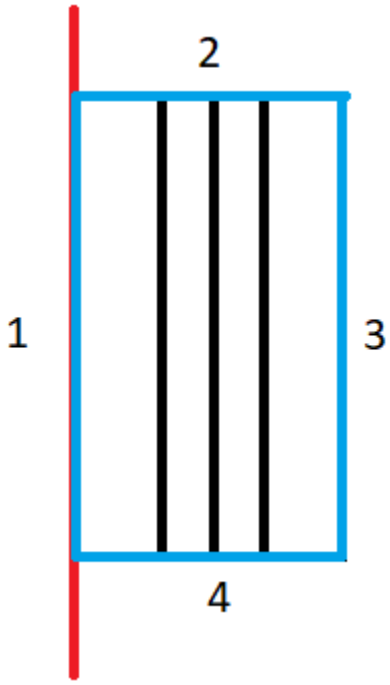


Figure 3-2 Boundary Conditions Defined in Heat Transfer Model. Domains are axisymmetric about the red centerline

Table 3-6 Summary of Heat Transfer Boundary Conditions and Inputs

Boundary	Condition	Governing Equation	Input
1	Symmetry	$-n \cdot q = 0$	-
2	Thermal Insulation	$-n \cdot q = 0$	-
3	Convective Heat Flux	$-n \cdot q = q_0$ $q_0 = h(T_{ext} - T)$	h = 4.41 Btu/hr-ft^2-R (25 W/m^2-K), air T_{ext} = 527.67 R (293.15 K)
4	Thermal Insulation	$-n \cdot q = 0$	-

The laminar flow interface was included in this analysis to simulate the flow of the combustion gas from the burning surface area to the inlet into the rocket nozzle. We assumed the pressure inside the combustion chamber to be constant at the combustion pressure estimated from the Cantera model. Thus, the flow was modeled with a weakly compressible flow which uses the compressible form of the Navier-Stokes and continuity equations but neglects the pressure dependence of density so the density is evaluated at the reference pressure [95]. A complete summary of the governing equations for the COMSOL models are provided in Appendix E: Equations in COMSOL Model. The reference temperature of the laminar flow interface was left at the default 527.67 R (293.15 K) while the reference pressure was changed to 0 psia (0 atm) so the absolute pressure of the combustion chamber could be used. The initial fluid properties are the same as defined in Table 3-4. In the laminar flow model, four boundary conditions were defined as denoted in Figure 3-3.

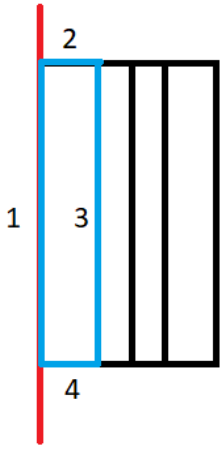


Figure 3-3 Boundary Conditions Defined in Fluid Flow Model

Boundary 1 is defined as an axial symmetry node which has a no-flux condition across the boundary. Boundary 2 is defined as a wall node which is generally used to describe fluid flow conditions at stationary, moving, and leaking (i.e. porous) walls. A no slip condition with zero translational velocity was used on this boundary to model a solid wall where the fluid velocity relative to the wall velocity is zero. Boundary 3 was defined as an inlet with an initial mass flow rate over the surface area. The mass flow rate was retrieved from the Cantera model results. Boundary 4 was defined as an outlet to represent the condition of the fluid flow into the inlet (i.e. into the converging section) of the rocket motor nozzle. The relative pressure at the boundary was calculated by subtracting the reference pressure of 1 atm from the estimated chamber pressure of the rocket motor. The suppress backflow check box was selected which adjusts the outlet pressure to prevent any fluid from entering the domain through the outlet boundary [95]. A complete summary of the boundary conditions and inputs used in the fluid flow model is provided in Table 3-7.

Table 3-7 Summary of Fluid Flow Boundary Conditions and Inputs

Boundary	Condition	Governing Equation	Input
1	Axial Symmetry	-	-
2	Wall	$\boldsymbol{u} = 0$	-
3	Inlet	$-\int_{\partial\Omega} \rho(\boldsymbol{u} \cdot \boldsymbol{n}) d_{bc} dS = \dot{m}$	$\dot{m} = 0.4177$ lb/s $(0.1895\ kg/s)$
4	Outlet	$[-p\boldsymbol{I} + \boldsymbol{K}]\boldsymbol{n} = -\hat{p}_0 \boldsymbol{n}$ $\boldsymbol{K} = \mu(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T)\boldsymbol{n}$ $\hat{p}_0 \leq p_0$	$\hat{p}_0 = 248.58\ psia$ $(1.7139\ MPa)$

The vector $\boldsymbol{K}$ is the viscous stress tensor and d_{bc} is the channel thickness defined as the area across which the mass flow occurs [95]. A complete list for all governing equations that are used to define the boundary conditions and solve for the fluid flow and temperature distribution in the rocket motor is provided in Appendix E: Equations in COMSOL Model.

3.1.5 Separation

The research team pursued two separation methods to successfully complete stage separation of the upper and lower stages. For the rocket, a mechanical stage separation consisting

of two sets of curved clasps were designed. The rocket nosecone separation also utilizes a black powder charge due to the complexity of alternative stage separation methods for such a small component of the model rocket. A mechanical stage separation was chosen due to the originality and innovative aspect of the separation method.

3.1.6 Black Powder Charges

Black powder charges are the most used method for stage separation in model rocketry. Because of their simplicity, black powder is a popular stage separation method, and separation occurs at apogee for separation component deployment at high speeds. This significantly increases parachute shock as well as reduces drag forces on the nosecone. Given an inner diameter of 3.9 in, the volume can be calculated using a length approximation of 4 in. In meters, the inner diameter of the Blue Tube is 0.099 m, and the length is 0.1016 m. These numbers can be used in Eq. 2**Error! Reference source not found.** to yield a value for mass of black powder corresponding to the volume of the section that is pressurized when the powder is ignited. The temperature for black powder combustion is 2670 deg Fahrenheit (1738.7 K) and R, the universal gas constant, is 8.314 J/mol-K (10.732 Psi-ft^3-lb/mol-°R) [97]. The molar mass of black powder is also necessary to calculate the moles of black powder needed, and this value is 162.1406 g/mol (5.719 oz/mol) [98]. Using this data, R_s can be calculated as 0.05128 J/g-K and n to be 1.766g (0.035274 oz). Based on hobbyist rocketry sources [99], the pressure to eject a nose cone is approximately 15 psig (103421 Pa) [99]. This number can vary based on how tight of a frictional fit the nosecone attaches to the upper stage. However, 15 psig was used to calculate the force needed to eject the nosecone since this is the largest value and we wanted to avoid an unsuccessful separation. Absolute pressure was also taken into account for calculations, using an atmospheric pressure of 14.18 psia (102042.4 Pascal) at an altitude of 1200 ft (365.76 m), the absolute pressure is 29.18 psia, or 201189.02 Pa [98]. All of these values and their corresponding units are shown in Equations 26-28.

$$V = \frac{\pi D^2 L}{4} \tag{26}$$

$$n = \frac{pV}{R_s T} \tag{27}$$

$$R_s = R n_{BP} \tag{28}$$

Table 3-8 Summary of Input Data Necessary to Calculate Black Powder Ejection of the Nosecone

Property	Value (Metric)	Unit (Metric)	Value (Imperial)	Unit (Imperial)
Volume, $\boldsymbol{V}$	7.83 X 10^{-4}	m^3	477815.9	in^3
Diameter, $\boldsymbol{D}$	0.09906	m	3.9	in
Length, $\boldsymbol{L}$	0.1016	m	4	in
Grams of Black Powder, $\boldsymbol{n}$	1.766	g	0.062293817	oz
Internal Pressure, $\boldsymbol{p}$	201189.02	N/m^2	201189.02	Pa
Universal Gas Constant, $\boldsymbol{R}$	8.314	J/mol-K	10.732	Psi-ft^3-lb/mol-°R
Temperature for Black Powder Combustion, $\boldsymbol{T}$	1738.7	K	2669.99	°F
Molar Mass of Black Powder, $\boldsymbol{n_{BP}}$	162.1406	g/mol	5.7193413549	oz/mol
Specific Gas Constant, $\boldsymbol{R_s}$	0.051276	J/g-K	0.3066274	ft-lb/slug-°R

3.1.7 Mechanical Stage Separation System (Analysis Task 3)

The innovative stage separation consisted of two curved clasps on either side of the exit nozzle in the upper stage. The clasps can be seen in Figure 3-4.

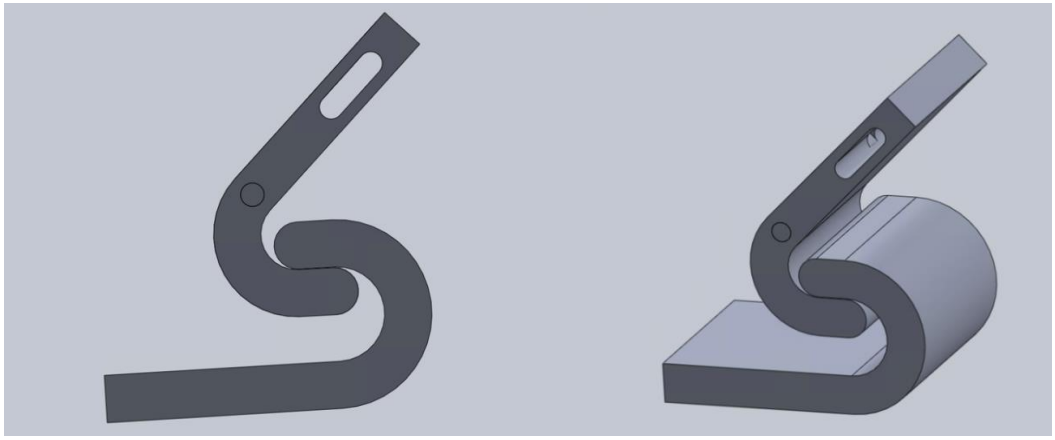


Figure 3-4 Left View and Trimetric View of Circular Clasp SOLIDWORKS Model

Numerous mechanical stage separation methods were considered, including rollers, a pully system, and the removal of shear pins using a linear actuator. Originally, the separation system consisted of two linear actuators on opposing sides of the exit nozzle attached to the inner Blue Tube. The actuators would push down on tabs connected to the lower stage or booster stage of the model rocket. However, because the model rocket can't safely be constructed without shear pins, this method was rejected since the actuators would not produce enough force to shear 3 shear pins. The force required to separate the stages can be calculated using Equations 28 and 29, taking into account nylon shear pins. The typical shear pin made of nylon takes a force of 95N (21.3568 lbf) to shear [100], multiplied by 3 shear pins would result in a total force of 285N (64.0705 lbf) to shear the three shear pins alone [100], [101]. Adding this to the friction fit of the two stages would account for another approximately 10N (2.24809 lbf) of force. The friction fit could be altered using sandpaper if needed, especially if shear pins were used as well, the friction fit would not need to be as tight. The linear actuator is only able to move a load of 20N (4.49618 lbf), so even with two actuators, one on either side of the exit nozzle, 40N (8.99236 lbf) would not be anywhere near enough force to separate the stages.

$$F = F_f + F_{shear} \tag{28}$$

$$F_{shear} = n_{pins} F_{shear\ one\ pin} \tag{29}$$

Another idea incorporated the use of a roller where the actuator would push a roller atop a safety clasp to release the clasp using a pin. An additional option utilized wiring and a pulley system in which wiring attached to the head of the linear actuator and shear pins would pull out two shear pins inserted into the inside of the Blue Tube. Most innovative ideas were centered around various methods of incorporating a linear actuator and converting the vertical motion of the actuator into horizontal motion to separate the two model rocket stages. The SOLIDWORKS model of the ECO LLC linear actuator is displayed below in Figure 3-5.

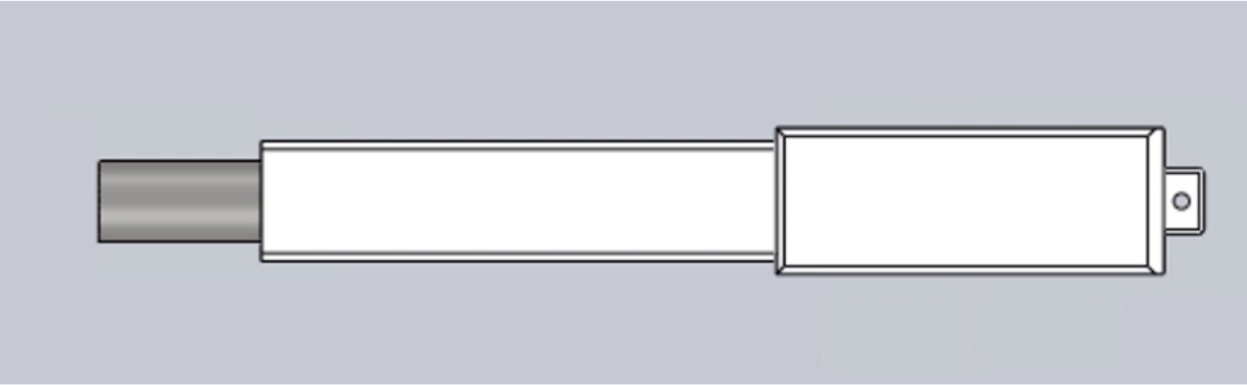


Figure 3-5 SOLIDWORKS Model of the Linear Actuator (Overall Length of 5.2 in)

Of these ideas, a clasp seemed like the most realistic option, as well as the safest method to prevent the stages from unintentionally separating. The design started out with a basic rectangular clasp displayed below in Figure 3-6, but it wouldn't successfully hold the rocket in place because there's no horizontal contact.

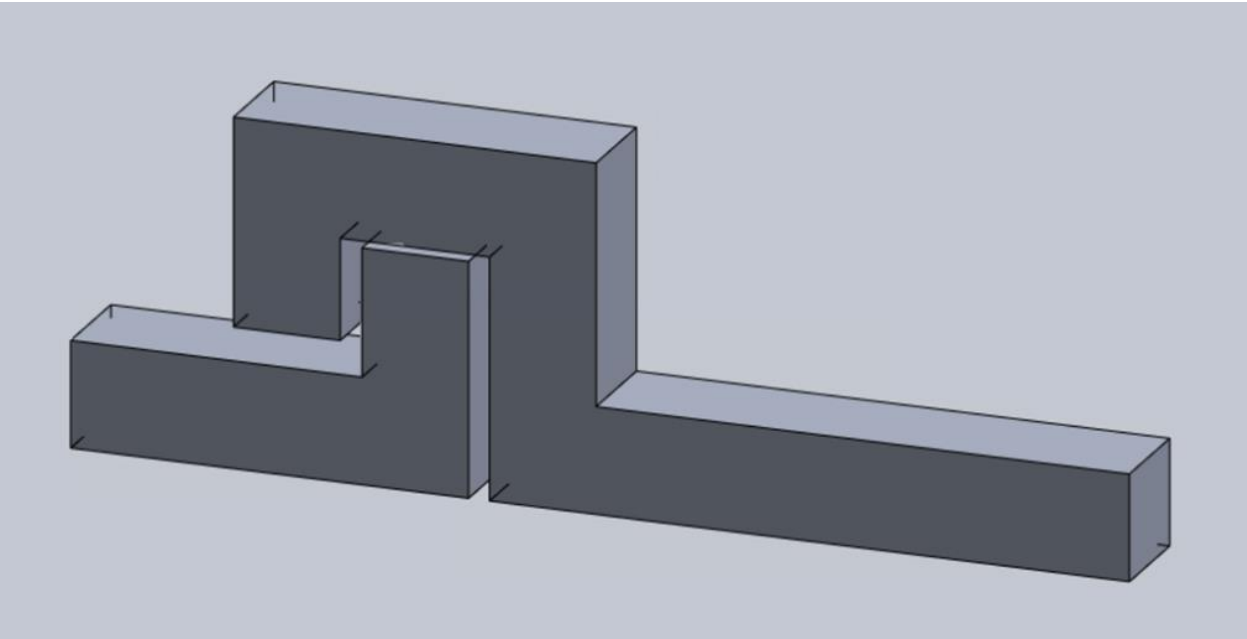


Figure 3-6 SOLIDWORKS Model of Rectangular Clasp (Overall Length of 0.25 in)

Adding an extra turn to the end of the clasp incorporated a horizontal component to prevent the two stages from falling apart. It was important to round the bottom edges to ensure that there's

space for the clasp to move when the actuator applies a downward force. The third iteration was a swirled clasp with a pin in the middle to provide easier rotational separation when the actuator pushed on the leg of the clasp. However, the distance that the linear actuator could plausibly rotate the spiral clasp for it to safely allow the stages to separate was unknown. Additionally, the pin would sit in the middle of the spiral for optimal rotation, but this would make it harder for the two clasps to release. The next iteration was a simple semi-circle clasp, which was then developed into the final iteration of a semi-circle rounded clasp with an upward angled leg to ensure that the two clasps can easily separate. The basic semi-circle clasp didn't provide enough horizontal contact for it to properly hold the stages together, especially if there is jostling during 'flight'. Although the leg is slightly horizontal, it is at a small enough angle that the actuator would still be pushing most of its downward force onto the leg.

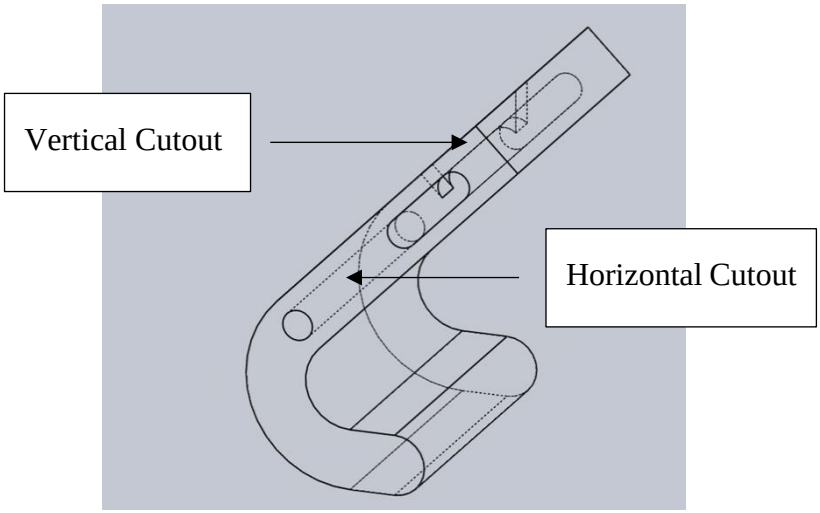


Figure 3-7 Upper Clasp Cross-Sectional Area Illustrating Cutouts for Linear Actuator

As shown above in Figure 3-7, there is a horizontal cutout along the side of the upper clasp arm, as well as a vertical cutout in the top of the arm as well. These cutouts will connect to the head of the linear actuator using wiring and held together via the hole in the linear actuator head as shown below in Figure 3-8. The wiring in the slit in the upper clasp has room to travel along the clasp leg as the actuator head extends, to account for any translational or rotational motion the separation system may encounter. The distance of the horizontal cutout was calculated using trigonometry to properly account for the horizontal component of the linear actuators motion as it extends and is slightly longer than necessary to account for any inaccuracy during 3D

printing. The details of the cutout match the dimensions of the linear actuator head and have straight, rectangular entrance points to compliment the shape of the actuator head which ensures that the pieces will work properly together during motion. Figure 3-8 shows all necessary components of the mechanical stage separation within the Blue Tube of the rocket. The figure incorporates both clasps on opposing sides of the inner airframe tube as well as a centering ring for additional support. The centering ring will be laser cut out of wood to meet exact specifications. The purpose of the clasps is to act as a safety mechanism since the two stages won't have shear pins. To physically separate the stages, two additional linear actuators will sit directly on a centering ring and push the stages apart at apogee. This will be completed using two, 3 inch extending linear actuators to ensure that stages separate completely and successfully.

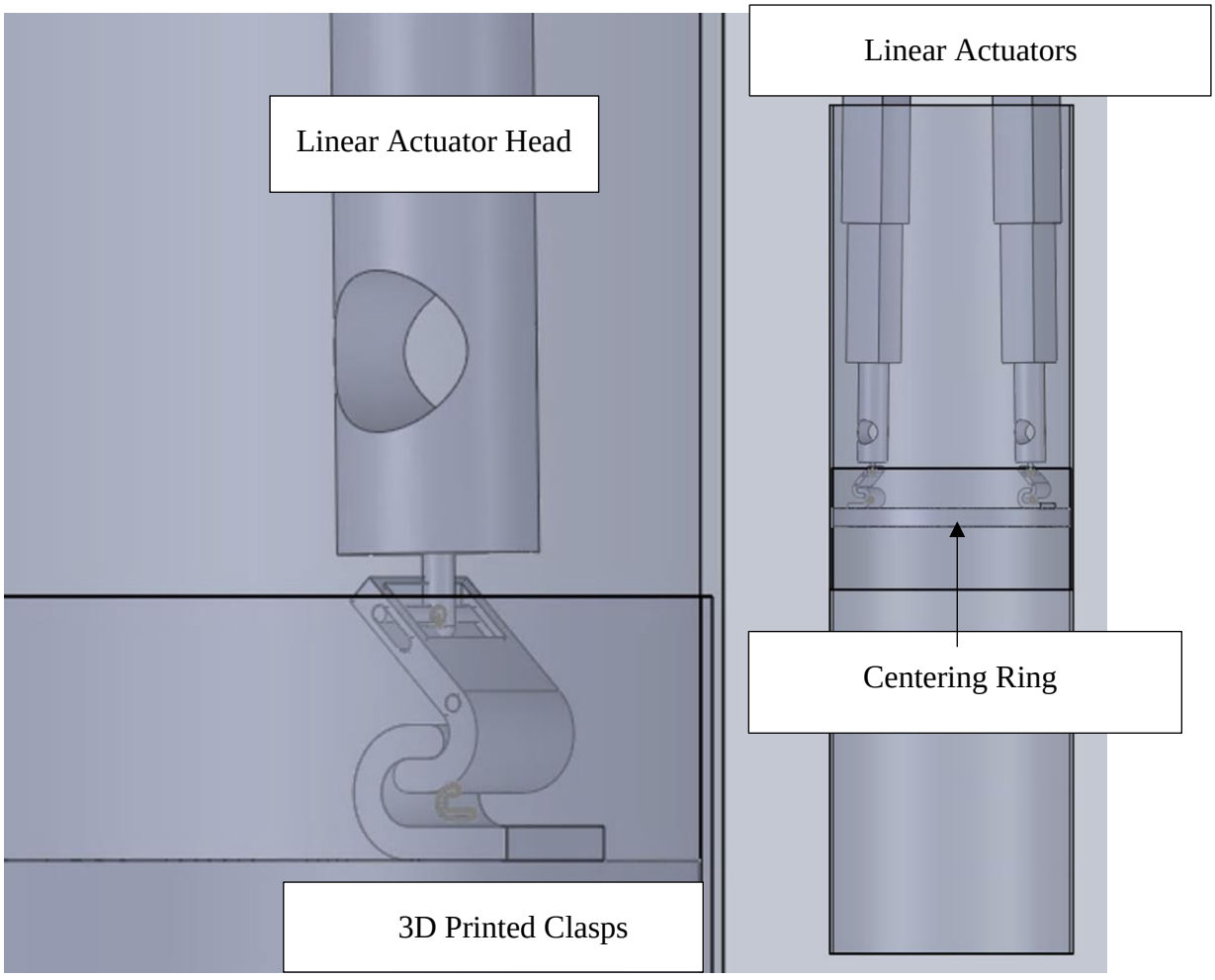


Figure 3-8 Mechanical Separation System SOLIDWORKS Model

3.1.8 Cold Gas Thruster

A cold gas thruster is being used to propulsively land the rocket. Two things are needed to successfully land the rocket. The thrust to weight ratio must be above one at all points during the descent and there must be enough total impulse available to slow the rocket down to an acceptable speed for landing.

The most common gas used in a cold gas thruster system is nitrogen, but due to the low budget it was decided that the rocket's cold gas thruster would use air as nitrogen makes up 78 percent and has a similar performance. The other option considered would be carbon dioxide, but it would be cost prohibitive due to need to use a tank pressurized to a minimum of 3000 psi to fill the rocket's tank.

The cold gas system consists of four main parts: a reservoir, a regulator, a valve, and a nozzle. The configuration, assuming a 134.25 cubic in. tank, is shown in Figure 3-9 In order to keep costs down, commercially available parts for the reservoir, regulator, and valve were selected. Taking inspiration from Vanderbilt's cold gas roll control project [40], it was decided to try to design the system around paintball parts. Paintball retailers offer high pressure air tanks that come with a regulator that outputs 800 psi. Any option for a valve would be one that is commercially produced. The nozzle will be machined or additively manufactured.

Figure 3-9 Cold Gas System Configuration

Two different types of tanks were considered. Paintball tank manufacturers make aluminum tanks rated for 3000 psi (20.684 MPa) and carbon overwrapped pressure vessels (COPV's) rated for 4500 psi (31.026 MPa). The COPV's were the preferred tank as they are lighter, rated for higher pressure, and come in larger sizes. The only problem is that the COPV's have a 4.5-inch (114.3 mm) diameter and the rocket uses 4-inch (101.6 mm) Blue Tube. This limited the option for tanks to the aluminum tanks that are 3.5-inch (88.9 mm) in diameter.

After a calculating the impulse required for landing the rocket with zero-velocity landing (366.58 Ns), a soft landing (5 m/s) (332.55 Ns), and a hard landing (10 m/s) (253.12 Ns) and the impulse available from the paintball tanks, it became apparent that the paintball tanks did not have enough total impulse to land the rocket. The 90 ci 4.5k psi, 68 ci 4.5k psi, 62 ci 3k psi, and 48 ci 3k psi tanks had 242.97 Ns, 183.58 Ns, 108.85 Ns, and 84.27 Ns. It was then decided to design a custom tank that would have enough total impulse and fit the body tube.

A tank pressure of 4500 psi was selected due to paintball tank regulator being rated for up to 4500 psi. Ti-6Al-4V was chosen as the material for the construction of the tank. A flow coefficient of 0.04 was assumed for the valve for calculations [102]. The tank was design for a zero-velocity landing based on the best mass estimate available.

3.2 Analysis

In this section, the result of the PTSS subteam's analysis is presented along in addition to work which assesses its validity.

3.2.1 Motor Performance Analysis (Analysis Task 1) Results

This section presents the results of the motor performance model that was developed using Cantera. The MATLAB code used to evaluate the combustion reaction and subsequent motor performance parameters can be found in Appendix D: Cantera Model MATLAB Code. After performing the equilibrium condition in Cantera, the mole fraction for each dominant species was found. Since a similar analysis was performed using NASA's Chemical Equilibrium with Applications (CEA) calculator [87], it was possible to validate the Cantera model results as shown in Figure 3-10.

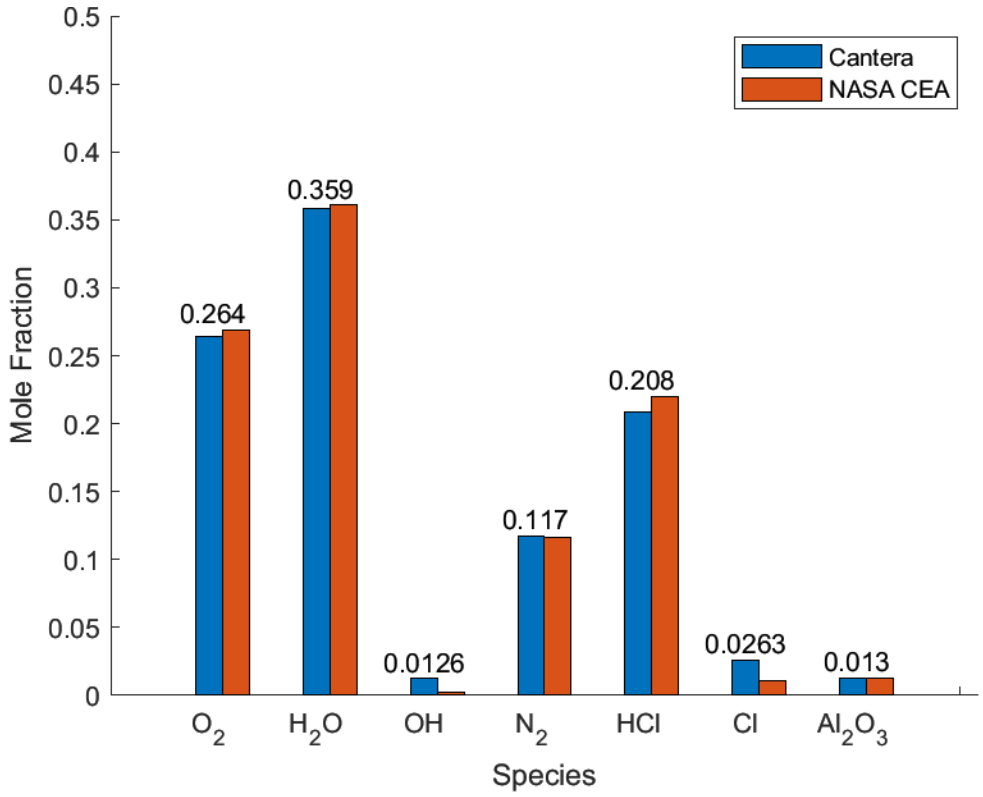


Figure 3-10 Mole Fractions of Combustion Products from each Model

The results for the mole fractions calculated using the Cantera and NASA model appear to agree. The differences observed in the mole fractions appears to primarily be the result of the NASA model considering many trace species such as elemental oxygen which was found to have a mole fraction of 0.00012. Properties of the combustion gas were retrieved and compared for both models as well. Note that the NASA CEA model required a pressure input so the pressure from the Cantera model was used. These results are summarized in Table 3-9.

Table 3-9 Combustion Gas Properties from Cantera and NASA CEA Models

Property	Cantera Model	NASA Model
Equilibrium Temperature	2,460.4 K (4,428.72 R)	2,044.05 K (3,679.29 R)
Pressure	1.7139 MPa (248.58 psia)	1.7139 MPa (248.58 psia)
Density	2.3676 kg/m^3 (0.1478 lb/ft^3)	2.9114 kg/m^3 (0.1817 lb/ft^3)

Specific Heat at Constant Pressure	1,587.7 J/kg-K (0.3792 Btu/lb-F)	1,775.4 J/kg-K (0.4240 Btu/lb-F)
Specific Heat Ratio	1.2275	1.2069
Mean Molecular Weight	28.2591 g/mol (28.2591 lb/lbmol)	28.484 g/mol (28.484 lb/lbmol)

The combustion equilibrium temperature for the Cantera model is noticeably higher than the NASA model. This difference may result from assumptions made for the Cantera model. However, due to the lack of documentation for the NASA model it was not possible to compare the assumptions between the models. The nozzle geometry was provided to both the Cantera and NASA models, and an isentropic 1-D nozzle flow analysis was performed using the Cantera results. These results were compared to results from the NASA model and published performance data for the K360 motor [82]. These results are summarized in Table 3-10.

Table 3-10 Motor Performance Parameters for Cantera and NASA Analyses

Property	Cantera Model	NASA Model	Motor Test Data
Exit Mach Number	3.1479	3.199	-
Exit Temperature	1,156.7 K (2,082.06 R)	922.44 K (1,660.39 R)	-
Exit Pressure	29,190 Pa (4.23 psia)	27,533 Pa (3.99 psia)	-
Exit Density	0.0858 kg/m^3 (0.00535 lb/ft^3)	0.1047 kg/m^3 (0.00653 lb/ft^3)	-
Exit Velocity	2,790.7 m/s (9,155.8 ft/s)	575.1 m/s (1,886.811 ft/s)	-
Thrust	471.73 N (106.05 lbf)	280.75 N (63.11 lbf)	405.151 N (91.08 lbf)

Specific Impulse	253.75 s	187.53 s	184.4 s

For the purpose of the analysis the maximum thrust of the K360 motor was compared to the Cantera and NASA models. The Cantera and NASA models appear to agree with the motor test data with the exception of thrust and exit velocity. The thrust and exit velocity of the NASA model are significantly lower than expected. This is possibly due to the NASA model only being provided dimensions for the diverging section of the nozzle when dimensions for both the converging and diverging sections of the nozzle could have been used as inputs. The converging geometry of the nozzle wasn't included in the NASA model due to the lack of information on the motor being analyzed. It is also notable that the thrust and specific impulse of the Cantera model appear to be higher than the motor test data. This difference is likely the result of the assumed adiabatic combustion condition that resulted in a higher combustion equilibrium temperature. The thrust may also be higher for the Cantera model due to the idealized isentropic flow equations that were used to calculate the thrust that do not include viscous or boundary layer losses. The specific impulse may also differ from assuming complete combustion and no reactions amongst the product species.

3.2.2 Thermal Model (Analysis Task 2) Results

This section presents the results of the thermal and velocity distribution models that were developed in COMSOL using inputs from the Cantera model. The boundary conditions, domains, material properties and initial conditions were discussed in Section 3.1.4. As the burn time of the motor was provided at 3.5 seconds [82], the results in this section were evaluated at 3.4 seconds so that a thin boundary for the fuel grain could be included in the model and the model would be

evaluated towards the end of the burn when the casing is expected to be the hottest. The temperature distribution of the motor is shown in Figure 3-11.

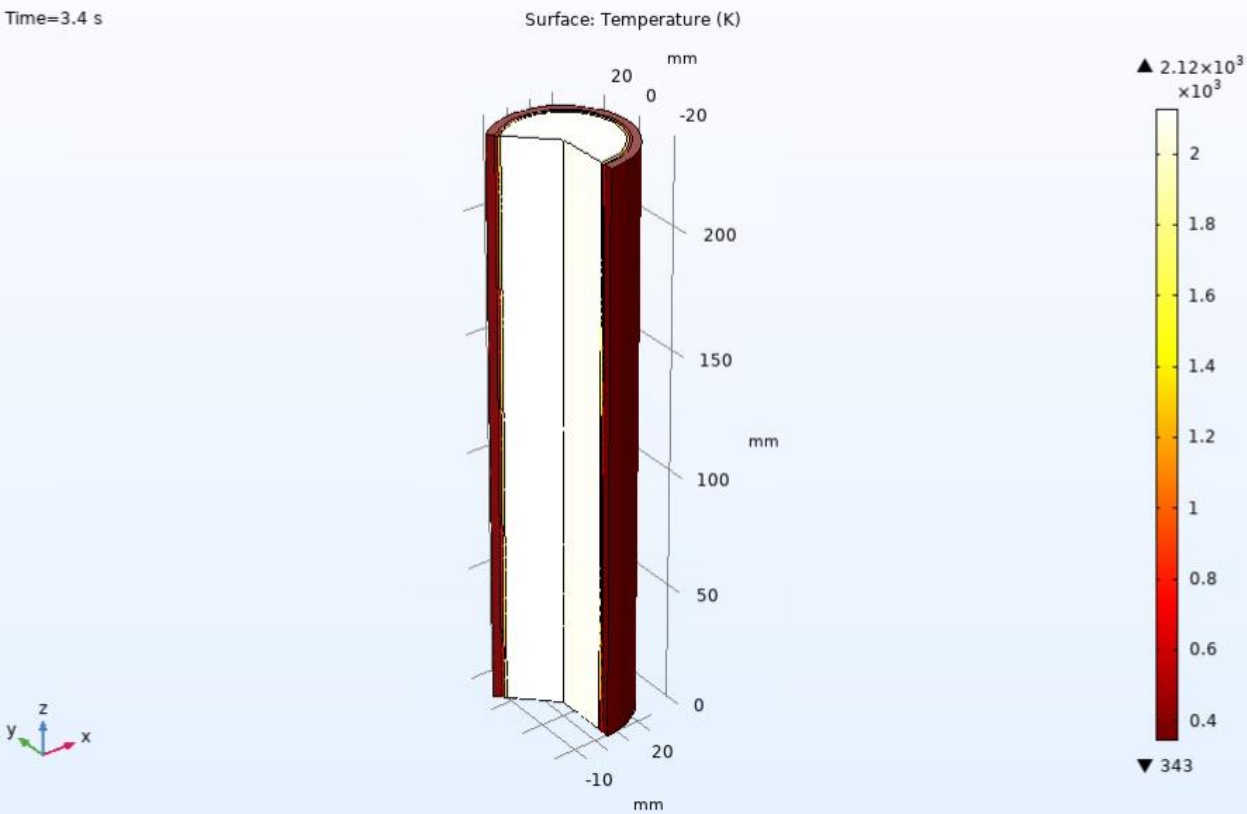


Figure 3-11 Motor temperature distribution at t = 3.4 s

COMSOL is able to provide the value of the temperature at a point on the surface when it is clicked on. Using this feature in COMSOL, the temperature of the fluid combustion temperature was determined to be approximately 2,093 K (3,767.4 F). It is important to note that the combustion temperature estimated here is significantly less than the combustion equilibrium temperature that was found using Cantera. This difference is due to the zero-heat loss assumption used in the Cantera model. The temperature on the outside of the motor case was found to be 343 K (617.4 R). This is an important result as the National Fire Protection Association (NFPA) code 1125 does not allow rocket motor manufacturers to exceed a temperature above 498 K (887.4 R) on the exterior of the motor case during or after burning [103].

Since the temperature distribution model was coupled with a fluid flow model, the velocity distribution can be observed at a time of 3.4 seconds. The results of this model are shown in Figure 3-12.

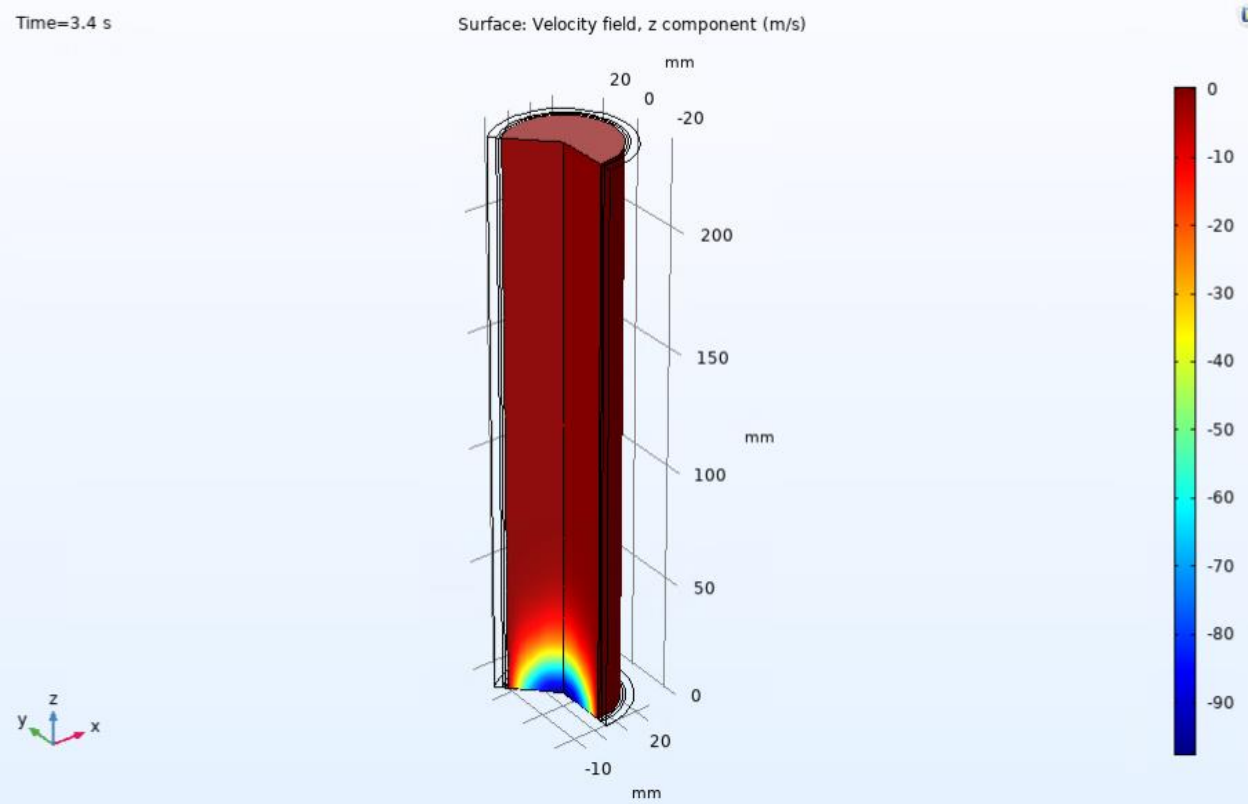


Figure 3-12 Velocity distribution in the motor at t = 3.4 s

The fluid is accelerated through the combustion chamber which results in the velocity distribution being higher at the outlet. In COMSOL the negative velocity sign indicates that the fluid is traveling in the negative z-direction which is towards the defined outlet boundary. The direction of the flow can be verified by showing the streamlines of the flow as shown in Figure 3-13.

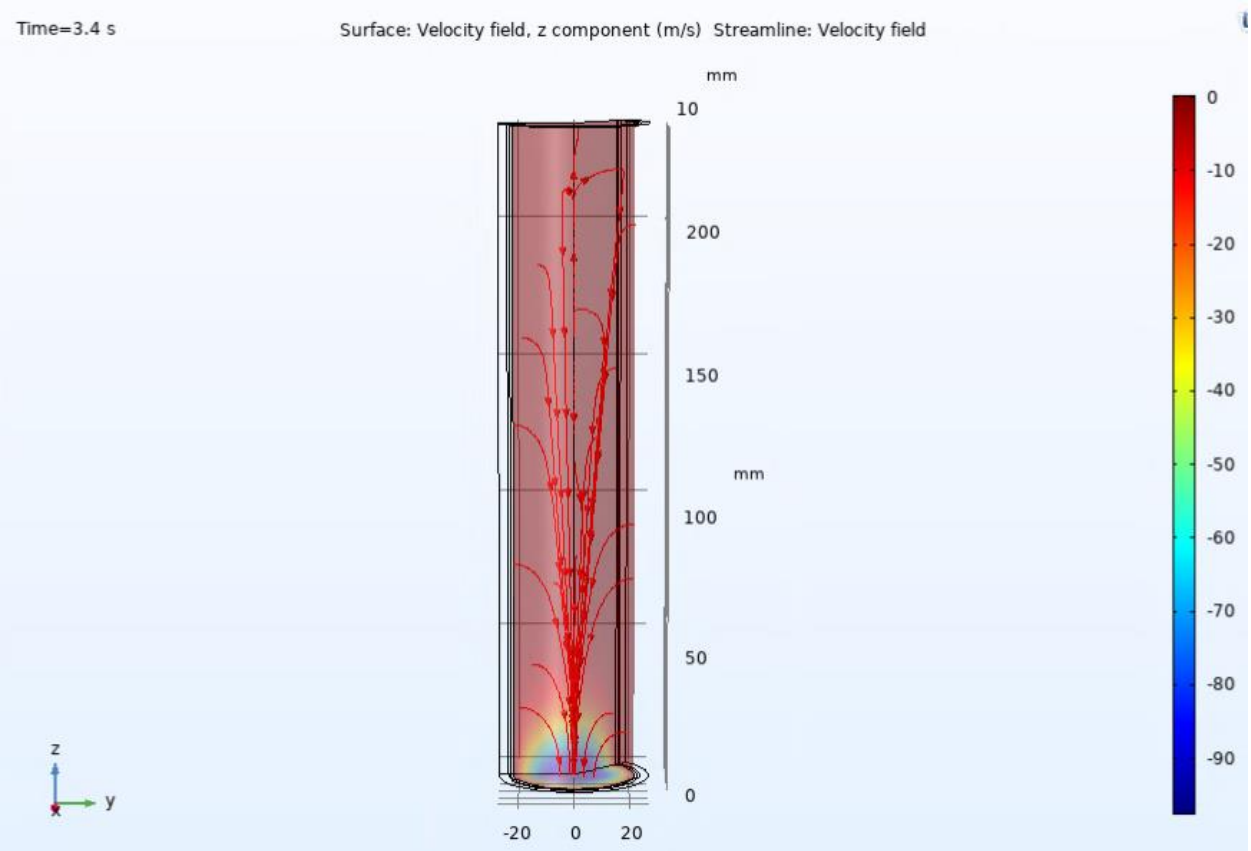


Figure 3-13 Velocity Streamlines for the Velocity Distribution in the Motor Case at t = 3.4 s

The streamlines shown in red point in the direction of the flow. This confirms that the fluid travels from the inlet boundary defined on the fuel grain wall and then is accelerated out through the outlet boundary which is position at the outlet of the fuel grain or the inlet of the rocket motor nozzle.

3.2.3 Mechanical Stage Separation Model (Analysis Task 3) Results

This section presents the results of the mechanical stage separation. A motion study was completed to ensure that the linear actuators properly separated the two stages. The motion study utilized all four linear actuators, the two sets of 3D printed clasps, 12V battery pack, and a small subsection of Blue Tube. The clasps were 3D printed in the Higgins laboratory using the resin "tough". Because of how small the clasps are, the Higgins printers were optimal since they can print such small parts. Table 3-11 summarizes the key aspects of the separation system.

Table 3-11 Mechanical Separation System Masses Per Unit

Property	Value
Mass of 1-inch actuator	0.0689 kg (0.1519 lb)
Mass of 3-inch actuator	0.1131 kg (0.2494 lb)
Time of 2-inch actuator extension	3.333 s
Total mass of clasps	0.0004 kg (0.000882 lb)
Mass of total system	0.1824 kg (0.4022 lb)

The four linear actuators are equally spaced around the Blue Tube. One set extends only 1-inch and is responsible for separation the clasps, the safety mechanism that holds the two stages of the model rocket together during flight. The second set of linear actuators extends 3-inches and is responsible for separating the two stages of the model rocket once the clasps have been released. This set of linear actuators are mounted using a bulkhead in the booster stage and separate the two stages by pushing the centering ring epoxied to the upper stage. Both sets of linear actuators rest inside the Blue Tube opposite to each other so that their distributed force is even.

The Blue Tube was measured and set to a 2-inch overlap and sanded down to loosen the friction fit and allow for easier separation using the actuators. The bulkhead and centering ring were not able to be 3D printed in time, so for the sake of the motion study, the linear actuators were mounted on the inside of the Blue Tube using thin metal wire. Similarly, the upper clasps that would otherwise sit on the centering ring instead were epoxied to two tabs that were then epoxied to the inside of the upper stage. This process was repeated for two additional tabs that the linear actuators could push on the separate the stages.

For the motion study, the parts were assembled by first epoxying the tabs with the upper clasps to the inside of the upper stage and properly mounting the linear actuators in the booster stage. The two tabs used as a placeholder for the centering ring were also epoxied to the upper stage at the same time. These are the tabs that the two 3-inch actuators push on in order to separate

the stages. From here, the two separate pieces of Blue Tube were attached. Prior to epoxying the tabs and mounting the actuators, the Blue Tube was marked to show how far to overlap the two stages to accurately mimic the overlap of the upper and lower stages of the actual model rocket. *Figure 3-15* below helps to illustrate how the motion study was completed by demonstrating the stage separation assembly.

It is important to note that the actual upper and lower stages were not used for the motion study, but instead smaller sections of Blue Tube just so that the separation system could be tested. The Blue Tube used for the model rocket was not delivered in time to be used for the separation test. Once the actuators, clasps, and tabs were properly placed, the two stages were connected using the markings showing how far to overlap the Blue Tube. The clasps can be secured by simply reaching into the Blue Tube and placing the clasps together.

This is a relatively easy process due to the use of wire that connects the upper clasp to the 1-inch linear actuators. However, during a flight test, this process would need to be altered since it would not be possible to reach into the Blue Tube and place the clasps if the cold gas thruster were already mounted inside the model rocket. Using the linear actuator controllers, the actuators were extended. Before the stages can separate, the clasps must be released, but all actuators were signaled at the same time since the 3-inch actuators take 3.3 seconds to extend enough to separate the stages whereas the 1-inch actuators separate the clasps in only 0.12 seconds. The motion study was successfully carried out and successfully separated the two stages. Figure 3-14 shows how the clasps are attached to the actuator head and the closed and open position of the safety clasps.

Figure 3-14 Clasp Safety Mechanism Latched and Unlatched

Although the rocket couldn't physically be launched, *Figure 3-15* shows how the mechanical separation system would be assembled in the rocket. First the actuators would be mounted in the bulkhead and placed in the upper stage of the model rocket. This includes all four linear actuators. The upper clasps are secured to the two 1-inch linear actuators using wiring that can be inserted into the actuator head and clasp cutouts. Separately, the centering ring with the two lower clasps epoxied to it can be placed onto the clasps by fitting the two sets of clasps together. Once placed and hanging from the clasps, the centering ring could be placed into the booster stage and epoxied- or bolted for reusability purposes. Similarly to how the Blue Tube was marked in the motion study, the exact location of where the centering ring would need to sit inside the Blue Tube could be marked so that it could easily and accurately be secured at the launch site.

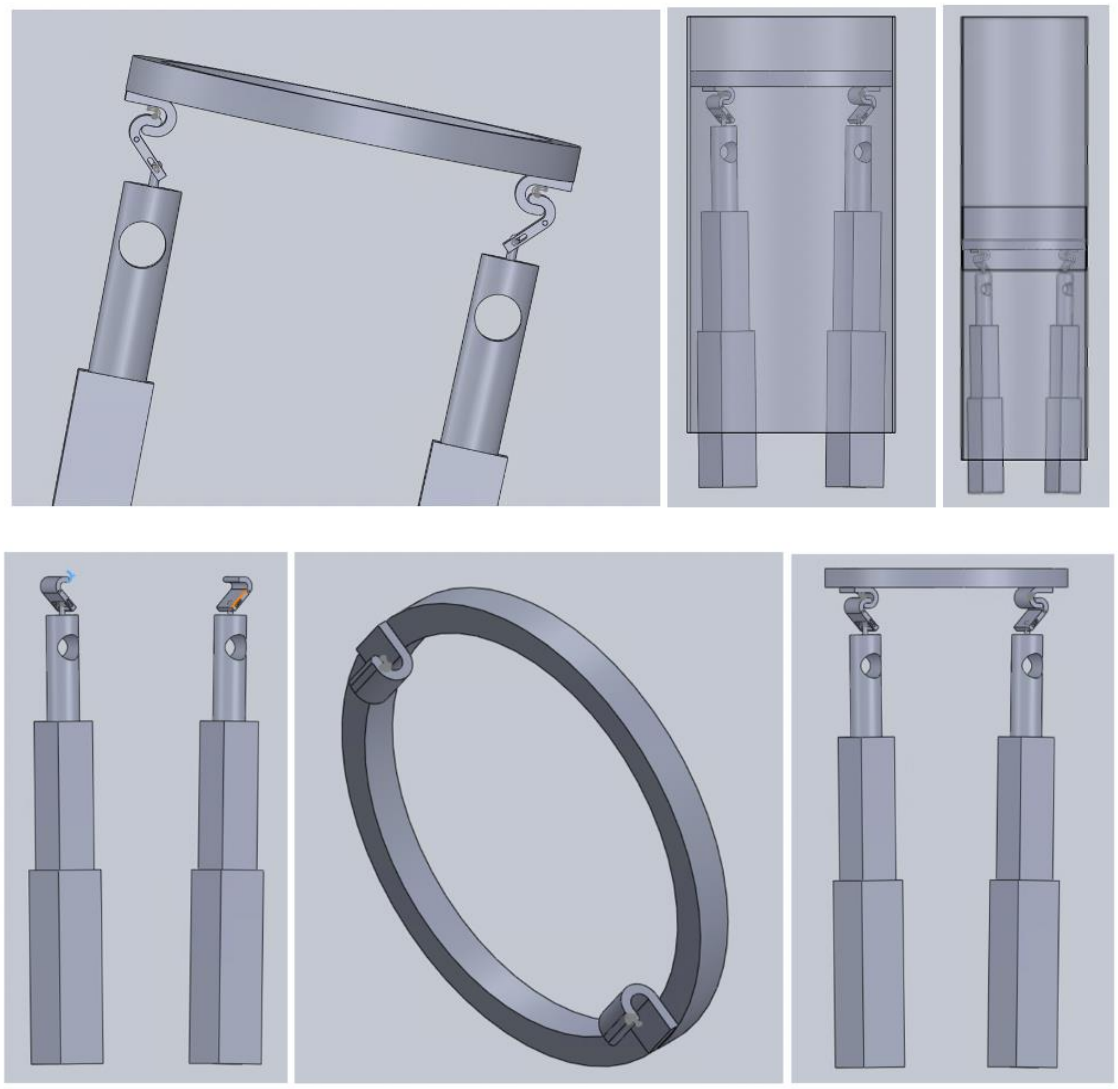


Figure 3-15 Assembly of Separation System

3.2.4 Cold Gas Thruster Descent Model (Analysis Task 4) Results

This section presents the analysis of the cold gas thruster descent model. We will discuss how we came the conclusion that off-the-shelf paintball high pressure air tanks were not suitable to serve as the propellant tank for the system. We will also describe the design of a tank that can meet the mission requirements.

The first step in evaluating the paintball tanks was to find the mass of air that could be used before the nozzle unchokes. The mass of the air was calculated by taking the mass of air needed to fill the tank at the rated and subtracting from it the mass of air needed to fill the tank at 800 psi (5.516 MPa). Paintball tanks are rated in Imperial units, but the calculations were done in SI units. Psi was converted to Pascals and cubic inch (ci) was converted to cubic meters. The first step of finding the mass of air (the propellant) was finding the density for each pressure in kg/m^3. Eq. 25 the equation for density using a compressibility factor, Z [104], for high pressure air, was used to find the density at 4500 psi (31.026 MPa), 3000 psi (20.684 MPa), and 800 psi (5.516 MPa). The compressibility factor for 800 psi is Z=0.9867, for 3000 psi is Z=1.030414, and for 4500 psi is Z=1.1136 The specific gas constant, R, is found by dividing the universal gas constant by the molecular mass in Eq. 26.

$$\rho = \frac{P}{ZRT} \tag{25}$$

$$R = \frac{R_{unv}}{M} \tag{26}$$

The four tanks size chosen to explore are a 48 ci (7.868E-4 m^3) tank rated for 3000 psi (20.684 MPa), a 62 ci (1.0E-3 m^3) tank rated for 3000 psi (20.684 MPa), a 68 ci (1.114E-3 m^3) tank rated for 4500 psi (31.026 MPa), and a 90 ci (1.475E-3 m^3) tank rated for 4500 psi (31.026 MPa). The total available mass of air was found by multiplying the density of air at the rated pressure by the tank volumes as shown in Eq. 27. The remaining mass of air was found by multiplying the density of air at 800 psi (5.516 MPa) by the volume of the tank as shown in Eq. 27.

$$m = \rho V \tag{27}$$

The mass of air that could be used before the nozzle unchokes is shown in Eq.32 by subtracting the remaining mass of air from the total available mass of air. The mass of air available

that can be used before the nozzle unchokes is shown in Eq. 28. The remaining air mass is found by taking the volume of tank multiplied by the density of air at 800 psi.

$$m_{AirMass} = m_{TotalAirMass} - m_{RemAirMass} \quad (28)$$

Table 3-12 Mass of Air in Tank Available for Use Before Nozzle Unchokes Based on Tank Size and Pressure

Tank Size	Air Mass (g)
48 ci 3000 psi	135.4
62 ci 3000 psi	174.9
68 ci 4500 psi	295
90 ci 4500 psi	390.5

The next steps in designing the designing the cold gas thruster including finding the mass flow rate through the throat of the nozzle from which the thrust and the burn time can be found. To find the mass flow rate, it is an iterative process due to fact that there is a pressure drop across the valve which is determined using the mass flow rate. The pressure drop is determined by the flow rate, but the mass flow rate is determined by the regulator pressure minus the pressure drop. The mass flow rate is calculated in Eq. 29 from a guess of the pressure drop. The mass flow rate is used to calculate a volumetric flow rate (Q) as shown in Eq. 30 in Standard Cubic Feet Per Hour (SCFH) as the valves use Imperial units. The density used was calculated from Eq. 25 using a compressibility factor, Z = .989 [104] as there is not much of a pressure drop and the Z chosen is close enough for most pressures. The pressure drop is calculated in Eq. 31 [105] from the volumetric flow rate (Q) and the flow coefficient, C_v, which is assumed to be 0.04 [102] as that is what the flow coefficient is for most small valves that are rated for 800 psi (5.516 MPa). The

calculated pressure drop is used in Eq. 29 [22] to find the true mass flow rate. $(P_i - \Delta P)$ is substituted in for P_1, the chamber pressure

$$\dot{m} = A_t(P_i - \Delta P)k\frac{\sqrt{\left[\frac{2}{k+1}\right]^{\frac{k+1}{k-1}}}}{\sqrt{kRT_1}} \quad 29$$

$$Q = \frac{\dot{m}}{\rho}\left(\frac{35.315\ ft^3}{1\ m^3}\right)\left(\frac{60\ sec}{1\ min}\right)\left(\frac{60\ min}{1\ hr}\right) \quad 30$$

$$\Delta P = \left(\frac{T\ SG}{P_i}\right)\left(\frac{Q}{1360\ Cv}\right) \quad 31$$

The mass flow rate was calculated for a throat diameter from 2.9 mm to 3.5 mm. The throat area is calculated using Eq. 32.

$$A_t = \frac{\pi D_t^2}{4} \quad 32$$

The thrust can be calculated from the mass flow rate multiplied by the exhaust gas velocity assuming a nozzle expanded for sea-level. The exhaust velocity can be calculated by Eq. 33. P_2 is the nozzle exit pressure. P_1 is nozzle chamber pressure. T_1 is the chamber temperature. k is specific heat ratio. R is the specific gas constant. The thrust calculated by Eq. 34 assuming the exit pressure and atmospheric pressure are the same.

$$V_e = \sqrt{\frac{2k}{k-1}RT_1\left[1 - \left(\frac{P_2}{P_1}\right)^{k-1/k}\right]} \quad 33$$

$$F = \dot{m}V_e \quad 34$$

A correction factor was applied to the thrust for a conical nozzle. The correction factor was calculated for a 15-degree half angle nozzle in Eq. 35. The final thrust is calculated in Eq. 36 by multiplying the ideal thrust by the correction factor.

$$\lambda = \frac{1}{2}(1 + \cos(\alpha)) \quad (35)$$

$$F_{Corrected} = F\lambda \quad (36)$$

The burn time is lastly calculated by dividing the mass of air that could be used before the nozzle unchokes when the reservoir pressure drops below 800 psi by the mass flow rate as shown in Eq. 37.

$$t_b = \frac{m_{AirMass}}{\dot{m}} \quad (37)$$

Lastly the required burn times were calculated for a hard landing, a soft landing, and a zero-velocity landing based on the calculated thrusts from a descent model that calculates the landing velocity from the drag, thrust, and burn time. A plot was created to show the thrust vs burn time in Figure 3-16.

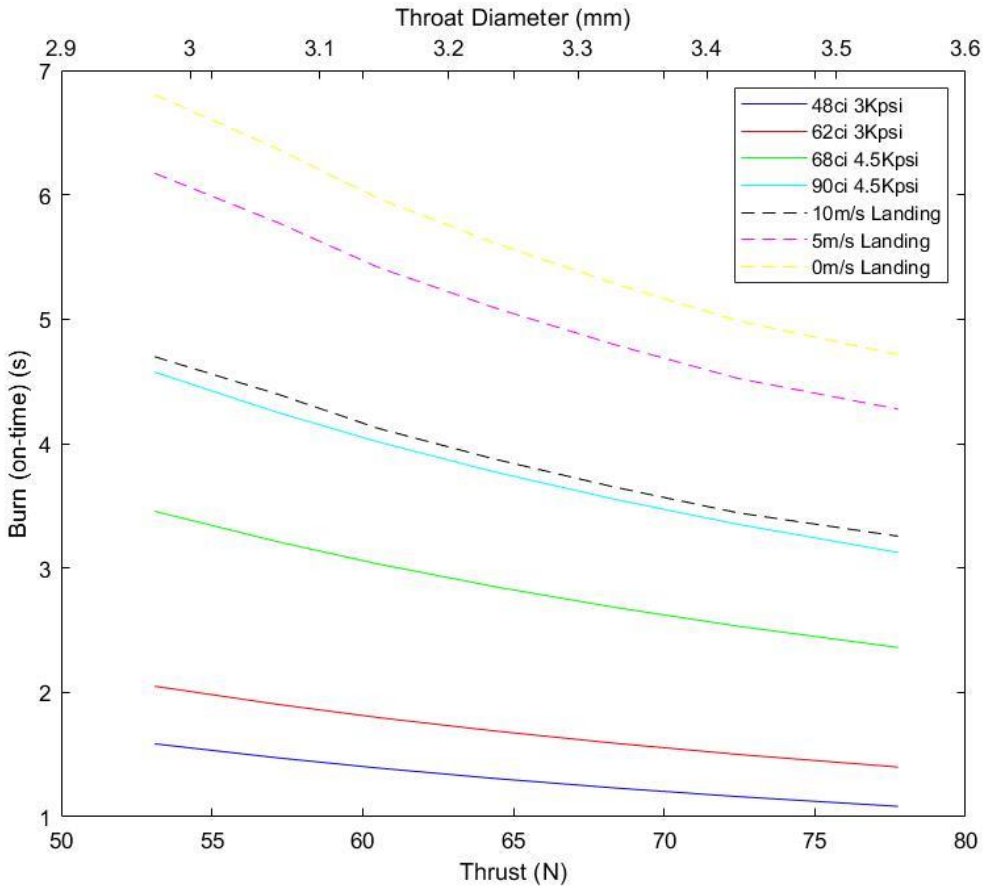


Figure 3-16 Thrust vs. Burn Time

In Figure 3-16, the thrust is found as a function of the nozzle diameter. The mass flow rate which is used to calculate thrust is used to find the on-time based on the tank size and pressure. The thrust is used with the aerodynamic model to find the on-time required for a few desired landings. The solid lines are the required on-time for the commercially available paintball tanks. The dashed lines are the required burn time for each type of landing. As shown in the Figure 3-16, the required burn time is greater than the largest paintball tank can provide and significantly greater than what the largest aluminum tank can provide. This shows that even the largest available paintball tank would not be able to support a safe landing of the rocket.

The conclusions drawn from the plot were that it was necessary to design a custom tank. The nozzle throat diameter was chosen to be 3.5mm to give enough thrust even if the mass of the rocket increased. The calculated pressure drop across the valve and mass flow rate are used from the analysis of the paintball tanks as the same operating pressure will be used. From the plot, 5.22 seconds of burn time are needed to land the rocket with zero-velocity with 3.5mm nozzle throat diameter. The amount of air required to flow through the nozzle can calculated from multiply the mass flow rate by the required burn time. The air required amounts to 0.58 kg of air.

$$m_{air\,req} = \dot{m} t_b \tag{38}$$

The volume of the tank required is calculated from the mass of air required and densities of air at 4500 psi (storage pressure) and of air at 800 psi (regulator pressure). The volume of the tank required is 0.0022 m^3, and total mass of the air required is 0.7256 kg.

$$V_{tank} = \frac{m_{air,req}}{\rho_{air,4500} - \rho_{air,800}} \tag{39}$$

$$m_{air,total} = V_{tank}\rho_{air,4500} \tag{40}$$

The tank was chosen to be a cylindrical tank with spherical endcaps. The radius chosen for the tank was 0.046 m (1.811 in), slightly smaller than the inside of the Blue Tube. Using the radius and the volume of the tank required, the length of the tank can be found. The wall thickness is also calculated with a factor of safety of 2.

$$L_{tank} = \frac{V_{tank} - \frac{4}{3}\pi {r_{tank}}^3}{\pi {r_{tank}}^2} \quad (41)$$

$$t_{wall} = FoS\left(\frac{P_{tank} r_{tank}}{\sigma_{ult}}\right) \quad (42)$$

The length of the cylindrical walls of the tank are 0.2714 m (10.685 in). The thickness of the walls was calculated to be 3.17E-3 m (0.125 in). The tank designed will land the rocket with zero-velocity provide thrust up until the tank drops below 800 psi, the pressure of the regulator.

4 Flight Dynamics Analysis

The Flight Dynamics Analysis subteam goal for the project was to run multiple analysis for the vehicle in flight. The major focus of the analysis was to simulate the flight of the rocket and examine the aerodynamic loads during the flight. The two major tools used for the analysis were MATLAB and Ansys Fluent. The MATLAB code was used to achieve a six degree of freedom simulation. Ansys Fluent was used to analysis the aerodynamic loads for the rocket. This data was than input into the simulation to achieve a realistic flight for the rocket. In addition to analysis the FDA subteam was responsible for designing the fins, the avionics bay, and integrating the power system for the rocket.

4.1 Methodology

4.1.1 Stability Fin Design

Research conducted on the aerodynamics of different fin shapes contributed to the final design. It was found that airfoil fins help to reduce the drag on the rocket and that the most aerodynamic fin shape was elliptical [106]. However, the elliptical fin shape was not selected for our rocket due to the difficulty of manufacturing the fins to be both elliptical and has an airfoil cross section. Instead, the delta clip shape, another option that has low drag, was selected [106].

This option allowed us to pick a NACA airfoil that would work best for what we were looking for in the stability fins.

When planning the dimension of the stabilizing fins, the selections were made by examining the stability of the entire rocket. The stability of our rocket was simulated in OpenRocket and was found to be less than one. As mentioned in Section 1.1.3.1 stability for model rockets less than one is not a desirable value and the design should be modified to increase the stability. The fins selected were picked to help increase the stability of the rocket. Once the dimension for the fin, including the maximum thickness for the stability fin, was found it was then possible to select a NACA airfoil for the fin. The NACA 0004 was selected because for the camber length from OpenRocket the maximum thickness would closely match the dimension used in OpenRocket. Additionally, due to us selecting a thin airfoil with a maximum thickness of 4%, this helps to reduce the drag on the stability fins. Reducing the drag was an important factor we had to consider because some parts of the rocket have high drag in the ascent configuration, such as the grid fins. The schematic for the stability fins is shown below in Figure 4-1. The stability fins will be 3D printed using Acrylonitrile butadiene styrene (ABS) and bolted to the rocket.

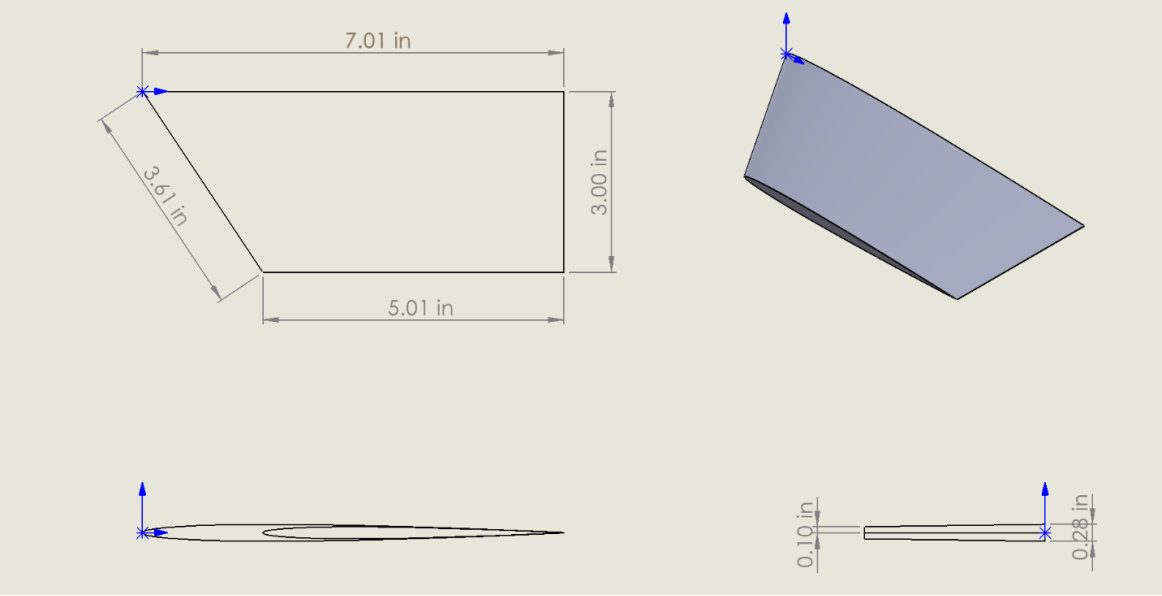


Figure 4-1 Stability Fin

4.1.2 Avionics Implementation

For this project, we are actively controlling our rocket's decent. In order to accomplish this goal, we need multiple electronic components. Most prior MQP's have relied on passive control recovery systems and only need the avionics bay to set off stage separation. In those cases, a Raven 4 altimeter [107] was enough to accomplish their goals of recording telemetry and triggering deployment events. However, due to our design having grid fins that will be actuated as needed and a cold gas thruster intended to slow descent there are multiple parts required for an avionics bay to work successfully. Figure 4-2 show the wiring of all of the components of the avionics bay.

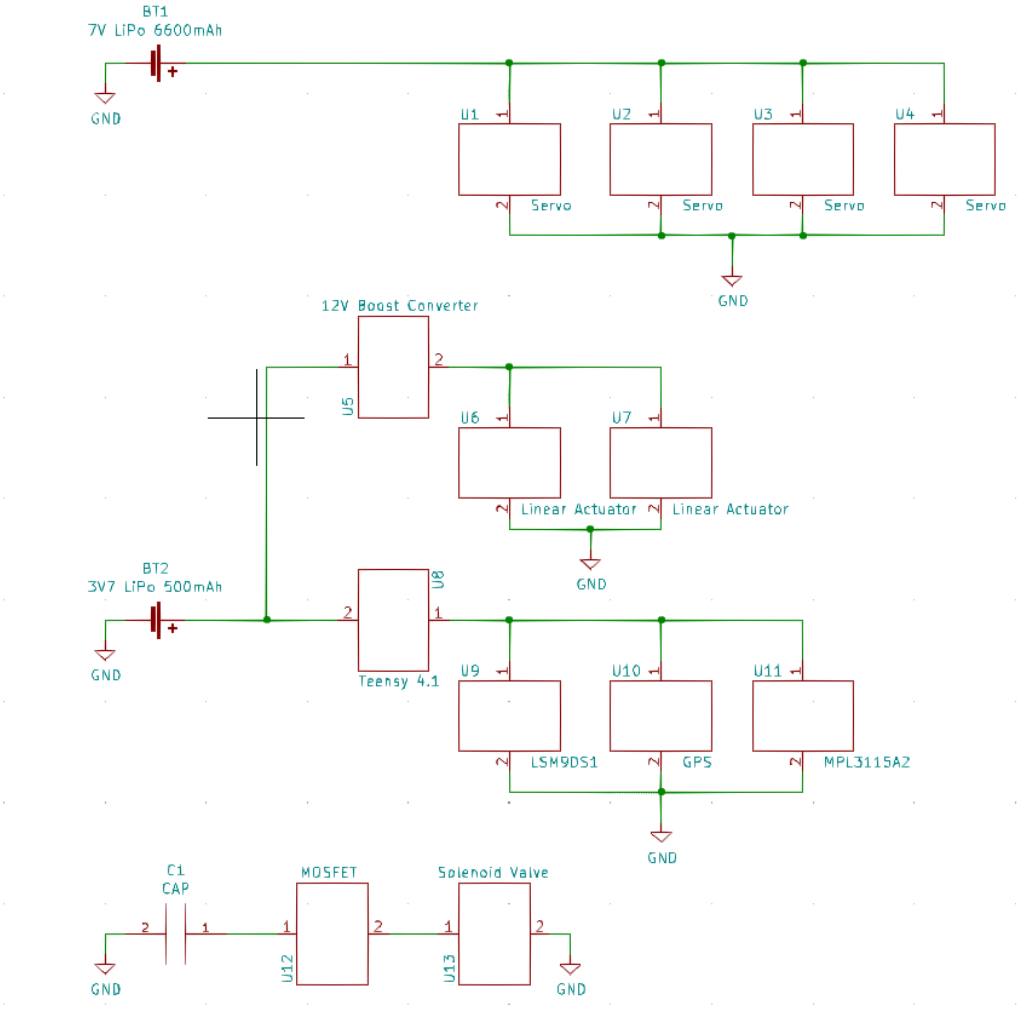


Figure 4-2 Avionics Bay Electrical Diagram

A microcontroller can be used to actuate the servos in order to stabilize the rocket body on descent through rotating the grid fins in addition to turning on the cold gas thruster. The microcontroller selected is a Teensy 4.1 [72]. This product has a powerful processor that will enable us to perform floating point math with high accuracy and precision. This is important to avoid overcorrecting or under correcting the rotation needed on the grid fins and ensuring that our cold gas thruster valve will be opened at the correct moment. The controller is programed in an Arduino environment which makes it easy to program and will be helpful for implementing the control laws. In addition, the controller will be able to hold the data collected from the flight on an SD card. This will be useful in the future to compare our simulated values to the actual values found from a test flight.

The next set of components that are important in our avionics bay are sensors. Sensors are used to monitor the state of the vehicle. They are what will determine when stage separation occurs in addition to indicating when a grid fin needs to be rotated in order to stabilize the rocket and determining the moment the cold gas thruster will be actuated. The three sensors used are a barometer, a global positioning system (GPS), and an inertial measurement unit (IMU). All of our sensors were selected from Adafruit in order to simplify ordering. The barometer selected was an MPL3115A2 [108]. This type of sensor is used to determine when the pressure of the air changes and will be used to determine the altitude of the rocket. The next sensor chosen is the GPS, model PA6H [109]. This will help us with determining the position of our rocket. The IMU chosen is an LSM9DS1 [110]. IMUs are a sensor that include an accelerometer, a gyroscope, and a magnetometer all in one. This was selected to simplify our design and make our design more compact. Together these sensors will determine when apogee is reached for stage separation, if our rocket is stable on descent or when the grid fins will need to be rotated, and when to open the valve for the cold gas thruster on descent.

The last component of our avionics bay is the power system. One Lithium-Ion Polymer 3.7 V 500mAh battery will be used to power the whole avionics bay [111]. This type of battery was chosen due to it matching the operating voltage of the controller and having a high energy density. As a result, this type of battery minimizes mass, space, and cost required for our avionics bay. These considerations will be discussed further in the following section.

4.1.3 Avionics Bay

To contain the avionics within the body, they are housed within the avionics bay as seen in Figure 4-3. This is housed within the body tube and secured with bolts. The controller and sensors are mounted on a tray made of perf-board. This allows for easy assembly of the circuit and for the ability to remove the entire subsystem by sliding the tray out through a side panel access. While designing a custom Printed Circuit Board (PCB) was considered for this purpose, their primary advantage, manufacturability, does not justify the additional tasks involved in designing a PCB in a one-of-a-kind system such as this. Features such as component sockets can still be implemented on perf-boards.

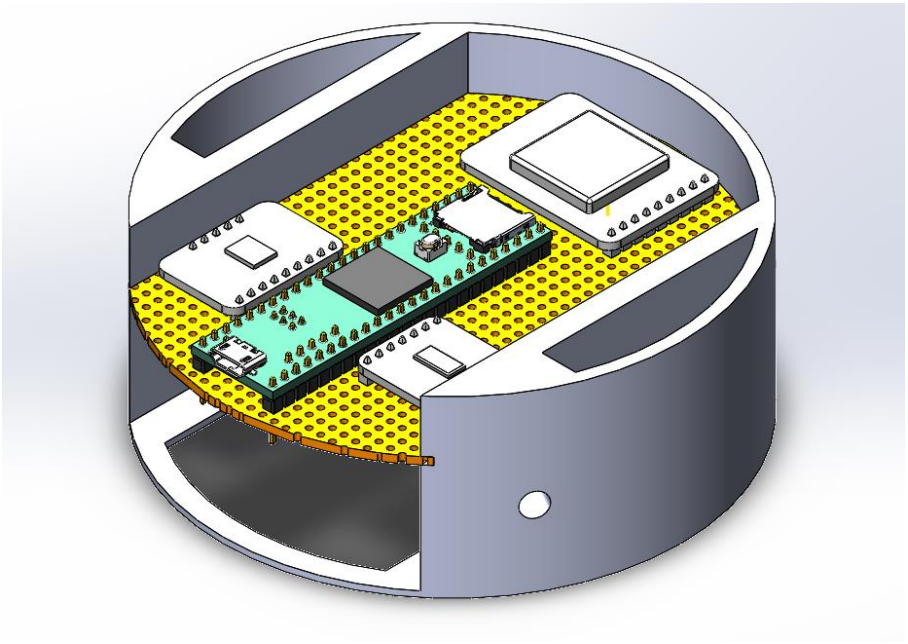


Figure 4-3 Isometric View of the Avionics Bay.

To accommodate wires with a non-restrictive length to allow for tray removal, a significant amount of space is left open. This space also contains a bracket to secure the avionics battery, shown in blue in Figure 4-4. There is also access on both the top and bottom of the bay for wires to be routed to connect to other parts of the vehicle such as the thrusters and grid fins.

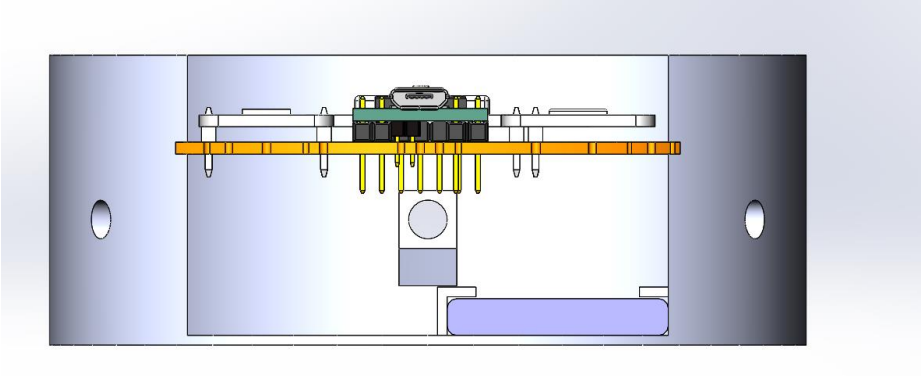


Figure 4-4 Front View of the Avionics Bay.

The structure of the bay is a single 3D printed component. Because it is not a load bearing component nor near the heat of the motor, the part will be printed using PLA. In order to bolt into the airframe, heat-set threaded inserts are set into holes in the structure.

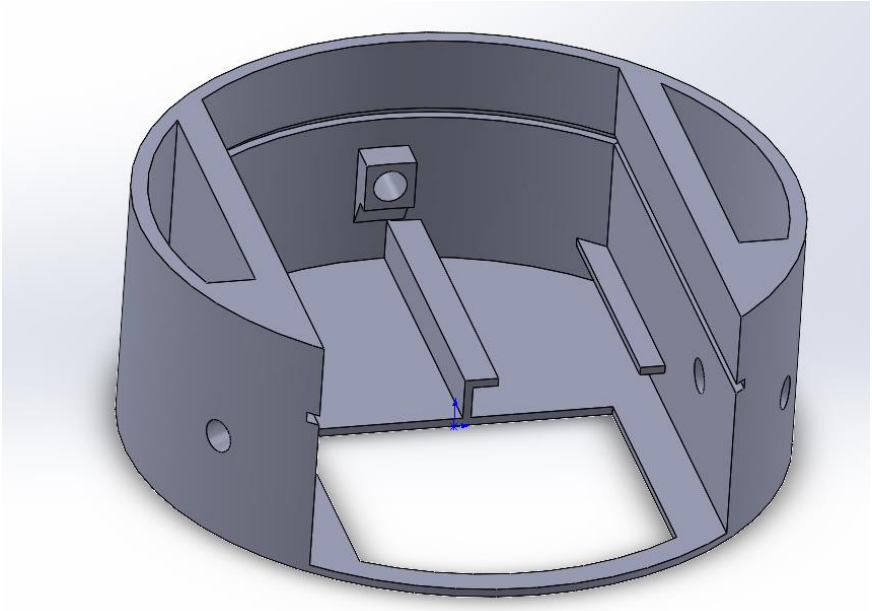


Figure 4-5 The Avionics Bay Structure.

4.2 Analysis

In this section, the result of the FDA subteam's analysis is presented along in addition to work which assesses its validity.

4.2.1 Analysis Task 1 (Vehicle Dynamics and Performance Model)

4.2.1.1 Vehicle Model

In order to model the dynamics of the vehicle, the problem was split into two coupled problems. These are a center of mass (COM) model, which uses Newton's Second Law to model the motion of the center of mass, and an Attitude Model, which uses Euler's equations to model the rotational motion of the rigid body. To compute the attitude of the vehicle, the simulator uses quaternions which define an axis and an angle about it to rotate. While traditional Euler angles are convenient and human readable, quaternions are well suited to computations as they do not suffer from mathematical singularities caused by Euler angle axis alignment. Splitting up the model allows both of them to be tested separately before being combined.

There are two reference frames used in the simulation, the inertial frame $\mathcal{I}$ and the body fixed frame $\mathcal{B}$. The x and y axis of the $\mathcal{I}$ frame form the ground plane while the positive z axis points upwards. The $\mathcal{B}$ frame is fixed to the center of mass of the vehicle with the positive x axis pointing along the central axis of the vehicle towards the nose cone as can be seen in Figure 4-6. Because the maximum apogee of model rockets flying on hobby scale motors is very small compared with the radius of the Earth, gravity is assumed to be constant and the effects of the Earth's rotation are assumed to be negligible. To simplify calculations, the simulator uses SI units.

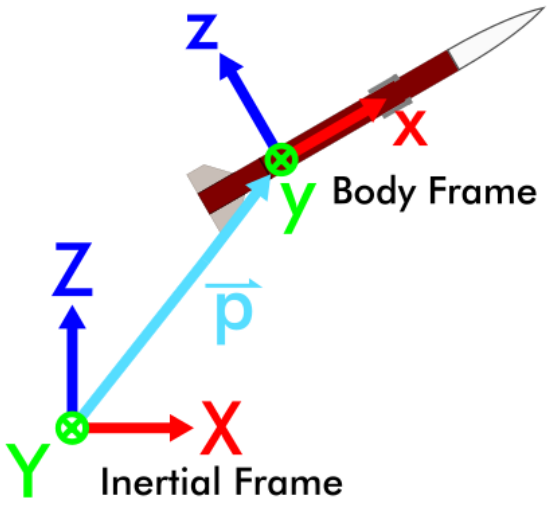


Figure 4-6 The I and B coordinate frames.

With both Center of Mass and Attitude models, the full state of the vehicle is defined by a state vector $\boldsymbol{x}$ as follows.

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{P} \\ \widehat{\boldsymbol{Q}} \\ \boldsymbol{V} \\ \boldsymbol{\omega} \\ m_p \end{bmatrix}$$

$$\boldsymbol{P}: Center\ of\ Mass\ Position$$
$$\widehat{\boldsymbol{Q}}: Attitude$$
$$\boldsymbol{V}: Center\ of\ Mass\ Velocity$$
$$\boldsymbol{\omega}: Angular\ velocity$$
$$m_p: Propellant\ Mass$$

where $\boldsymbol{P}$, $\boldsymbol{V}$, and $\boldsymbol{\omega}$ are all three element vectors. These three vectors are all represented in the $\mathcal{I}$ frame. The variable $\widehat{\boldsymbol{Q}}$ is a quaternion so it has four elements and describes the rotation of the positive x axis of the $\mathcal{B}$ frame relative to the positive x axis of the $\mathcal{I}$ frame. In this report, the first element of $\widehat{\boldsymbol{Q}}$ is the quaternion's scalar component while the latter three elements form its vector component and are separated by a dotted line as seen in Eq. 43. Finally, m_p is a positive scalar value representing propellant mass.

$$\widehat{\boldsymbol{Q}} = \begin{bmatrix} Q_1 \\ \cdots \\ Q_2 \\ Q_3 \\ Q_4 \end{bmatrix} \qquad 43$$

To simulate a typical launch, most of the initial state would be set to 0 except for $\widehat{\boldsymbol{Q}}$ and m_p. The initial attitude should be close to vertical and the propellant mass is set to the initial propellant mass of the vehicle. To compute the state over time, the derivative of the state vector is calculated. This allows the state to be solved over time using a numerical differential equation solver. To solve for the entire flight, the algorithm will run until landing is detected. This is determined when the z component of $\boldsymbol{P}$ becomes less than 0.

To characterize the vehicle, the following vehicle parameters must be defined. All of these are scalar quantities. They are constant with the exception of $\dot{m}$ which is a function of time.

$$I_{sp}: Motor\ specific\ Impulse$$
$$\dot{m}(t): Propellant\ mass\ flow$$
$$m_{pi}: Initial\ propellant\ mass$$
$$m_s: Structural\ mass\ (All\ non-mass\ propellant\ mass)$$
$$l: Length$$
$$d: diameter$$

Additionally, there are three environmental parameters that are defined. None are time varying in this simulation.

$$g: Gravitational\ acceleration$$
$$\rho: Atmospheric\ density$$
$$\boldsymbol{V_w}: Windspeed\ Vector$$

4.2.1.2 Code Structure

To better manage code structure, the simulator uses object-oriented programing in MATLAB. This allows functions and parameters to be grouped logically and for increased modularity with object polymorphism. This also makes it easier to test individual components in development. The simulation itself will be controlled by a main *Simulator* object that contains all other objects along with the main function. The other primary objects will include *Rocket* which contains vehicle parameters and control functions, *Environment* which contains environmental parameters and functions for generating wind, and *Aerodynamics* which interfaces with the CFD lookup tables to calculate aerodynamic forces and moments. The creation of these lookup tables is described in Section 4.2.2.

With all the objects defined, running the simulator is done relatively simply through a case script. All objects such as the *Environment*, *Aerodynamics*, and *Rocket* are initiated with the desired parameters. These objects are then passed to the *Simulator* object when it is initialized. This initialization function also takes various other arguments to customize the simulation. These

include enabling or disabling either the COM or Euler components of the simulation, defining wind gusts or forces. Finally, the Simulator is run by calling it's Execute function with the initial state. Specifically, MATLAB's $ode45$ function is used to numerically solve for the vehicle's state over time.

4.2.1.3 COM Simulator

In order to simulate the translation of the COM of the vehicle, the derivatives of $\boldsymbol{P}$ and $\boldsymbol{V}$ must be found. This is trivial for $\boldsymbol{P}$ as its derivative is $\boldsymbol{V}$. To find the solution for $\boldsymbol{V}$, we can solve the equation of Newton's Second Law. To account for wind, $\boldsymbol{V_w}$ is added to $\boldsymbol{V}$ before these computations.

$$\boldsymbol{dP} = \boldsymbol{V} \tag{44}$$

$$\boldsymbol{F_{net}} = m\boldsymbol{a} = m\,\boldsymbol{dV} \tag{45}$$

$$\boldsymbol{dV} = \frac{\boldsymbol{F_{net}}}{m} \tag{46}$$

To find m, the mass of the vehicle, at each time step, the structural mass of the vehicle along with the current propellant mass is found.

$$m = m_s + m_p \tag{47}$$

To find $\boldsymbol{F_{net}}$, we compute each individual force acting on the vehicle and sum them together. The three primary forces computed are the force of gravity $\boldsymbol{F_G}$, the thrust $\boldsymbol{F_T}$, and the total aerodynamic force $\boldsymbol{F_A}$. To ensure that $\boldsymbol{V}$ is always calculated in the $\mathcal{I}$ frame, each force is either computed in the $\mathcal{I}$ frame or rotated into it before the summation.

$$\boldsymbol{F_{net}} = \boldsymbol{F_G} + \boldsymbol{F_T} + \boldsymbol{F_A} \tag{48}$$

$$\boldsymbol{F_G} = \begin{bmatrix} 0 \\ 0 \\ -gm \end{bmatrix} \tag{49}$$

$$\boldsymbol{F_T} = \widehat{\boldsymbol{Q}} \otimes \begin{bmatrix} 0 \\ \cdots \\ \dot{m} I_{sp} g \\ 0 \\ 0 \end{bmatrix} \otimes \widehat{\boldsymbol{Q}}^{-1} \quad (50)$$

$$\widehat{\boldsymbol{v}}' = \widehat{\boldsymbol{q}} \otimes \widehat{\boldsymbol{v}} \otimes \widehat{\boldsymbol{q}}^{-1} \quad (51)$$

As seen in Eq. 49, the $\otimes$ symbol indicates quaternion multiplication, also known as quaternion composition. The quaternion in the middle of the equation is a pure quaternion, that is, its scalar component is 0 ($Q_1 = 0$), whose vector component is the thrust in the $\mathcal{B}$ frame. This allows for the thrust to be rotated into the $\mathcal{I}$ frame using the quaternion implementation of Rodrigues' rotation formula [112]. This yields another pure quaternion whose vector component is the thrust in the $\mathcal{I}$ frame. The general form of this can be seen in Eq. 51. Note that while sometimes in literature the conjugate of $\widehat{\boldsymbol{q}}$ is used ($\widehat{\boldsymbol{q}}^*$), this is mathematically equivalent to its inverse ($\widehat{\boldsymbol{q}}^{-1}$).

When running independent from the Euler Simulator, the COM simulator makes the zero angle of attack assumption. Essentially, this means that the vehicle's attitude always matches the orientation of $\boldsymbol{V}$. To compute a $\widehat{\boldsymbol{Q}}$ that matches the orientation of $\boldsymbol{V}$, the yaw and pitch angles of $\boldsymbol{V}$ are found. These are then used to create a rotation matrix that rotates about the Z axis then the Y axis. This is used to compute a direction cosine matrix (DCM) that can then be converted to a quaternion using Algorithm 11.2 of Orbital Mechanics for Engineering Students [112].

$$\theta = \cos^{-1}\left(\frac{V_x}{\sqrt{V_y^2 + V_x^2}}\right)$$
$$\phi = \tan^{-1}\left(\frac{V_z}{\sqrt{V_y^2 + V_x^2}}\right)$$
$$R_{ZY} = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos(\phi) & 0 & -\sin(\phi) \\ 0 & 1 & 0 \\ \sin(\phi) & 0 & \cos(\phi) \end{bmatrix} \quad (52)$$
$$Q = \begin{bmatrix} \left(R_{ZY}\begin{bmatrix}1\\0\\0\end{bmatrix}\right)^T \\ \left(R_{ZY}\begin{bmatrix}0\\1\\0\end{bmatrix}\right)^T \\ \left(R_{ZY}\begin{bmatrix}0\\0\\1\end{bmatrix}\right)^T \end{bmatrix}$$

To compute the aerodynamic force, the angle of attack is first found using the unit vector of $\boldsymbol{V}$ and a unit vector of the attitude. This is then used in combination with a cubic function fit to the coefficient of drag (C_D) data found in CFD in Section 4.2.2. The cross-sectional area needed for the equation is estimated as a rectangle with height D and length $D + L\cos\alpha$. The force is applied in the direction of the velocity.

$$\widehat{\boldsymbol{u}}_{\boldsymbol{Q}} = \widehat{\boldsymbol{Q}} \otimes \begin{bmatrix} 0 \\ \cdots \\ 1 \\ 0 \\ 0 \end{bmatrix} \otimes \widehat{\boldsymbol{Q}}^{-1}$$
$$\widehat{\boldsymbol{u}}_{\boldsymbol{V}} = \frac{\boldsymbol{V}}{\|\boldsymbol{V}\|} \quad (53)$$
$$\alpha = \cos^{-1}\left(\widehat{\boldsymbol{u}}_{\boldsymbol{Q}} \cdot \widehat{\boldsymbol{u}}_{\boldsymbol{V}}\right)$$

$$A = D^2 + DL\cos\alpha$$
$$\boldsymbol{F_A} = -\frac{C_D A \rho \|\boldsymbol{V}\|^{\boldsymbol{2}}}{2} * \widehat{\boldsymbol{u}}_{\boldsymbol{V}} \quad (54)$$

4.2.1.4 Euler Simulator

To simulate the rotation of the vehicle, the derivatives of $\widehat{\boldsymbol{Q}}$ and $\boldsymbol{\omega}$ must be found. While not as simple as computing $\boldsymbol{dP}$, the equation for $\boldsymbol{d\widehat{Q}}$ is a function of only $\boldsymbol{\omega}$, which itself is found using Euler's equations.

$$\boldsymbol{d\widehat{Q}} = \frac{\widehat{\boldsymbol{Q}} \otimes \begin{bmatrix} 0 \\ \cdots \\ \boldsymbol{\omega} \end{bmatrix}}{2} \tag{55}$$

$$\boldsymbol{d\omega} = \begin{bmatrix} \dfrac{M_x - (I_z - I_y)\omega_y\omega_z}{I_x} \\ \dfrac{M_y - (I_x - I_z)\omega_x\omega_z}{I_y} \\ \dfrac{M_z - (I_y - I_x)\omega_y\omega_x}{I_z} \end{bmatrix} \tag{56}$$

To calculate $\boldsymbol{M}$, the net moment on the vehicle, we must look at the three forces acting on the rocket. Because gravity always acts through the center of mass and is assumed to be uniform, $\boldsymbol{F_G}$ never exerts any moments. Thrust is assumed to only act directly along the vehicle's central axis so $\boldsymbol{F_T}$ also produces no moment. The only moments on the vehicle are due to aerodynamic forces. To find these, the CFD data is used to calculate coefficient of moment in the same way coefficient of drag is calculated in Section 4.2.1.3. This C_M is then dimensionalized to become the real moment τ.

$$\tau = C_M A L * \frac{\rho \|\boldsymbol{V}\|^2}{2} \tag{57}$$

This moment is then split into components based on the difference in pitch and yaw between $\boldsymbol{V}$ and $\widehat{\boldsymbol{Q}}$ to find the total moment $\boldsymbol{M}$.

$$\widehat{\boldsymbol{u}}_{\boldsymbol{V}} = \frac{\boldsymbol{V}}{\|\boldsymbol{V}\|}$$

$$\widehat{\boldsymbol{u}}_{\boldsymbol{Q}} = \frac{\widehat{\boldsymbol{Q}} \otimes \begin{bmatrix} 0 \\ \cdots \\ 1 \\ 0 \\ 0 \end{bmatrix} \otimes \widehat{\boldsymbol{Q}}^{-1}}{\left\| \widehat{\boldsymbol{Q}} \otimes \begin{bmatrix} 0 \\ \cdots \\ 1 \\ 0 \\ 0 \end{bmatrix} \otimes \widehat{\boldsymbol{Q}}^{-1} \right\|} \tag{58}$$

$$\theta = sin^{-1}\left(\frac{\widehat{\boldsymbol{u}}_{\boldsymbol{Q}_z}}{\|\widehat{\boldsymbol{u}}_{\boldsymbol{Q}}\|}\right) - sin^{-1}\left(\frac{\widehat{\boldsymbol{u}}_{\boldsymbol{V}_z}}{\|\widehat{\boldsymbol{u}}_{\boldsymbol{V}}\|}\right)$$

$$\phi = tan^{-1}\left(\frac{\widehat{\boldsymbol{u}}_{\boldsymbol{V}_y}}{\widehat{\boldsymbol{u}}_{\boldsymbol{V}_x}}\right) - tan^{-1}\left(\frac{\widehat{\boldsymbol{u}}_{\boldsymbol{Q}_y}}{\widehat{\boldsymbol{u}}_{\boldsymbol{Q}_x}}\right)$$

$$\boldsymbol{M} = \widehat{\boldsymbol{Q}}^{-1} \otimes \begin{bmatrix} 0 \\ \cdots \\ 0 \\ \tau\theta \\ \tau\phi \end{bmatrix} \otimes \widehat{\boldsymbol{Q}}$$

Note that $\widehat{\boldsymbol{Q}}$, and its inverse are flipped in Eq. 58. This is because $\widehat{\boldsymbol{Q}}$ is a rotation from $\mathcal{I}$ to $\mathcal{B}$ so flipping the two results in the opposite rotation to transfer the body frame moments to the inertial frame. There is also the additional caveat that the radians unit from the angles θ and ϕ does not completely cancel out of the equation. This will likely cause moments to be greater than those simulated in CFD.

4.2.1.5 Integrated Simulator

With both enabled, the COM simulator and Euler simulator work together to solve for the full translational and rotational motion of the vehicle as a rigid body. In each time step, the COM simulator is executed first. A flow chart showing the equations used in the full integrated simulator can be seen in Figure 4-7. This provides the most comprehensive simulation of the vehicle's flight.

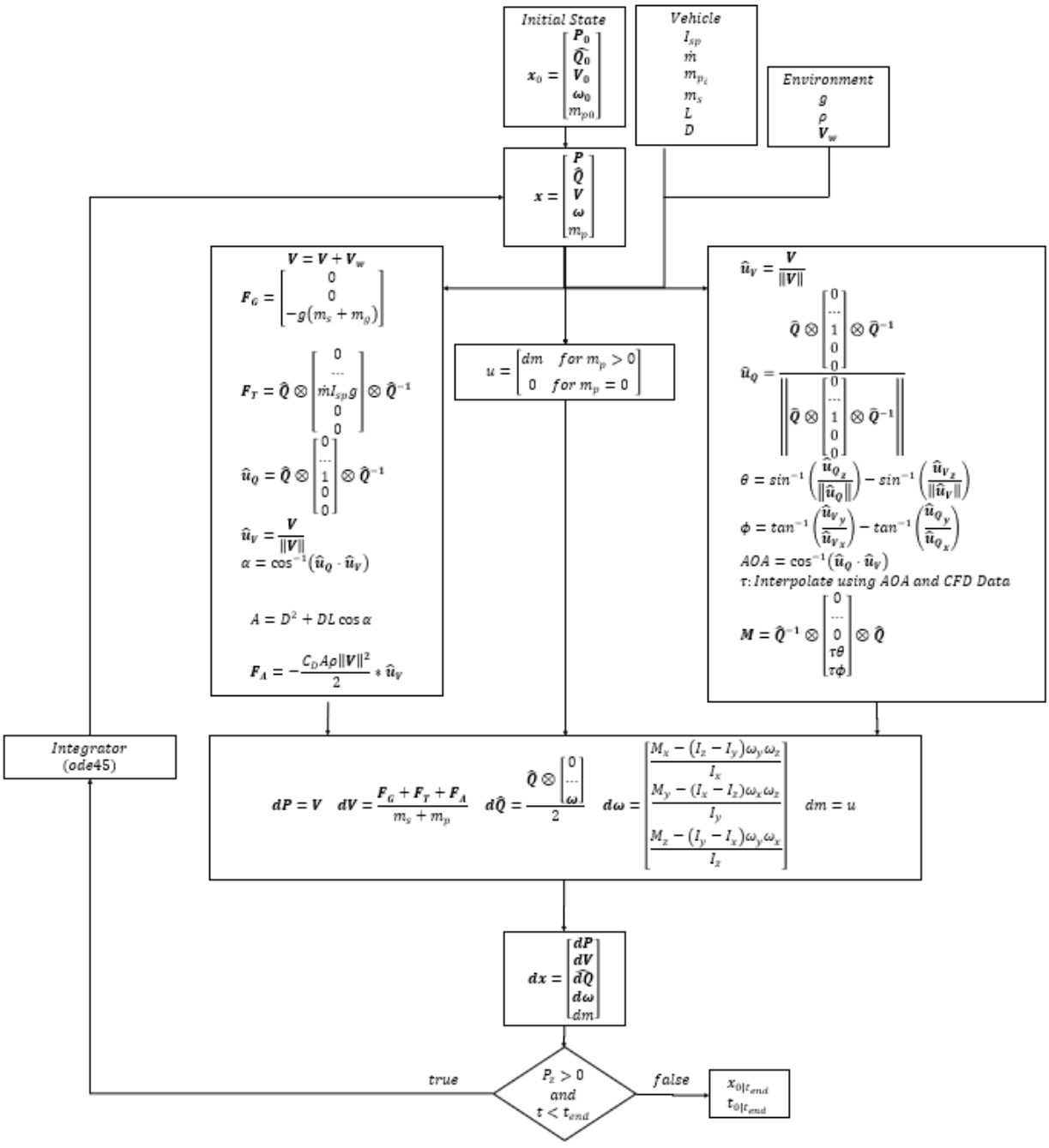


Figure 4-7 Integrated Simulation Flow Chart

In one iteration of the solver, the current state is first passed to the control solver which computes the current rate of propellant flow. Next, it goes to the COM component, which solves for all forces on the vehicle. Lastly, the Euler segment computes all moments on the vehicle. These forces and moments, all of which are in the $\mathcal{I}$ frame, are then used to compute the derivative of the

state vector, $\boldsymbol{dx}$. The *ode45* function then checks to see if the vehicle has landed, that is if $p_z \leq 0$, or if the simulator has reached the maximum end time in which case, the simulator terminates. Otherwise, a new iteration is begun with the new state. The MATLAB code used to implement this dynamical simulator can be seen in Appendix F: Dynamical Simulator Code.

4.2.1.6 Simulation Cases

To examine the results of the simulator, five different cases are presented. Each one demonstrates a specific aspect of the simulator. The code used in each case can be seen in Appendix F: Dynamical Simulator Code. The vehicle and environmental parameters are given in tables Table 4-1 and Table 4-2 and are used in every case unless otherwise specified.

Vehicle Parameters	
m_s	6.969 kg
m_{p0}	0.708 kg
I_{sp}	184 s
L	1.757 m
D	0.102 m
t_b	3.4 s

Table 4-1 Vehicle Parameters

Environmental Parameters	
g	9.80665 m/s^2
ρ	1.225 kg/m^3

Table 4-2 Environmental Parameters

4.2.1.6.1 Case 1: Movement of a Point Mass

In Case 1, a simple point mass is simulated using the COM simulator. There is no propellent, and thus no thrust, nor is gravity acting on the vehicle. Only two predefined forces act. The first one, which acts from $t = 1s$ to $t = 2s$, is defined by the vector [2 2 3] while the second one, which acts between $t = 3s$ and $t = 4s$, is defined as $[0\ 0\ -3]$. The purpose of this case is to demonstrate the ability of the COM simulator to react appropriately to forces.

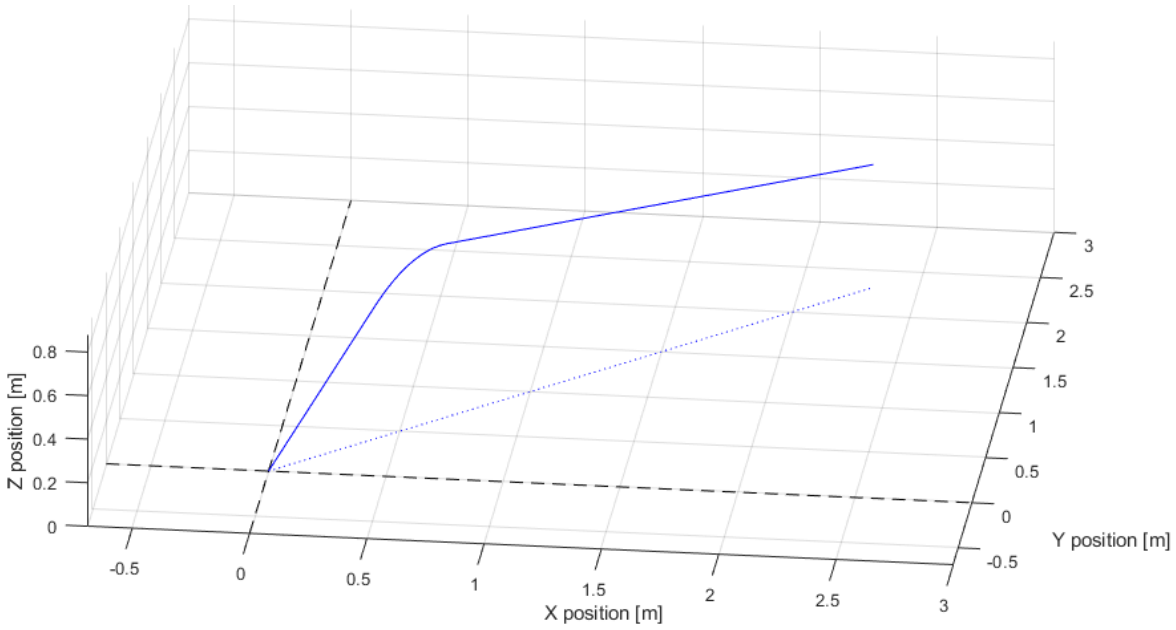


Figure 4-8 Case 1 Flight Path

A 3D view of the particle's flight path can be in Figure 4-8 as the solid blue line. Note that the dotted blue line shows a line drawn on the X-Y plane between the starting and ending points of the simulation which was done to make the graph more readable. The components of $\boldsymbol{P}$ and $\boldsymbol{V}$ were also graphed as well as the magnitude of $\boldsymbol{V}$ in Figure 4-9 and Figure 4-10.

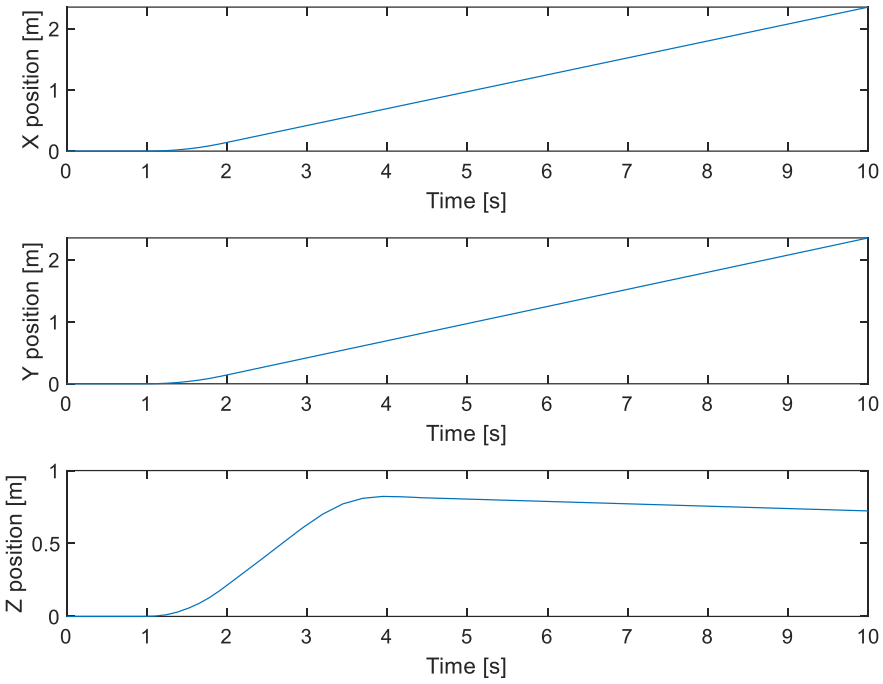


Figure 4-9 Case 1 Position over Time

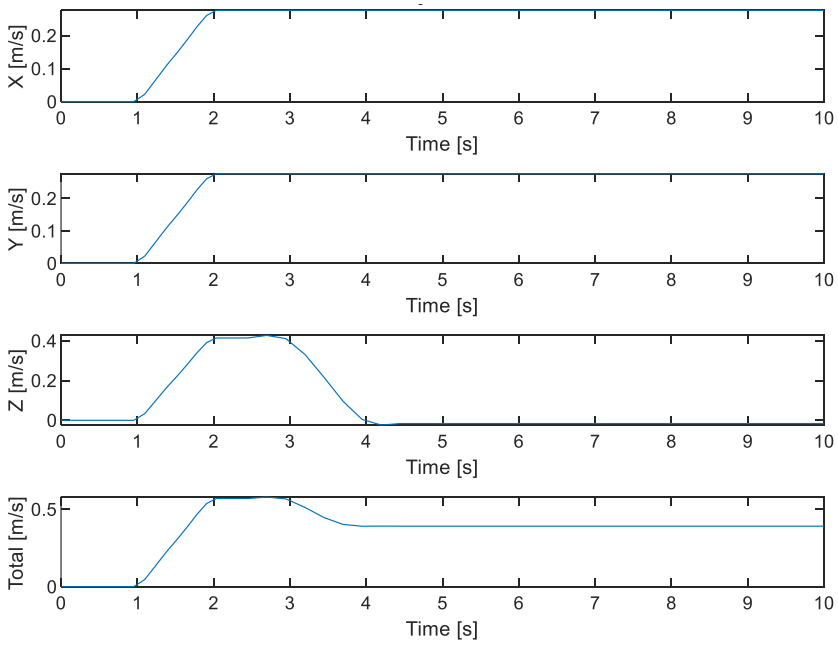


Figure 4-10 Case 1 Velocity over Time

We can then validate this using the law of conservation of momentum. The initial state was stationary and thus had no momentum, so all momentum must come from the two impulses exerted on the particle. We first calculate the expected momentum on each axis.

$$\boldsymbol{p_e} = \begin{bmatrix} \int_{t_{01}}^{t_{f1}} F_x dt + \int_{t_{02}}^{t_{f2}} F_x dt \\ \int_{t_{01}}^{t_{f1}} F_y dt + \int_{t_{02}}^{t_{f2}} F_y dt \\ \int_{t_{01}}^{t_{f1}} F_z dt + \int_{t_{02}}^{t_{f2}} F_z dt \end{bmatrix} \tag{59}$$

$$\boldsymbol{p_e} = \begin{bmatrix} 2 \\ 2 \\ 0 \end{bmatrix} \frac{kg}{m/s}$$

Next, we can compute the final momentum of the particle. Because it has the properties of the rocket but no propellant, we use the structural mass of the vehicle.

$$\boldsymbol{p_a} = m_s \boldsymbol{V_f}$$

$$\boldsymbol{p_a} = 6.6961 * \begin{bmatrix} .2762 \\ .2762 \\ -.0162 \end{bmatrix} \tag{60}$$

$$\boldsymbol{p_a} = \begin{bmatrix} 1.849 \\ 1.849 \\ -0.1087 \end{bmatrix} \frac{kg}{m/s}$$

Finally, we compute the difference between the expected and actual momentums.

$$\boldsymbol{\Delta p} = \begin{bmatrix} 2 \\ 2 \\ 0 \end{bmatrix} - \begin{bmatrix} 1.849 \\ 1.849 \\ -0.1087 \end{bmatrix}$$

$$\boldsymbol{\Delta p} = \begin{bmatrix} 0.151 \\ 0.151 \\ 0.1087 \end{bmatrix}$$

As expected, the final results for $\boldsymbol{\Delta p}$ are very close to zero. The most probable reason that $\boldsymbol{\Delta p}$ is not precisely zero is $ode45$'s dynamic time step. When MATLAB is integrating using $ode45$, the time step is not constant nor is it aligned with the $1s$ intervals the forces are applied over. As such, a small error is expected. From this case, we can conclude that the COM simulator reacts appropriately to forces.

4.2.1.6.2 Case 2: Zero Angle of Attack Flight

Case 2 is a full flight simulation done using the COM simulator. This means that the vehicle is simulated as a point mass on which gravity, thrust, and drag forces are applied. The initial attitude is set to a near vertical angle, the initial propellant mass is that of a K360 motor, and all other state variables are initially zero.

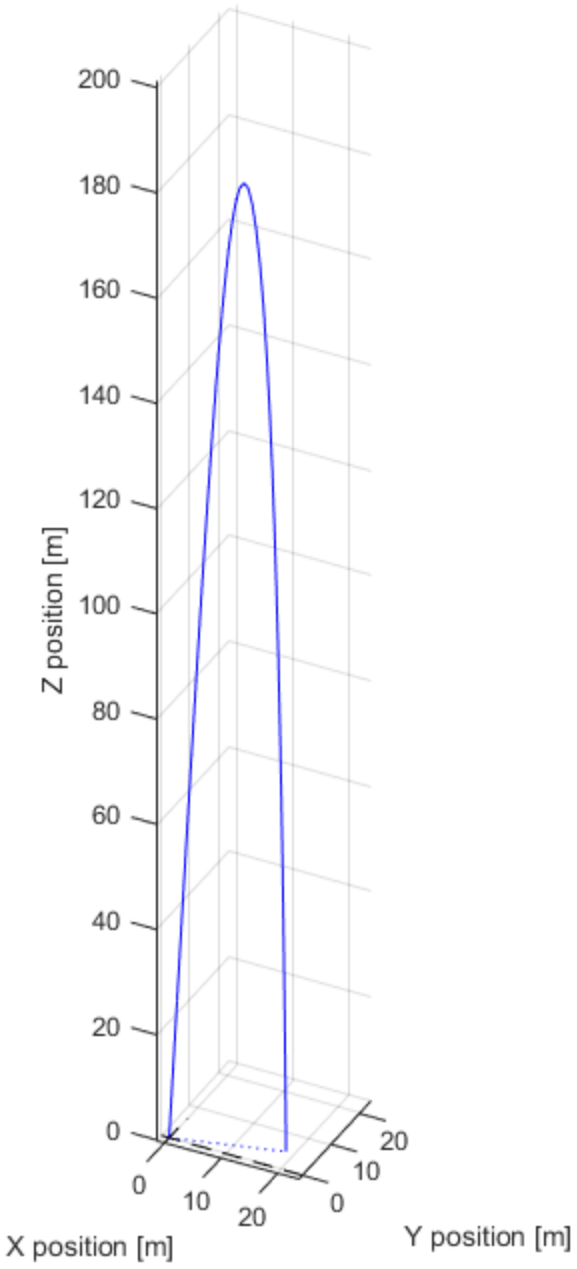


Figure 4-11 Case 2 Flight Path

As seen in Figure 4-11, the flight path of the vehicle is parabolic. This is expected as gravity is the only force that acts off the vehicle's central axis. ***P*** and ***V*** are also presented in Figure 4-12 and Figure 4-13. These also follow the anticipated behavior as ***V*** initially increases rapidly under the driving force of the motor before quickly falling off due to drag.

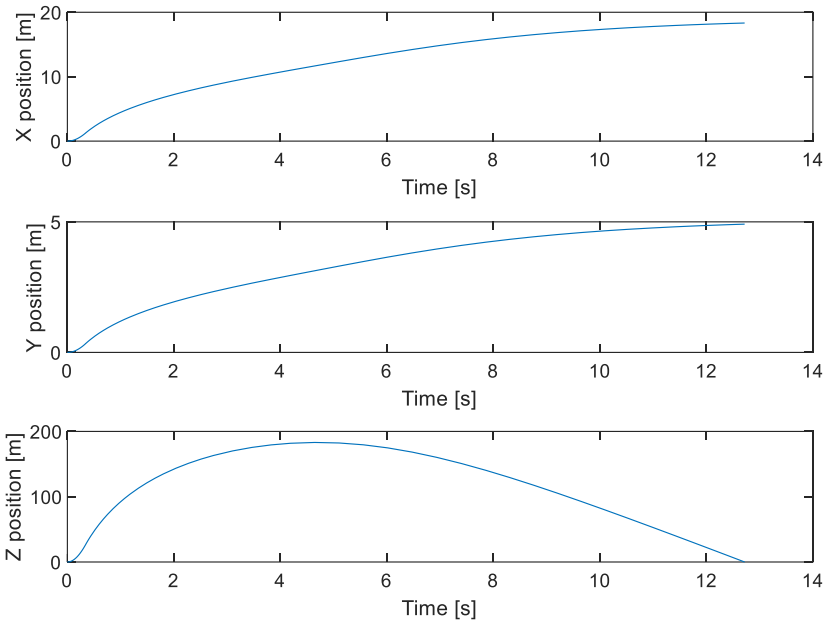


Figure 4-12 Case 2 Position vs. Time

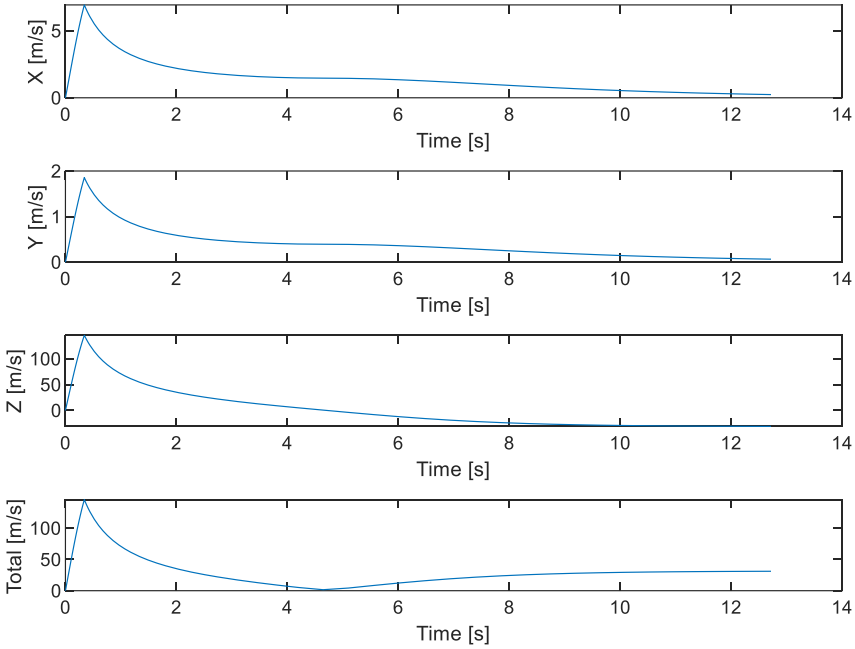


Figure 4-13 Case 2 Velocity vs. Time

4.2.1.6.3 Case 3: Rotation of a Rigid Body

For Case 3, we simulate the vehicle as a rigid body using only the Euler simulator. Just as Case 1 was meant to test the COM simulator's response to forces, this case tests the Euler simulator's response to a moment. No propellant is used, gravity is not simulated, and there is no aerodynamic forces or moments. Instead, a 5 Nm rolling moment is applied from $t = 5s$ to $t = 10s$.

To display the results of the Euler solver, the attitude is converted from the quaternion output to more human readable Euler angles. These are represented as pitch, yaw, and roll as applied in that order. Finally, these angles are wrapped around a range of $-180°$ to $180°$.

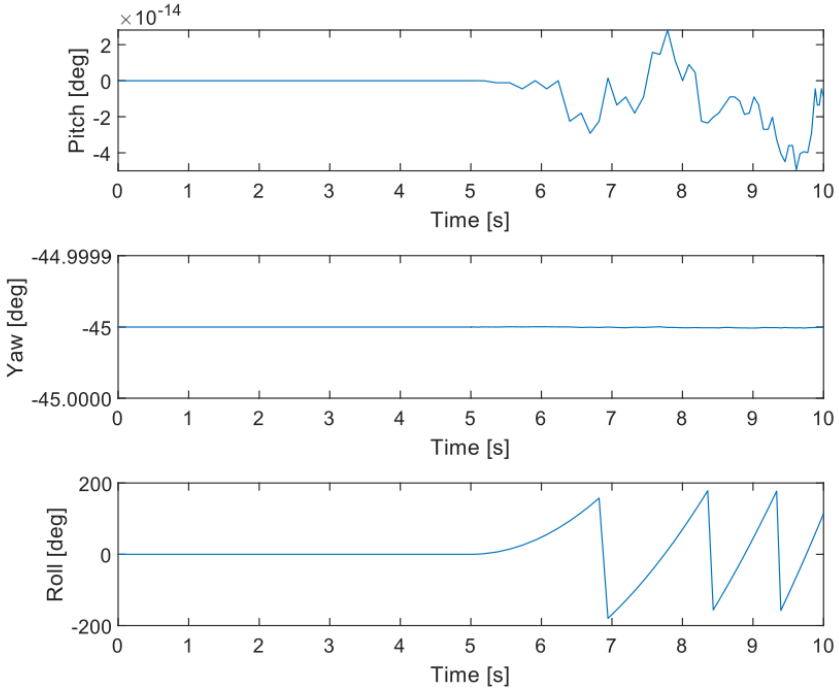


Figure 4-14 Case 3 Attitude vs. Time

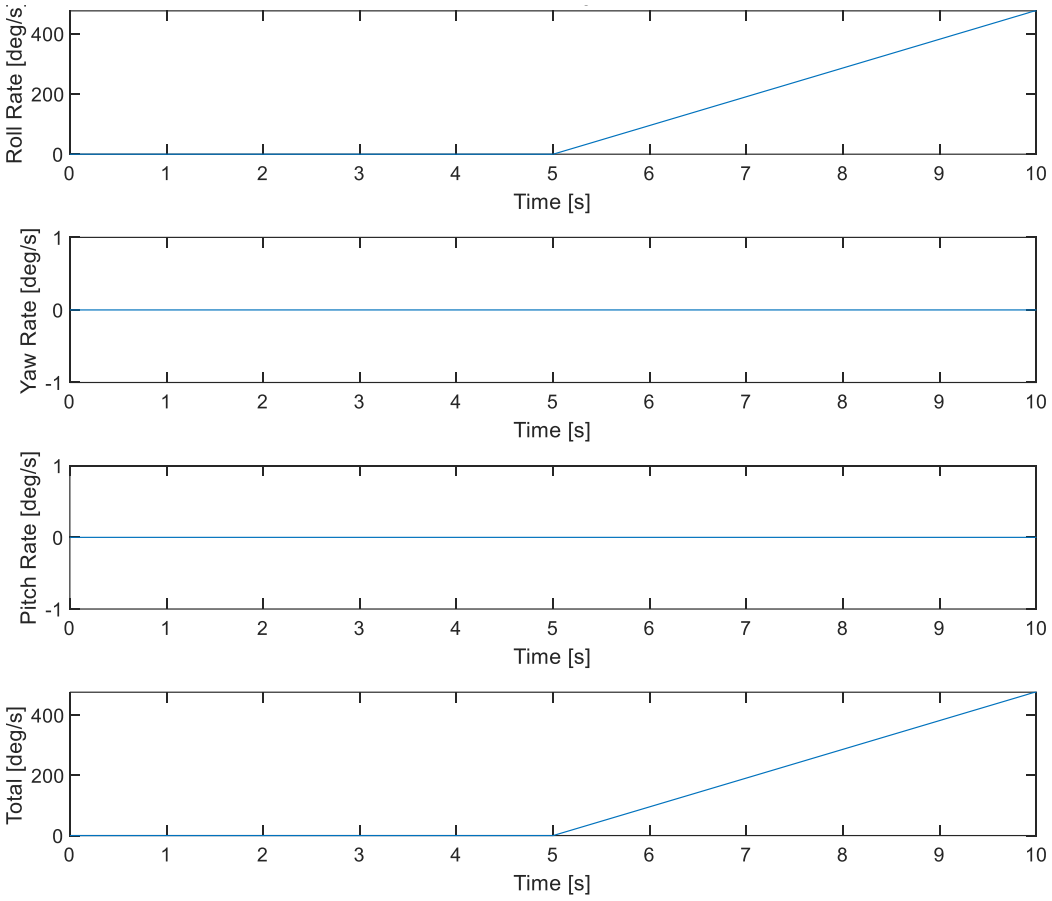


Figure 4-15 Rotational Velocity vs. Time

Note that while the pitch angle appears to vary wildly in Figure 4-14, the scale of the graph is on the order of 10^{-14} which is within the expected error of the numerical solver. As for $\boldsymbol{\omega}$, its roll (ω_x) component increases linearly as expected from the rolling moment. Secondly, the roll component of the attitude increases exponentially. This behavior indicates that the Euler solver is acting in a realistic manner.

4.2.1.6.4 Case 4: Rigid Body Precession

Case 4 is intended to test whether or not the Euler simulation is capable of precession. Precession is the process by which the rotational axis of a rotating body itself rotates about a second axis. While precession without any moments exerted is possible, this example looks at the case of moment induced precession. In this case, the vehicle is pointed 20° from the z axis and given an

initial roll rate of 60 rad/s. The base of the rocket is treated as being fixed to the ground, so there is a moment that is caused by the force of gravity acting through the center of mass.

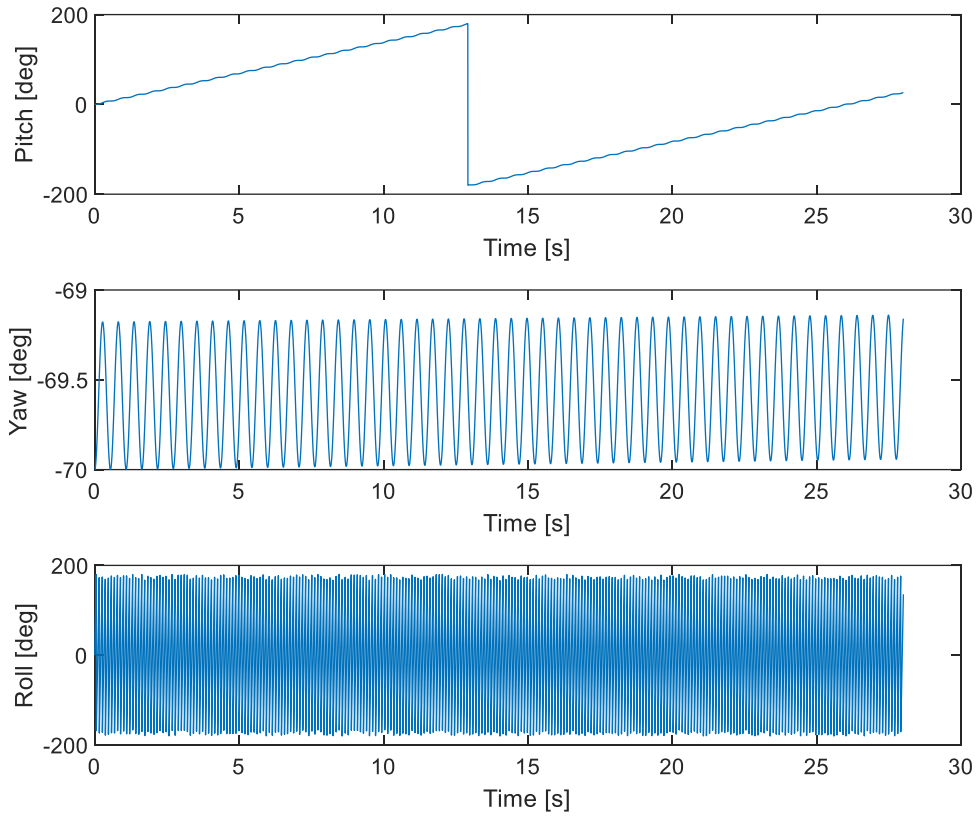


Figure 4-16 Case 4 Attitude vs. Time.

As can be seen in Figure 4-16, the vehicle consistently yaws about the z axis as expected. Also, of note is the oscillation about the y axis. This is caused by a similar phenomenon to precession known as nutation which the axis of precession changes. The observation of both precession and nutation indicates that the Euler solver is behaving correctly in response to moments under a non-zero $\boldsymbol{\omega}$.

4.2.1.6.5 Case 5: Full Integrated Flight

The final case is intended to demonstrate the full integrated simulator. In this simulation, the vehicle starts stationary on the launch pad oriented 5° off the z axis. Both the COM and Euler simulators are active and working together. Additionally, a constant, south-westerly wind of 8 m/s is present.

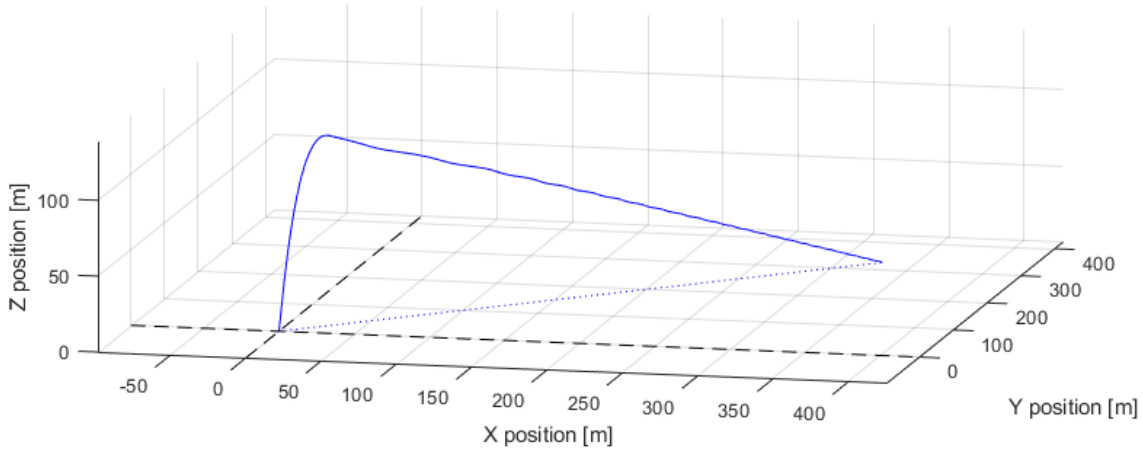


Figure 4-17 Case 5 Flight Path

As seen in Figure 4-17, the vehicle ascends quickly under the power of the motor. After apogee, the rocket begins descending at a slower terminal velocity. Because no parachutes are simulated, the vehicle is tumbling at this point which explains the significantly higher drag on descent than Case 2. All the while, the vehicle moves diagonally due to the wind. This flight path is consistent with the expectation for vehicles tumbling during descent.

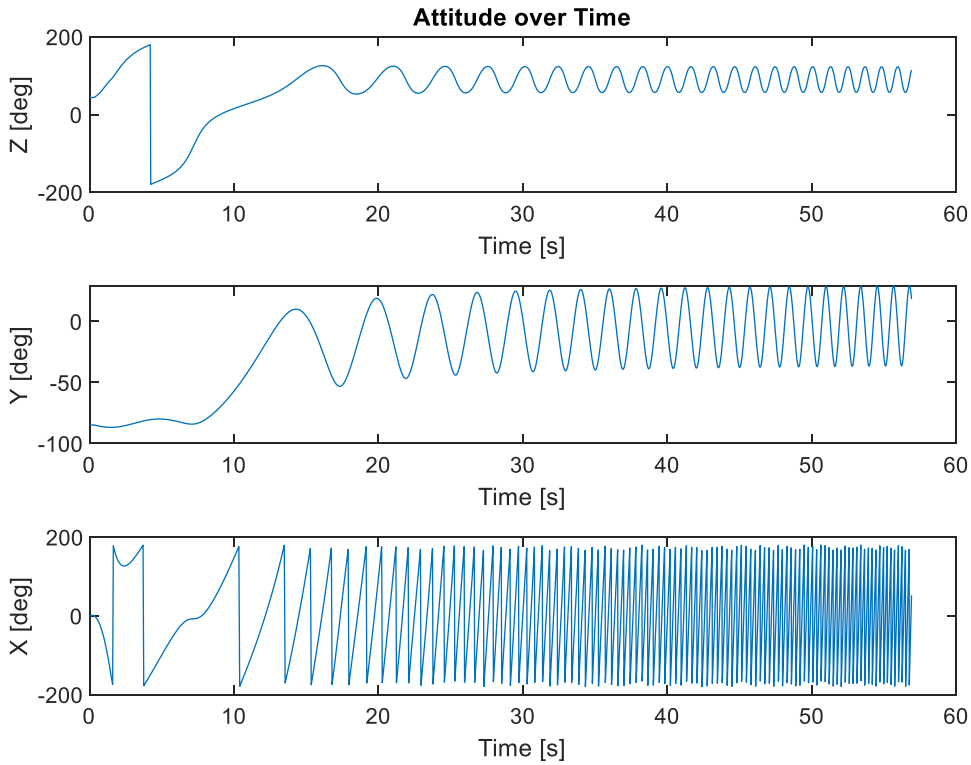


Figure 4-18 Case 5 Attitude vs. Time

In Figure 4-18, we can see the attitude as a function of time. The vehicle initially only pitches over slowly as the initially high velocity ensures the vehicle is very stable. As velocity reduces near apogee, stability decreases until it eventually begins to tumble. Graphs of the other state variables in case 5 can be found in Appendix D.

4.2.2 Analysis Task 2 (Vehicle Aerodynamic Loads Simulation)

The process for FDA Analysis Task 2 was similar to how ARS completed their aerodynamic load analysis, described in Section 2.2.4. This task was also completed by using Ansys Fluent. A general description of how CFD works is listed in Section 1.1.3.4. An explanation of the process chosen, boundary condition selected, and assumptions made to complete these models are presented in Section 2.2.2. This section details the specific process used to analyze the aerodynamic loads on the rocket in the ascent configurations.

The initial step was to determine what regime of flow the rocket would experience on ascent. This is important because the flow the rocket experiences will affect the aerodynamic loads. In order to characterize the flow regime, the Reynolds number must be calculated. If the number is low it will be a laminar flow. However, in general if the Reynolds number is above 10^4 it is considered a turbulent flow [43]. This number can be calculated using Eq. 38.

$$Re = \frac{\rho V L}{\mu} \tag{38}$$

In (54), ρ represents the density of the fluid, V is the velocity of the fluid, L is the characteristic length, and μ is the viscosity of the fluid [62]. To simplify our modeling in Ansys, the fluid was assumed to be air at sea level as shown in Table 2-2. This was chosen since the maximum height simulated through OpenRocket was found to be relatively low, ranging from 495 m to 1624.02 m, which would not change the density or viscosity of the air significantly compared to sea level. In addition, OpenRocket was used to find the range of velocities on ascent to be from 0 m/s to a maximum velocity of 133 m/s or 436.35 ft/s. The diameter of the rocket was chosen from the dimension entered in OpenRocket. The calculated Reynolds number was found to be above 10^4 and is therefore considered to be a turbulent flow. This now confirms that the same initial set up for the solver can be performed on the rocket on ascent as it was for the grid fins

The first part of modeling in Ansys was to create the geometry of the object being modeled. The geometry for the rocket on ascent is shown in Figure 4-19. To accomplish this a simplified version of the rocket was generated, where the grid fins were modeled as solid plates that fit flush against the body tube of the rocket and all parts were considered one continuous piece. This was created in SOLIDWORKS and was then imported into Ansys. Once this was completed a small cylinder was placed around the rocket geometry. A Boolean subtraction tool was then used to subtract the rocket geometry form the cylinder geometry. This was done because for a CFD model the focuses on how the fluid interacts with the geometry of the rocket not the structure of rocket itself. An additional larger cylinder was added to the model afterwards. The reason for two different cylinders is the smaller one acts as a body of influence and helps to create a more accurate mesh for this geometry later on. After the cylinders were made, the geometry was sliced into sections based on where the complex geometry, such as the nose cone, grid fins, and stability fins,

were located. This was done so a more detailed mesh could be created around complex geometry where mesh errors are likely to occur.

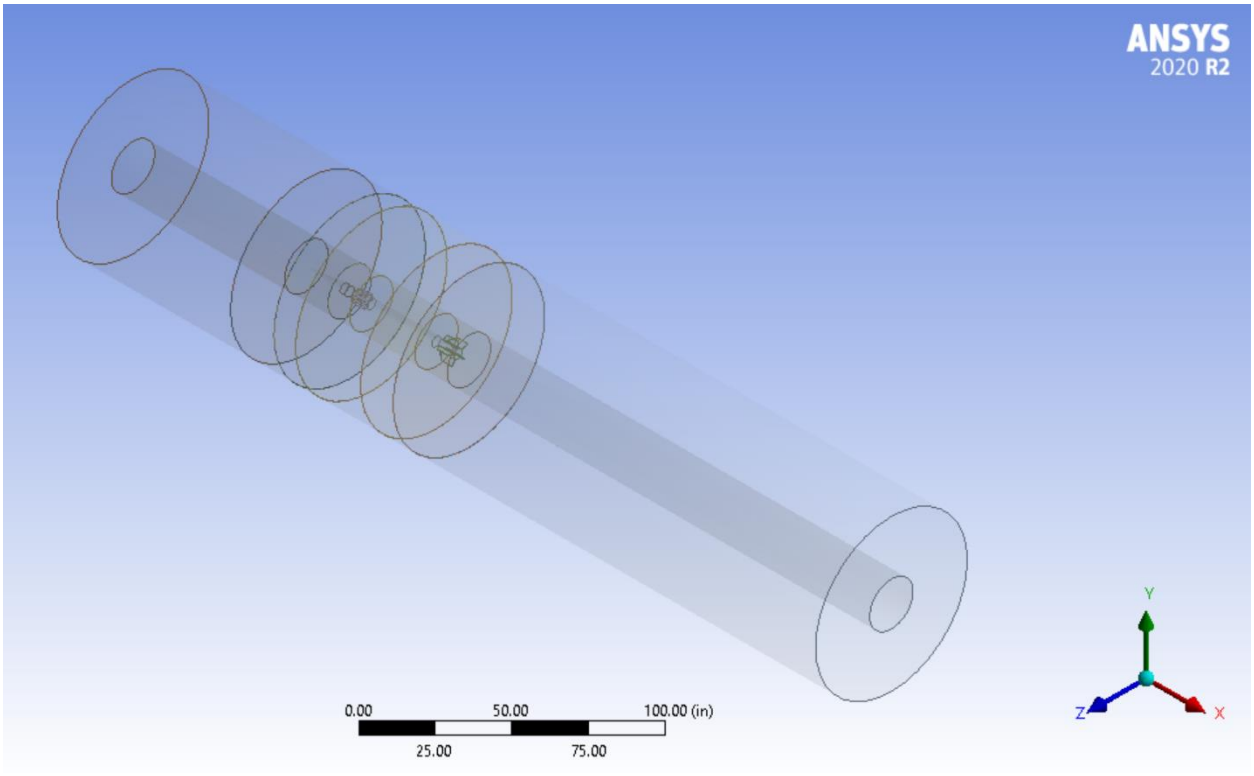


Figure 4-19 Rocket Geometry for Ansys Simulation

Creating a mesh for a geometry tends to be a complex and iterative part of modeling. Errors tend to occur at this point and sometime this process has to be reevaluated later to try and create a more accurate result. During this step issues appeared due to the complex geometry of the rocket and as a result the geometry was simplified. Example of issues FDA ran into for the mesh include (especially on slower processing computers) that the mesh generation tool would not run or not all of the fins would appear with the automatic mesh. An automatic mesh is where the software makes a crude mesh that is less detailed and accurate but in general quicker than a custom mesh. To resolve these problems, a different computer with higher processing power was used, and face sizing was added to the grid fin and stability fin sections of the body. Once this ran successfully, an inflation layer was added to the rocket to get a more detailed mesh and accurate results. In

addition, the mesh method was then selected to be the patch conforming method not an automatic mesh. After this was completed the set up for the simulation could begin.

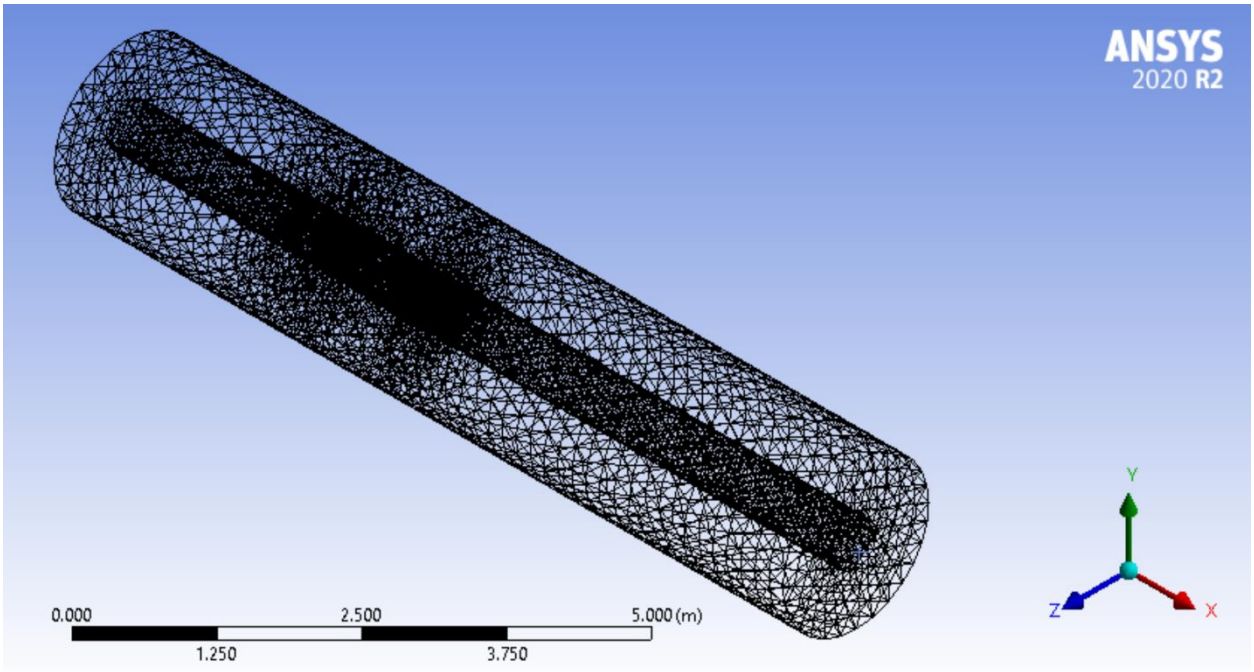


Figure 4-20 Rocket Mesh

In order to start the set up, some values must be known. The purpose of this analysis task was to determine the lift, drag, and moments produced on the rocket on ascent. These values will change depending on the free stream velocity and the angle of attack of the rocket, therefore multiple cases had to be run. From the Ansys Fluent data collected; the lift coefficient, drag coefficients, and the moments were calculated and utilized for FDA Analysis Task 1. Due to the Mach number being less than 0.4, the coefficients are assumed to be constant in the flow regime and therefore was used for the entire ascent of the simulation. This data was then used in a lookup table that was incorporated into FDA Analysis Task 1, our six degree of freedom simulation. For this data one velocity was used for multiple angles of attack ranging from 0 degrees to 15 degrees. This range was due to time constraints and because this tends to be the range of a realistic flight.

To run simulations of each case a new mesh and set up would have to be made each time. This is due to the body having to be rotated for each different angle of attack while the free stream velocity is held constant in one direction. This approach would be time consuming and challenging, however there is a way to simplify this. It was decided that the rocket body would be fixed at a zero angle of attack and the free stream velocity will have two components based on the values of the angle of attack. A visual representation of this approach is demonstrated in Figure 4-21. This approach saved time and gave accurate results. Once this selection is made the boundary condition can be chosen for the set up.

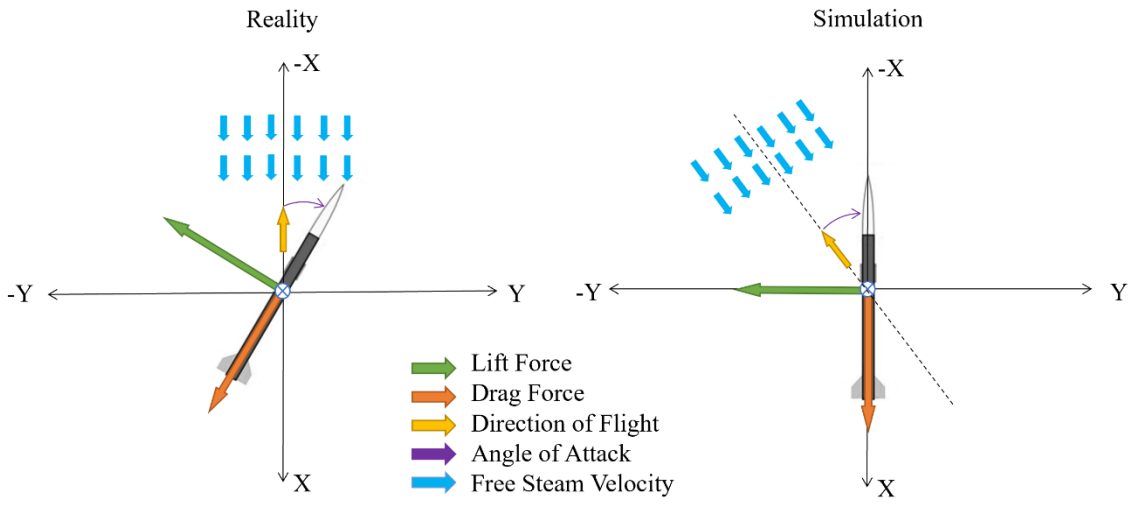


Figure 4-21 Orientation of Rocket and Free Stream Velocity

Values used for the boundary conditions of the rocket were similar to the grid fins. The main difference was the free stream velocity, and the location on the geometry selected for the conditions. The reason a different value was used for the velocity was because this model is for ascent and the grid fin model is for descent of the rocket, therefore they will have different free stream velocities. The velocity chosen for the ascent rocket was 85.34 m/s or 280 ft/s, this is slightly larger than the mean of the maximum velocity, 133 m/s or 436.35 ft/s, found in OpenRocket and the initial velocity of zero when the rocket is on the launch pad right at ignition. This value was selected because, unless a catastrophic failure happened, the rocket would likely

reach the specified value at some moment of its ascent. It is not guaranteed that the rocket would reach the simulated maximum velocity in OpenRocket. In reality, the rocket would probably not reach its maximum velocity due to unaccounted mass or the rocket not being as aerodynamic as it is in the OpenRocket simulation. The location of the boundary conditions on the rocket ascent model are shown in Figure 4-22 and

Table 4-3 lists the boundary conditions and the values used. Once these conditions are input into Ansys Fluent, selection can be made to solve for lift, drag, and moments. The values found will be incorporated into the lookup table for FDA Analysis Task 1.

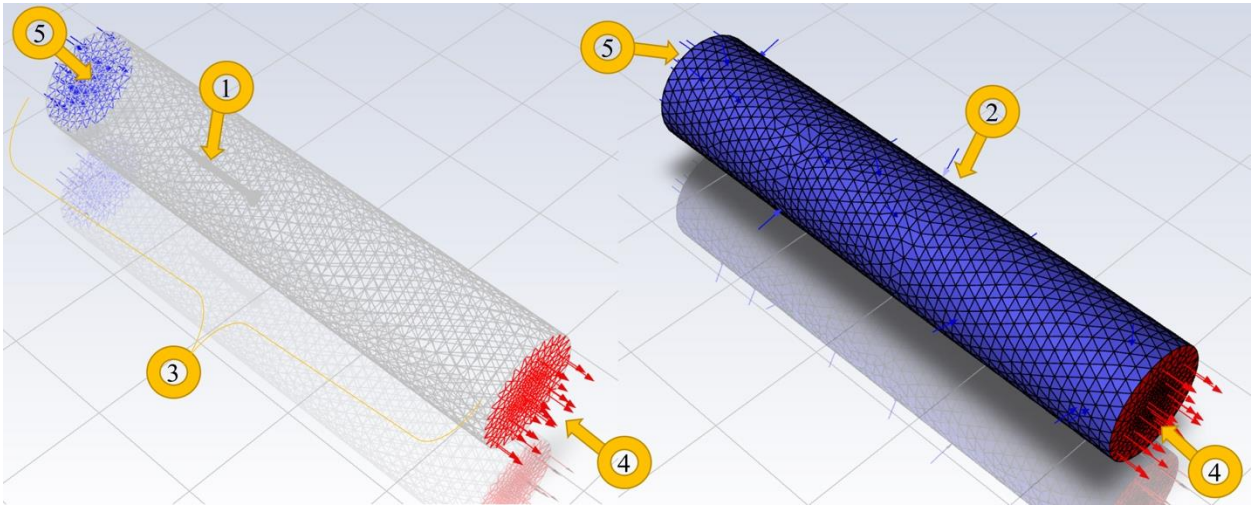


Figure 4-22 Rocket Ascent Set Up

Table 4-3 Boundary Conditions for Rocket Ascent Model

Boundary Condition (Figure 4-22)	Boundary Condition Type	Associated Values
1	Wall (Rocket Body)	See Table 2-2

2	Velocity Inlet	V = Velocity = 280 ft/s = 85.34 m/s Flow X Component = cos (α) Flow Y Component = sin (α) Flow Z Component = 0
3	Interior	Considered a zone, the fluid interior, rather than a boundary condition
4	Velocity Inlet	V = Velocity = 280 ft/s = 85.34 m/s Flow X Component = cos (α) Flow Y Component = sin (α) Flow Z Component = 0
5	Pressure Outlet	See Table 2-2

Once the initial setup was complete the case was initialized. Calculations were than run for multiple iterations in the hopes that chosen values would all converge to an answer. However, the first time this was attempted the values would not converge and instead oscillate continually. This was due to a reverse flow occurring at the back end of the geometry. Through examining the Fluent discussion forum and the Ansys Fluent Manual it was discovered that this often occurs when the outlet is too close to the body of interest, in this case the rocket body. In order to resolve this problem, the back end of cylindrical geometry was extended. By modifying the geometry, the mesh began to run into errors with the quality being poor and taking too long to run. After many attempts the geometry of the stability fins and the grid fins was simplified to resolve the problem.

Once the new mesh was created and the setup was completed a new case was ran. Another error became apparent. The contact regions between the outer and inner cylinder and the sliced

plane were appearing as wall resulting in incorrect data. This problem was fixed by going back to the geometry and making all sections of the geometry a single part therefore solving the problem of interface walls. Once all issues were resolved, four cases were run for angles of attack ranging from 0 degrees to 15 degrees. The aerodynamic load data was then integrated into the simulator described in Section 4.2.1. The relationship between lift and the angle of attack for each case is shown in Figure 4-23.

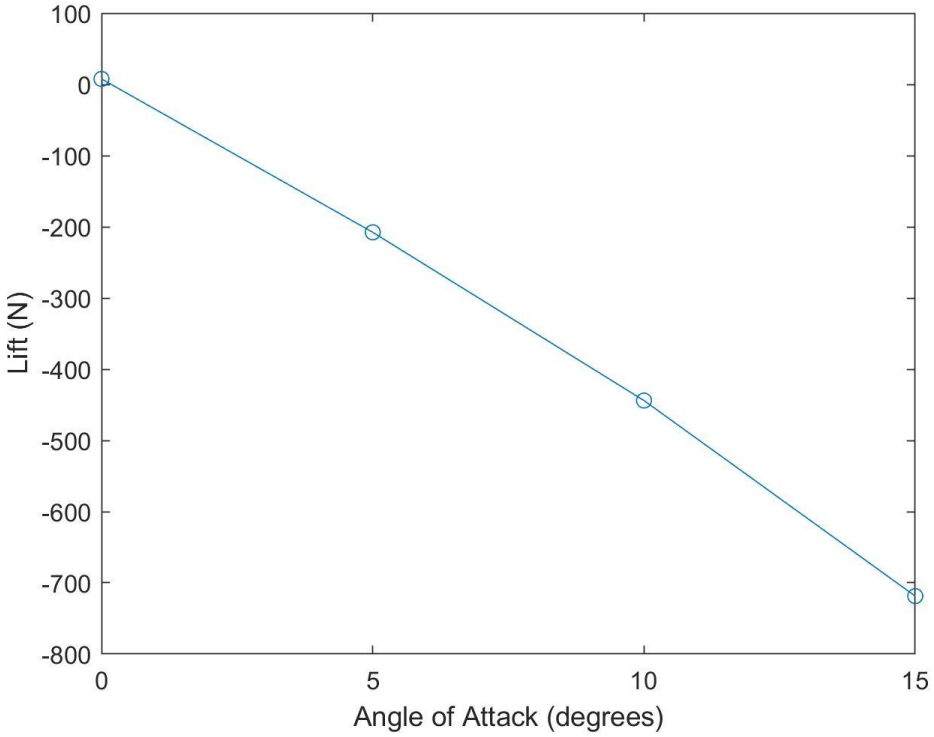


Figure 4-23 Plot of Lift as a Function of Angle of Attack

From the small data sample analyzed, the lift follows the most linear path found between each angle of attack. The lift data was plotted against the angle of attack and a polynomial fit line was used to connect the data. The negative sign and the apparent trend that the lift decrease is due to the sign convention of the model in the relative body fixed frame of reference as indicated in Figure 4-21 above. The negative sign indicates that the lift force is in the opposite direction of the free stream velocity. The absolute value of the lift force increases as the angle of attack increases due to the y component of the free stream velocity increasing in relation to the angle of attack, therefore the pressure gradient increases and causes this effect.

Graphs for other data found through Ansys Fluent do not appear as linear as the relation for the lift. For a small angle of attack the relation between the data should appear fairly linear although this does not appear to be the case. This is demonstrated in Figure 4-24, the yawing moment as a function of angle of attack, and Figure 4-25, the coefficient of the roll moment as a function of angle of attack. However, the margins between each data point are relatively small and perhaps if the sample size was larger the relation between the variable and angle of attack would be more apparent. From the appearance of the data, there is possibly an instability somewhere, potentially due to the asymmetry of the rocket. The graphs for the drag force, the other moments and the other moment coefficients were also created and are found in Appendix H: Ansys Fluent Aerodynamic Load Plots.

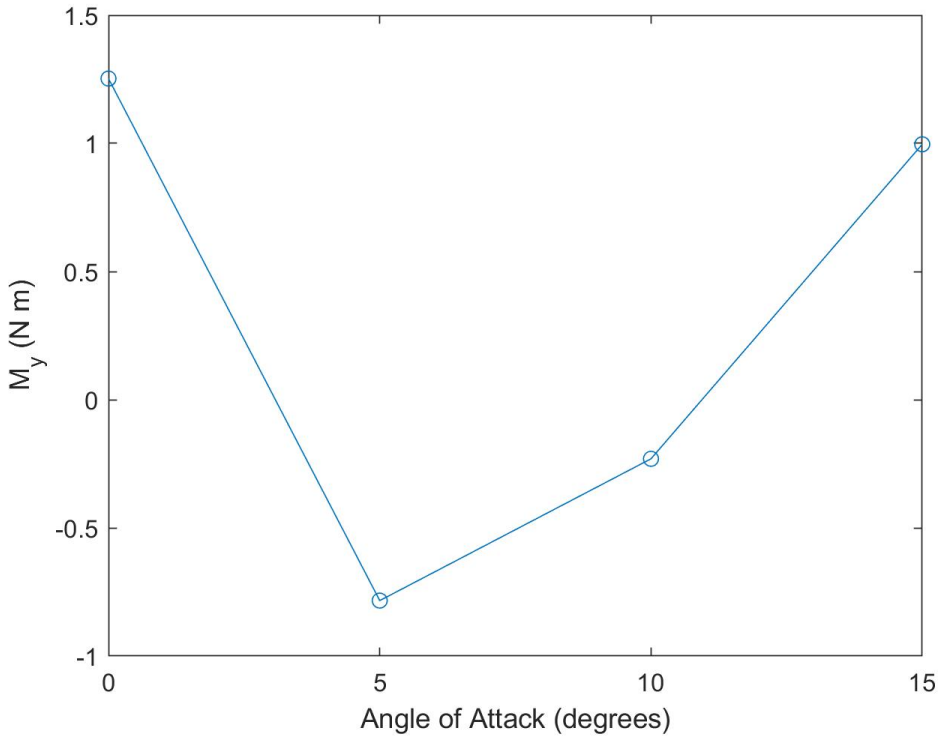


Figure 4-24 Plot of Y Moment as a Function of Angle of Attack

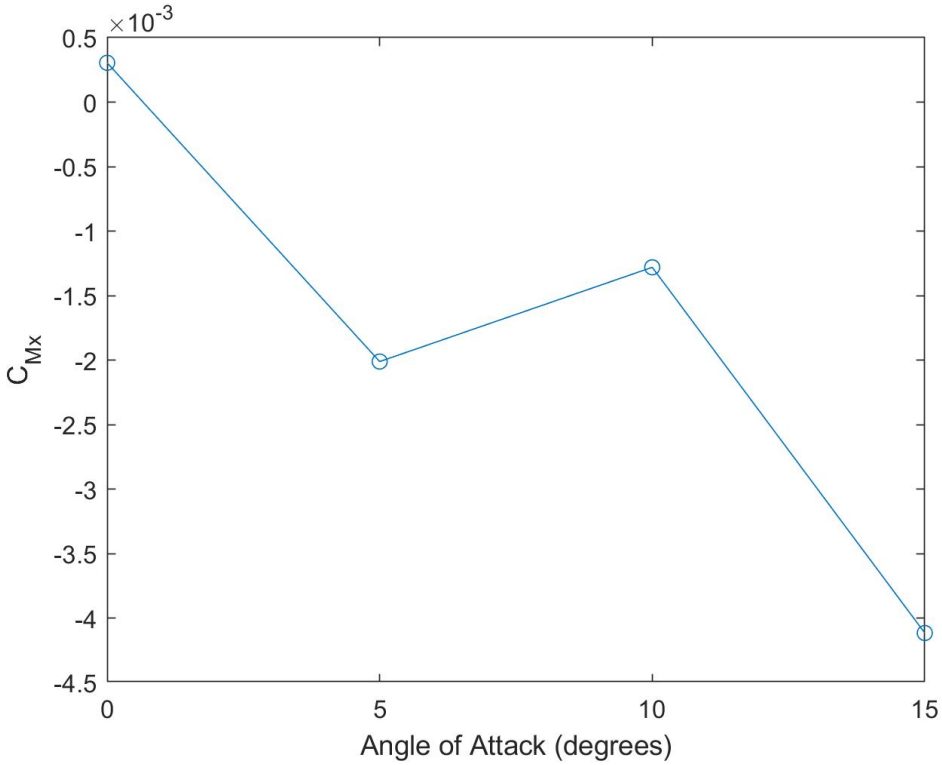


Figure 4-25 Plot of X Moment Coefficient as a Function of Angle of Attack

The data that was collected from Ansys Fluent was the lift and drag forces and the moments in the x, y, and z direction. When calculating the forces, the direction of the lift and drag where labeled based on the rocket body frame of refence as noted in Figure 4-21. The moments were calculated using an estimated center of mass of -0.61 m or - 2 ft in the x component relative to the coordinate system origin as shown in Figure 4-21. This dimension was taken from the OpenRocket model to get the estimated moments. The direction of the moments was also considered in the rocket body frame of refence.

Although the forces and the moments were calculated in Fluent, the coefficients were calculated externally. This was done to reduce the run time for a single case to converge. It is possible to perform the coefficient calculations in Fluent by modifying the refence values, which include values needed for the dynamic pressure analysis, however the general coefficient of the aerodynamic forces and moments are simple enough to calculate that it was not necessary. The equations need needed to calculate force and moment coefficients are given by Eq. 39 and Eq. 40.

$$C_F = \frac{F}{\frac{1}{2}\rho V^2 S} = \frac{F}{QS} \quad (39)$$

$$C_M = \frac{M}{\frac{1}{2}\rho V^2 S\bar{c}} = \frac{M}{QS\bar{c}} \quad (40)$$

To calculate the coefficients there are a few variables that will be constant for the force (F) or moment (M). For the equations, the area (S) was taken to be the area of the rocket body tube and the cord length ($\bar{c}$) was taken to be the length of the body of the rocket both values were collected form the OpenRocket schematic. For both cases, the density (ρ) is the same and was considered to be the density of air at sea level. The free stream velocity (V) was also the same, 85.34 m/s or 280 ft/s and was not broken into components for lift and drag or moments in any direction. If the value of the density and the velocity are known it is possible to calculate the dynamic pressure using Eq. 41. This variable is helpful, when displayed as a contour, to determine if the data appear physically correct.

$$Q = \frac{1}{2}\rho V^2 \quad (41)$$

Dynamic pressure contour plots were created for each angle of attack case and are presented in Appendix I: Ansys Fluent Dynamic Pressure Contours. These plots appear correct for the given conditions. A comparison of the dynamic pressure contours for zero angle of attack and fifteen-degree angle of attack are shown in Figure 4-26. As shown in the figure the higher dynamic pressure, the warm color in the figure, appears around sharper edges such as the joint between the nose cone and body tub in addition to the blunt sides such as the top of the grid fins and stability fins. This is due to the air being compressed and having a higher velocity at those location. Whereas the locations with lower dynamic pressure depicted by the cool colors in the figure, have lower velocities and at some locations pressure vacuums occur such as behind the grid fins, stability fins, and as a whole the body of the rocket. It is also apparent that as the angle of attack increases the dynamic pressure on the right side of the rocket, in the path the free stream velocity, decreases and the dynamic pressure on left side of the rocket, the side facing the free stream velocity, increases.

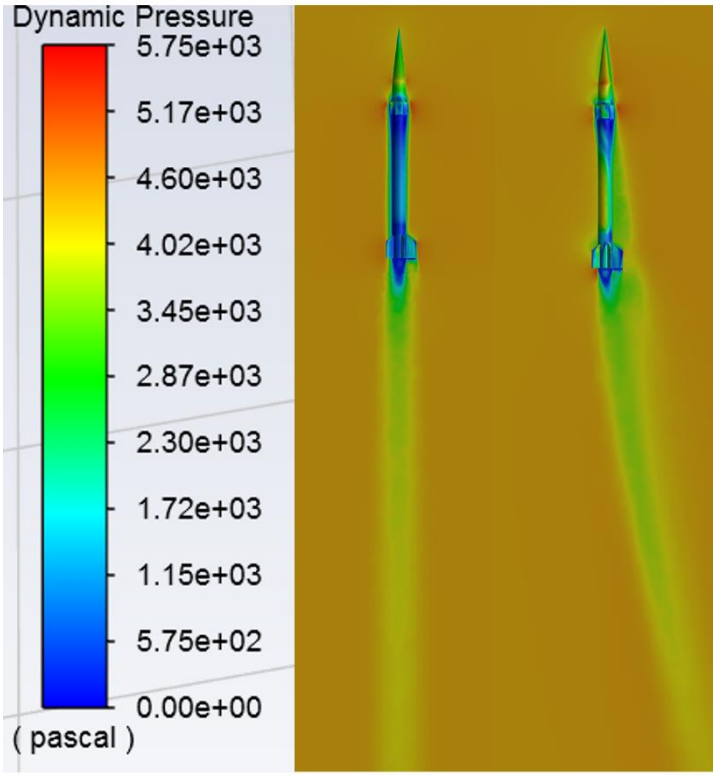


Figure 4-26 The Dynamic Pressure Contours at 280 ft/s

5 Summary, Conclusions, Recommendations, Broader Impacts

The following section provides a summary and conclusion for each of the subteam's analysis tasks. A path forward and recommendations are provided for future projects working on a similar project. The broader impact of model rocketry is also assessed.

5.1 Airframe and Recovery System

5.1.1 Analysis Task 1 (Airframe Stress Distribution)

The goal of analysis task was to identify critical locations of high stress throughout the airframe and internal structure during peak acceleration loads and was completed successfully. This task proved to be the most difficult as communication between all subteams was needed for successful completion. In order to complete this task, the airframe as well as the internal bulkheads had to be CAD modeled. In order to model the separation system and cold gas thruster system bulkheads, communication between the ARS and PTSS subteams was needed.

This task ended up being completed last, after completion of the tasks focusing on grid fins. In hindsight the CAD model of the airframe and communication between teams should have taken place earlier in the project. Since this task was completed last, it prevented the FDA subteam from obtaining values from the airframe model. Therefore, it is recommended for future teams that this task be completed earlier or at least that a valid CAD model of the airframe be completed early in the project.

5.1.2 Analysis Task 2 (Grid Fin Aerodynamic Loads)

The analysis task to evaluate the aerodynamic forces and moments on a single grid fin was completed successfully. In order to complete both this task and ARS Analysis Task 2, a model of the grid fin and all parts involved was created. Once the grid fin was created in solid works a simulation was created in Fluent to show the aerodynamic loads across the grid fin. The results of this task were then used in the vehicle dynamics model and were used as an input in ARS Analysis Task 3.

A couple of problems did arise during this task, the largest was difficulties meshing in Fluent and getting the simulation to run properly. These problems came mostly from our subteam's inexperience with the Fluent program. Therefore, for future teams, it is recommended that more time is dedicated to learning the Fluent program before an analysis tasks is started in order to expedite the process.

5.1.3 Grid Fin Stress Distribution (Analysis Task 3) (Grid Fin Stress Distribution)

The goal of this analysis task was to identify critical locations of high stress throughout the grid fin during peak acceleration loads was also completed successfully. The results (aerodynamic loads) from ARS Analysis Task 2 were used as an input for the Ansys simulation of the grid fin. The results obtained proved that the grid fin could withstand a descent at terminal velocity and the material would stay intact.

In order to perform this task similar steps were taken as in ARS Analysis Task 2. The same CAD model was used in Task 3 and 2, as well as the aerodynamic loads mentioned above. It is recommended, similar to Task 2, that effort is put into learning Ansys earlier in the project.

5.2 PTSS

5.2.1 Analysis Task 1 (Motor Performance Analysis)

The results from the Cantera model were convincing but could be significantly improved upon. The analysis conducted here only included combustion between the oxidizer, ammonium perchlorate, and the fuel, aluminum. In reality there is a third component in the rocket motor fuel grain that was mentioned in the datasheet for the motor: the binder. A typical binder for SRMs is HTPB which contributes to the combustion reaction as a less efficient fuel than aluminum. If appropriate thermodynamic data is found this could also be added to the Cantera model.

Many assumptions went into the Cantera model to simplify the model including zero heat loss, no reactions amongst reactant species and only major species were considered. It was not obvious if there was a way that heat loss could be included in the Cantera model. One option may be to initially calculate the equilibrium temperature in Cantera by assuming zero heat loss. Then use this temperature to determine a heat source within the COMSOL model which will calculate the combustion temperature with heat loss. Afterwards, the Cantera model could be updated by holding temperature and another variable constant using the temperature from the COMSOL model as an input. This process could be iterated until the temperature from Cantera and COMSOL converge.

While learning Cantera it was found that adding chemical reactions between the product species is fairly straightforward and requires few modifications to the input file. The National Institute of Standards and Technology (NIST) provides a kinetics database which can be used to find the necessary information for chemical reactions that need to be included in the Cantera input file [113]. This addition could further improve the Cantera model by considering chemical reactions within the combustion gas. It would also be fairly simple to include more trace species in the chemical reaction. One of the sample input files provided when Cantera is downloaded contains data for all species retrieved from a NASA database. Simply copying over the desired species to the input file being used could provide more accurate results in the Cantera model.

The input file used for the Cantera analysis is provided in Appendix C: Cantera Model Input File. Figuring out how to create this input file was difficult at the beginning of this project. It was useful to explore the sample files that were provided by Cantera to gain an understanding

of what data was required for the input file and how the files were formatted. The NASA ThermoBuild database was useful in retrieving the necessary thermo data required for the species in the Cantera model [88].

5.2.2 Analysis Task 2 (Thermal Model)

The thermal model in COMSOL appears to have provided successful results as the temperature on the outside of the motor case significantly increases as expected but remains within the allowed limits. One improvement on the Cantera model would be to include graphite nozzle geometry in the axisymmetric geometry of the model. This would provide a more realistic outlet boundary condition and include more detail on the nozzle flow.

The COMSOL model can be improved further by considering the ambient conditions around the motor case. One assumption may be that air, at ambient temperature, is flowing past the motor case at the velocity that the rocket is traveling. This assumption would be used as the motor geometry is held at a constant position in the simulation.

5.2.3 Analysis task 3 (Mechanical Separation System Model)

The mechanical stage separation was successful during the motion study, both in securing the two model rocket stages together as a safety mechanism utilizing 3D printed clasps as well as separating the two stages using linear actuators. However, the motion study was not able to incorporate the centering ring and bulkhead in which the four linear actuators would be mounted in during a real test flight.

The stage separation mechanism could have been improved by simplifying the system in such a way that it more efficiently separated the two stages. Ideally, only one system would need to be used to both act as a safety mechanism and still physically separate the booster stage from the upper stage at apogee. This was the initial goal for the separation system, but it was revised due to the lack of space within the airframe and after multiple design iterations. It also would have been helpful to test the separation system with the 3D printed centering ring and bulkhead for a more accurate motion study. Lastly, the SOLIDWORKS motion study was unable to be completed due to the complexity and shear number of components in the model.

Future teams should investigate a simpler method of separation. Although the mechanical methods started out as a simple clasp separation model, numerous revisions further and further complexified the system, making it unnecessarily complicated. One major challenge presented with clasps is the translation of vertical motion to horizontal motion. This proved to be especially difficult when working with such a limited space.

5.2.4 Analysis task 4 (Cold Gas Descent Thruster Model)

Overall, the cold gas thruster descent model was successful in designing a system that could land the rocket safely. It was determined that the cold gas thruster flow rate was high enough to provide the thrust levels required.

The model could be improved by analyzing the flow through the system. This was not done due to lack of data for the regulators used for paintball guns. Also, there was assumed to be no leakage from the tank and regulators. For a number of reasons, the actual performance of the cold gas system would have delivered lower thrust and lower total impulse.

One last improvement that could made in designing of the cold gas thruster system is to use a carbon overwrapped pressure vessel. The rocket overall would have been lighter, and the cold gas thruster would have better performance.

Future teams should consider analyzing the flow through the system. They should also try to design around parts that come with data. The completion of the cold gas thruster descent model was delayed due to not having much in terms of component (i.e. regulator, valve, tank) performance data.

5.3 Flight Dynamics Analysis

5.3.1 Analysis task 1 (Vehicle Dynamics and Performance Model)

Overall, the dynamical simulator was successful as it was able to compute trajectory as well as attitude over the course of the flight. The full simulation behaves in a manner consistent with the flight of real model rockets. By integrating the COM and Euler simulators, the dynamical simulator no longer needed to make the zero angle of attack assumption used in previous years.

The primary point of concern with the simulation is the method through which aerodynamic moments are calculated. Due to the challenges discussed in Section 4.2.2, there was very little time to work with the CFD data and incorporate it into the simulator. While the current method produces expected behavior, there is a radians unit in the equation that is not cancelled out which likely means the moments are stronger than they should be. A solution was attempted by trying to convert the angles to nondimensional ratios, however, time ran out before a working implementation could be created.

Future teams should look into finding more efficient and accurate methods of incorporating CFD results within the dynamical simulator. It may be desirable to create a new reference frame in which matches the CFD simulations. This would allow the moments to be calculated directly from the CFD results without splitting them into components before being rotated back to the $\mathcal{B}$ frame. Additionally, it is recommended that an altitude model for density be implemented. While the effects of altitude on density are not very significant in the low altitude flights of Andromeda, this is an easy way to extend the simulator to work at much higher altitudes such as those achieved by higher power models and experimental rockets. It may also be a good idea to split the singular aerodynamic force used in this dynamical simulator into its lift and drag components. Future students should also be careful to manage complexity expectations. The original plans for the dynamical simulator included plans to individually compute forces and moments on each fin, the number of which, would change in flight. This, along with some other features, ultimately didn't fit into the allotted time. It is important to set reasonable expectations to balance fidelity with practicality.

It is also strongly recommended that future teams continue to follow the object-oriented paradigm. This significantly improved the ease of working on the code base. While the whole simulator could have technically been written all in one massive state derivative function, the structured nature of the code made navigation logical. This also makes it easy to unit test individual objects and functions to ensure that they behave correctly within the full dynamical simulator. This is a software project, so it is important to utilize software engineering best practices, including unit testing and unified modeling language (UML) diagrams, to ensure a well-managed, documented, and functional final project.

Lastly, it is recommended that future MQP teams look into using document writing programs better suited to large reports. This year, Microsoft Word was used which, while it has many excellent features such as automatic tables of contents and equations, its nature as a WYSIWYG editor means that it is easy to mess up formatting. On a number of occasions, often near deadlines, formatting anomalies necessitated roll backs to previous states. For this, future teams should consider switching to LaTeX which is immune to these problems. While LaTeX can take some time to learn, this time would likely be less than the time spent on trying to solve formatting and deleted work issues in Word. Collaboration can be handled using a GIT repository such as those hosted by GitHub or through synchronous editors such as Overleaf.

5.3.2 Analysis Task 2 (Vehicle Aerodynamic Loads Simulation)

Overall, the data from the Ansys Fluent model appears valid. The dynamic pressure contour graphs look correct with higher pressure on the sharp edges and blunt sides of the rocket and low pressure behind the rocket or when the body of the rocket is blocking some of the free stream velocity. The aerodynamic load values are within a believable range of values. However, it is difficult to tell with the current trendline for the aerodynamic load data if they appear correct due to the small sample size. In the future, more data should be acquired for both a larger span of angles of attack and a range of velocity to help determine if the relationship seems correct.

There are however a few points of concern with the current model. At zero angle of attack the rocket still experiences a lift force 7.7 N. This is an issue because at zero angle of attack there should not be a lift component. In part due to the lift the moments for the rocket at zero angle of attack where not zero. The x, y, and z or roll, yaw, and pitch moments for the zero angle of attack ranged from around 0.02 N m to 4 Nm. This seems to imply that there was an error somewhere with the model or perhaps the asymmetry of the rocket could be causing this effect.

Ansys Fluent is a complex software tool and there is a large learning curve with it. Meshing the geometry proved to be a very challenging and time-consuming process. Many errors and warning with the geometry being meshed occurred over the course of the project and a lot of researched was conducted to try and resolve some of the problem. Overall quality of the mesh was fine but could have been improved with more time. Due to the quality of the mesh and a few other

factors that might have been missed when setting up the model this could have resulted in the errors for the results.

For the future project it is recommended to not design the rocket to be asymmetrical. This type of design makes modeling the rocket more complex and gets ride of the option to model a more detailed rocket geometry. This is due to the ability of modeling a horizontal sliced section of the rocket that could focus on the detail of that one section not being possible in the case of asymmetry. A sliced section that is symmetrical allows for a more details mesh and eventual data that can be applied to symmetric parts initially suppressed in the model. In addition, in order for the meshes to not take too long the detail of the rocket geometry may have to be sacrificed for an asymmetrical case. The process of creating an adequate mesh for the chosen geometry can be a time consuming and iterative process, so when it is possible simplifying the case is helpful. In general, the recommendation when first starting modeling is to simplify the geometry as much as possible. Once results are being acquired that make sense, the geometry can begin to become more detailed and the process can be attempted with the new geometry.

5.4 Broader Impacts

Throughout the design and analysis process of the high-powered model rocket, our team encountered a variety of societal, environmental and economic impacts of propulsively landing rockets.

During the project we learned how expansive the model rocket community is online. As we learned early on in the project, model rocket motors are available in a wide range of sizes and can be used easily in smaller model rocket kits or can be used in far more complex model rockets such as the one designed by our team. This makes model rocketry attractive to parents and kids who are interested in getting into engineering at a young age and can put them down a path to making a career out of engineering. High-powered model rocketry also appeals to academic groups or engineering professionals that want to learn and gain more experience in aerospace engineering. This leads to applying more complex engineering concepts to model rocketry which can lead to the development of innovative systems such as attitude control systems or descent propulsion to land a model rocket.

In terms of environmental impacts, model rockets can contribute pollution from propellant exhaust that emits black carbon into the stratosphere. It is also possible for these toxic chemicals to enter the soil, water, and ecosystem as a whole. As a result, they can negatively impact human health. However, using a cold gas thruster is one clean method that does not release pollutants. Similarly, there are other clean fuel sources that various types of model rockets use that have minimal to no negative health and environment implications. This was a theoretical project to see if it would be plausible to create a reusable rocket, so no harmful fuel was released or negative environmental impacts realized. Reusability is another significant aspect of environmental impacts. It reduces carbon footprint and has less of an impact on the environment, especially the ozone.

On a larger scale, the technology for self-landing rockets also has many economic implications. Launch vehicles are very expensive, requiring large amounts of resources and work hours to construct and fly. The ability to land rockets propulsively enables the vehicles to be reused multiple times. This can significantly reduce the operating costs of launch operators which leads to a reduction in cost per kilogram for payloads. As of the time this paper was written, SpaceX is the only company to fly a propulsively landed rocket, the Falcon 9, to deliver payloads to orbit. However, in the coming years it is possible that prices could further drop due to competition as Blue Origin has plans to fly New Glenn a similar, although larger, landing rocket [114].

6 References

1 T. Benson, "Flight of a Model Rocket," NASA GRC, June 2014. [Online]. Available: https://www.grc.nasa.gov/WWW/K-12/rocket/rktflight.html.

2 M. Newton, "National Association of Rocketry," January 2012. [Online]. Available: https://www.nar.org/wp-content/uploads/2014/03/NAR-Rocketry-Basics.pdf. [Accessed October 2020].

3 NAR, "National Association of Rocketry," 2021. [Online]. Available: https://www.nar.org/high-power-rocketry-info/. [Accessed 14 03 2021].

4 T. V. Milligan, "Apogee Rockets," 20 September 2016. [Online]. Available: https://www.apogeerockets.com/education/downloads/Newsletter426.pdf. [Accessed October 2020].

5 "High Power Rocket Safety Code," National Association of Rocketry, August 2012. [Online]. Available: https://www.nar.org/safety-information/high-power-rocket-safety-code/.

6 "Blue Tube Facts," Always Ready Rocketry., 2019. [Online]. Available: https://alwaysreadyrocketry.com/blue-tube-2-0/. [Accessed 7 October 2020].

7 SpaceX, *Falcon User's Guide,* Space Exploration Technologies Corp. , 2020. [Online]. Available: https://www.spacex.com/media/falcon_users_guide_042020.pdf

8 Dings, A., Cooper C Y-L., George, E., Fennick, J., Foster, K., Lapierre, N., Dohn, P., Moquin, T. "Design and Integration of a High-Powered Model Rocket-I," WPI Major Qualifying Project (MQP). Report NAG-1901, Advisor: J. Blandino, 2019.

9 C. Velledor, A. Wimmer and R. DeHate, "MIT Rocket Team Preliminary Design Review," Massachusetts Institute of Technology, 2011.

10 "Level 2 Certification allows flyers to fly High Power Rockets with a total installed impulse between 640.01 and 5120.00 n-sec," Tripoli Rocketry Association, Inc., 2020. [Online]. Available: http://www.tripoli.org/Level2#:~:text=Level%202%20Certification%20allows%20flyers,a%20level%202%20certification%20flight.&text=Scratch%20built%20rockets%20may%20contain%20commercially%20built%20components.

11 U. Berkeley., "UC Berkeley Space Technologies and Rocketry Preliminary Design Review Project U.R.S.A.," 2016. [Online]. Available: https://stars.berkeley.edu/assets/files/CalSTAR_PDR.pdf.

12 NAR, "Laws & Regulations," National Association of Rocketry, 2020. [Online]. Available: https://www.nar.org/find-a-local-club/section-guidebook/laws-regulations/. .

13 T. Milligan, "Drag of Nose Cones," Apogee Components, 2013. [Online]. Available: https://www.apogeerockets.com/education/downloads/Newsletter346.pdf.

14 T. Milligan, "The Different Rocket Recovery Techniques," Apogee Components, July 2017. [Online]. Available: https://www.apogeerockets.com/education/downloads/Newsletter447.pdf.

15 J. Manfredo, "Properly Sizing Parachutes for Your Rockets," Apogee Components, 2005. [Online]. Available: https://apogeerockets.com/education/downloads/Newsletter149.pdf. [Accessed 2020].

16 L. Blackmore, "Autonomous Precision Landing of Space Rockets," *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2016 Symposium,* pp. 33-41, 2016.

17 "New Shepard," Blue Origin, 2020. [Online]. Available: https://www.blueorigin.com/new-shepard/. [Accessed 2020].

18 F. Hiroshima and K. Tatsumi, "Grid pattern effects on aerodynamic characteristics of grid fins.," in *4th International Congress of the Aeronautical Sciences*, 2004.

19 T. Dodd, "Will the Falcon 9 Actually Be Reusable," Everyday Astronaut, 2018. [Online]. Available: https://everydayastronaut.com/will-the-falcon-9-actually-be-reusable-or-just-refurbish-able-like-the-space-shuttle/.

20 S. Manuwar, "ANALYSIS OF GRID FINS AS EFFICIENT CONTROL SURFACE IN COMPARISON TO CONVENTIONAL PLANAR FINS," *International Congress of the Aeronautical Sciences,* pp. 1-6, 2010.

21 P. Theerthamalai, "Aerodynamic Characterization of Grid Fins at Subsonic Speeds," *Journal of Aircraft,* vol. 44, no. 2, pp. 694-697, 2007.

22 G. P. Sutton and O. Biblarz, *Rocket Propulsion Elements*, 9th ed., Hoboken, New Jersey: John Wiley & Sons, Inc., 2017.

23 R. A. Braeunig, "Basics of Space Flight: Rocket Propellants," 2008. [Online]. Available: http://www.braeunig.us/space/propel.htm.

24 System Solaire , "Liquid Propellant Rocket Kit," 2020. [Online]. Available: https://store.fastcommerce.com/SystemeSolaire/liquid-propellant-rocket-kit-ff8081811928eb610119331daa8d6729-p.html.

25 Apogee Components, Inc. , "Cesaroni Propellant Kits," 2019. [Online]. Available: https://www.apogeerockets.com/Rocket_Motors/Cesaroni_Propellant_Kits.

26 Pratt Hobbies, Inc. , "Products," [Online]. Available: https://www.pratt-hobbies.com/categories.asp?cat=21.

27 ESRA, "What is IREC?," [Online]. Available: http://www.soundingrocket.org/what-is-irec.html.

28 Apogee Rockets, Inc., "Rocket Motor Basics," 2019. [Online]. Available: https://www.apogeerockets.com/Rocket-Motor-Basics-Quick-Start-Guide.

29 Cesaroni Technology Inc., "Safety Data Sheet: Pro-X Rocket Motor Reload Kits & Fuel Grains," 2015. Available: https://www.apogeerockets.com/downloads/MSDS/Cesaroni/Cesaroni_Propellant_SDS.pdf

30 Apogee Components, Inc. , "How Composite Rocket Engines Work," 2019. [Online]. Available: https://www.apogeerockets.com/Tech/How_Composite_Rocket_Engines_Work.

31 National Association of Rocketry , "Standard Motor Codes," 2020. [Online]. Available: https://www.nar.org/standards-and-testing-committee/standard-motor-codes/.

32 T. V. Milligan, "How 2-Stage Rockets Work," 2019. [Online]. Available: https://www.apogeerockets.com/Tech/How_2-Stage_Rockets_Work.

33 Apogee Components, Inc., "Electronic Staging of Composite Propellant Rocket Motors," *Apogee Peak of Flight Newsletter*, no. 91, 28 October 2002. Available: https://www.apogeerockets.com/education/downloads/Newsletter98.pdf

34 F. Cain, "Why Do Rockets Need Stages? The Quest to Build a Single Stage to Orbit (SSTO)," 2017. [Online]. Available: https://www.universetoday.com/135367/rockets-need-stages-quest-build-single-stage-orbit-ssto/#:~:text=Because%20the%20amount%20of%20fuel,of%20the%20empty%20fuel%20tanks..

35 "Amateur Rocket Motor Ignition & Igniters," [Online]. Available: http://www.jacobsrocketry.com/aer/ignition_and_igniters.htm.

36 D. Cook, "Model Rocket Igniter Controller," [Online]. Available: http://robotroom.com/Rocket-Ignition-System-1.html.

37 Apogee Rockets, Inc., "Advanced Construction Videos," 2019. [Online]. Available: https://www.apogeerockets.com/Advanced_Construction_Videos/Rocketry_Video_246.

38 T. V. Milligan, "How to Design & Build Engine Mounts," *Apogee Peak of Flight Newsletter*, no. 104, 1 June 2003.

39 S. Testé, A. Torp, D. Zanko, L. Miguel Rodriguez, A. Colombo, B. Balfanz, B. Prats and C. Williams, "AstroJays Technical Report," John Hopkins University, Baltimore, 2018.

40 C. T. Lyne, "Calibration of Testing of Rapid Prototyped Nozzles for Guidance and Control," *Vanderbilt University*, 2017.

41 R. J. Shaw, "Dynamics of Flight," NASA GRC, 12 June 2014. [Online]. Available: https://www.grc.nasa.gov/www/k-12/UEET/StudentSite/dynamicsofflight.html. [Accessed 18 10 2020].

42 T. Benson, "Rocket Stability," 12 June 2014. [Online]. Available: https://www.grc.nasa.gov/WWW/k-12/rocket/rktstab.html.

43 J. D. J. Anderson, *Fundementals of Aerodynamics*, New York City: McGraw-Hill , 2017.

44 T. Benson, "Conditions for Rocket Stability," NASA GRC, 12 June 2014. [Online]. Available: https://www.grc.nasa.gov/WWW/k-12/rocket/rktstabc.html. [Accessed 18 October 2020].

45 T. Van Milligan, "Model Rocket Stability," Apogee Components, [Online]. Available: https://www.apogeerockets.com/Peak-of-Flight/Newsletter462.

46 T. Benson, "Rocket Rotations," NASA GRC, 12 June 2014. [Online]. Available: https://www.grc.nasa.gov/WWW/K-12/rocket/rotations.html. [Accessed 18 October 2020].

47 T. Benson, "Practical Rocketry," NASA GRC, 12 June 2014. [Online]. Available: https://www.grc.nasa.gov/www/k-12/rocket/TRCRocket/practical_rocketry.html. [Accessed 18 October 2020].

48 F. J. Gomez and R. Miikkulainen, "Active Guidance for a Finless Rocket Using Neuroevolution," in *Genetic and Evolutionary Computation Conference*, 2003.

49 D. Wyatt, "An Actively Stabilised Model Rocket," Technical Report, University of Cambridge, 2006.

50 T. Benson, "Examples of Control," NASA GRC, 14 June 2014. [Online]. Available: https://www.grc.nasa.gov/WWW/K-12/rocket/rktcontrl.html.

51 R. Ferrante, "A Robust Control Approach for Rocket Landing," Technical Report, University of Edinburgh, 2017.

52 A. Ingabire and A. Sklyarov, "E3S Web of Conferences," *Control of longitudinal flight dynamics of a fixedwing UAV using LQR, LQG and nonlinear control,* vol. 10, no. 4, 2019.

53 A. Hyland, Z. Huaman, N. Amato, j. Procaccini, J. Pickunka, J. Koslow, S. Ranjit, W. Roe, J. Romankiw, D. Santamaria, J. Scarponi, B. St.Jacques and J. Tappen, "Design, Analysis, and Test of a High-Powered Model Rocket (JB3-2001),", Worcester Polytechnic Institute, 2020, Advisor: J. Blandino.

54 H. Henrique, L. C. Lopes and C. H. Silva, "Experimental Application of Predictive Controllers," *Journal of Control Science and Engineering,* vol. 18, 2012.

55 Barnard, Barnard Propulsion Systems LLC, 2020. [Online]. Available: https://bps.space/.

56 A. R. Alvarez, E. R. Kelly, H. G. Gerhardt, J. K. Whitehouse and J. R. O'Neill, "Design and Integration of a High-Powered Model Rocket-II," Worcester Polytechnic Institute, 2019, Advisor: J. Blandino..

57 S. Box, B. C. M and H. Hunt, "Stochastic Six-Degree-of-Freedom Flight Simulator for Passively Controlled High-Power Rockets," *Journal of Aerospace Engineering,* vol. 24, no. 1, 2010.

58 C. Moler, "Cleve's Corner: Cleve Moler on Mathematics and Computing," Mathworks, 26 May 2014. [Online]. Available: https://blogs.mathworks.com/cleve/2014/05/26/ordinary-differential-equation-solvers-ode23-and-ode45/.

59 S. D. Eilers and T. Whitmore, "The development of an infrastructure for end-to-end hybrid rocket flight simulation, motor and aerodynamicprediction and testing, and design analysis," Utah State University, Technical Report, 2008.

60 T. Lew, F. Lyck and G. Muller, "Chance-Constrained Optimal Altitude Control of a Rocket," in *8th EUCASS*, 2019.

61 P. K. Kundu, I. M. Cohen and D. R. Dowling, *Fluid Mechanics*, London: Academic Press, 2016.

62 N. Hall, "Boundary Layer," NASA GRC, 5 May 2015. [Online]. Available: https://www.grc.nasa.gov/WWW/k-12/airplane/boundlay.html.

63 D. Kuzmin, "Introduction to Computational Fluid Dynamics," [Online]. Available: https://www.mathematik.uni-dortmund.de/~kuzmin/cfdintro/lecture1.pdf.

64 N. Hall, "Navier-Stokes Equations," NASA GRC, 5 May 2015. [Online]. Available: https://www.grc.nasa.gov/WWW/k-12/airplane/nseqs.html. [Accessed 18 October 2020].

65 J. D. A. Jr., "Computational Fluid Dynamics as a Research Tool," in *Computational Fluid Dynamics The Basics with Applications*, New York City, McGraw-Hill Inc., 1995, pp. 6-9.

66 Simscale, "What Are Boundary Conditions?," 12 October 2020. [Online]. Available: https://www.simscale.com/docs/simwiki/numerics-background/what-are-boundary-conditions/#:~:text=Boundary%20conditions%20(b.c.)%20are%20constraints,of%20a%20boundary%20value%20problem.&text=They%20arise%20naturally%20in%20every,to%20be%20solved%20in%20time.

67 "What is CFD | Computational Fluid Dynamics?," SimScale, 5 June 2020. [Online]. Available: https://www.simscale.com/docs/simwiki/cfd-computational-fluid-dynamics/what-is-cfd-computational-fluid-dynamics/.

68 "Compare Ansys Fluent vs SimScale," G2, 2020. [Online]. Available: https://www.g2.com/compare/ansys-fluent-vs-simscale. [Accessed 18 October 2020].

69 Worcester Polytechnic Institute, "2020 – 2021 Student Code of Conduct: COVID-19 Addendum," Worcester Polytechnic Institute, Worcester, 2020. [Online]. Availible https://www.wpi.edu/news/announcements/wpi-student-code-conduct-covid-addendum. [Accessed 4 12 2021]

70 HiTec, "D89MW, 32-Bit, Wide Voltage, Metal Gear Servo," Hitec RCD USA, Inc, 2021. [Online]. Available: https://hitecrcd.com/products/servos/digital/d-series/d89mw-32bit/product. [Accessed 2021].

71 Formlabs, "Tough 1500 Resin," Formlabs, 2021. [Online]. Available: https://formlabs.com/store/materials/tough-1500-resin/. [Accessed 12 3 2021].

72 PJRC Teensy, *PJRC Teensy 4.1 Development Board.* [Online]. Available: https://www.adafruit.com/product/4622. [Accessed 2020].

73 Sir Gawain, "Sir Gawain G007 Mini Camera," Sir Gawain, 2020. [Online]. Available: https://www.sirgawain.net/minicamera. [Accessed 2020].

74 "Durable Plastic Nose Cone for High Power Rockets," Apogee Components, 2020. [Online]. Available: https://www.apogeerockets.com/Building_Supplies/Nose_Cones/High_Power_Nose_Cones/PNC_4_x_16-5_98mm#description.

75 "ANSYS Workbench Product Release Notes," ANSYS INC, 2005. [Online]. Available: https://kashanu.ac.ir/Files/Content/ANSYS%20Workbench.pdf..

76 "ANSYS FLUENT 12.0 Theory Guide," ANSYS, Inc., 2009. [Online]. Available: https://www.afs.enea.it/project/neptunius/docs/fluent/html/th/main_pre.htm.

77 N. Hall, "Mach Number," NASA GRC, May 2015. [Online]. Available: https://www.grc.nasa.gov/www/k-12/airplane/machrole.html.

78 "ANSYS FLUENT 12.0 User's Guide," ANSYS, INC., 2009. [Online]. Available: https://www.afs.enea.it/project/neptunius/docs/fluent/html/ug/main_pre.htm.

79 S. A. Kumar and K. B., "CFD analysis of missile with altered grid fins to enhance aerodynamic efficiency in subsonic flow regime," *International Journal of Engineering Inventions,* vol. 5, no. 4, pp. 44-53, 2016.

80 "3D Transonic Flow Over a Wing," ANSYS Inc,, 2020. [Online]. Available: https://courses.ansys.com/index.php/courses/transonic-flow-over-a-wing-3d/.

81 Formlabs Material Properties, "Material Data Sheet: Grey Pro," Formlabs, 2018. [Online]. Avalaible: https://support.formlabs.com/s/topic/0TO1Y000000IvreWAC/materials-form-2?language=en_US

82 Rocket Reviews, "CTI 1281K360-13A - Time-Thrust Curve and Data," Rocket Reviews, [Online]. Available: https://www.rocketreviews.com/cti-1281-k360-wh-13a.html. [Accessed 29 11 2020].

83 Cesaroni Technology Inc. , "Material Data Safety Sheet ProFire Igniter," 2007. [Online]. Available: http://www.pro38.com/MSDS/SDS--ProFire%20Igniter--ver4-01.pdf

84 ThrustCurve, "ThrustCurve Hobby Rocket Motor Data," 18 06 2011. [Online]. Available: https://www.thrustcurve.org/motors/Cesaroni/1281K360-13A/. [Accessed 11 28 2020].

85 D. G. Goodwin, H. K. Moffat and R. L. Speth, *Cantera: An Object-oriented Software Toolkit for Chemical,* 2018. [Online]. Available: https://cantera.org

86 Regulatory Affairs Department: Cesaroni Technology Inc., "Pro-X®Rocket Motor Reload Kits & Fuel Grains," 2015. [Online]. Available: https://www.apogeerockets.com/downloads/MSDS/Cesaroni/Cesaroni_Propellant_SDS.pdf. [Accessed 28 11 2020].

87 C. Snyder, "Chemical Equilibrium with Applications," NASA, [Online]. Available: https://cearun.grc.nasa.gov. [Accessed 28 11 2020].

88 C. Snyder, "NASA Thermo Build," NASA, 29 10 2020. [Online]. Available: https://cearun.grc.nasa.gov/ThermoBuild/. [Accessed 28 11 2020].

89 C.-C. Chen and M. McQuaid, "A Skeletal, Gas-Phase, Finite-Rate, Chemical Kinetics Mechanism for Modeling the Deflagration of Ammonium Perchlorate—Hydroxyl-Terminated Polybutadiene Composite Propellants," US Army Research Laboratory, 2016. Accession Number: AD1009001

90 S. Zhen and G. Davies, "Calculation of the Lennard-Jones n-m Potential Energy Parameters for Metals," *Physica Status Solidi,* vol. 78, no. 2, p. 595, 1983.

91 R. J. Kee, G. Dixon-Lewis, J. Warnatz, M. E. Coltrin, J. A. Miller and H. K. Moffat, "A FORTRAN COMPUTER CODE PACKAGE FOR THE EVALUATION OF GAS-PHASE, MULTICOMPONENT TRANSPORT PROPERTIES," 1998. [Online]. Available: https://www3.nd.edu/~powers/ame.60636/transport1986.pdf

92 Rocket Motor Components Inc. , "54mm Nozzle, 0.455" Throat," Rocket Motor Components Inc. , [Online]. Available: https://www.rocketmotorparts.com/54mm_Nozzle_0455_Throat/p1577809_7763027.aspx. [Accessed 28 11 2020].

93 Rocket Motor Components Inc. , "54mm Classic™ Propellant Grain," [Online]. Available: https://www.rocketmotorparts.com/54mm_Classic_Propellant_Grain/p1577809_18245202.aspx. [Accessed 11 29 2020].

94 D. M. Hanson-Parr and T. P. Parr, "Thermal Properties Measurements of Solid Rocket Propellant Oxidizers and Binder Materials as a Function of Temperature," *Journal of Energetic Materials,* 20 August 2006.

95 COMSOL, *CFD Module User's Guide,* Version 5.5 ed., 2019. [Online]. Available: https://doc.comsol.com/5.4/doc/com.comsol.help.cfd/CFDModuleUsersGuide.pdf

96 COMSOL, *Heat Transfer Module User's Guide,* Version 5.5 ed., 2019. [Online]. Available: https://doc.comsol.com/5.4/doc/com.comsol.help.heat/HeatTransferModuleUsersGuide.pdf

97 "Geox Powder, Inc.," 17 March 2009. [Online]. Available: https://www.epa.gov/sites/production/files/2015-05/documents/9530608.pdf.

98 "Atmospheric Pressure vs. Elevation above Sea Level," 2003. [Online]. Available: https://www.engineeringtoolbox.com/air-altitude-pressure-d_462.html.

99 "How to Size Ejection Charge," 17 05 2014. [Online]. Available: http://hararocketry.org/hara/resources/how-to-size-ejection-charge/.

100 "Shear Pin/Screw Calculations," [Online]. Available: http://www.feretich.com/Rocketry/Resources/shearPins.html.

101 "Small Nylon Shear Pins," [Online]. Available: https://www.apogeerockets.com/Building-Supplies/Misc-Hardware/Nylon-Shear-Pins-20-pack.

102 "Mcmaster.com," [Online]. Available: https://www.mcmaster.com/solenoid-valves/high-pressure-compact-solenoid-on-off-valves/.

103 National Fire Protection Association, "NFPA 1125: Code for the Manufacture of Model Rocket and High Power Rocket Motors," 2017. [Online]. Available: https://www.nfpa.org/codes-and-standards/all-codes-and-standards/list-of-codes-and-standards/detail?code=1125

104 "Engineersedge.com," [Online]. Available: https://www.engineersedge.com/fluid_flow/compressibility_factor_14171.htm.

105 "Flow Coefficient - Cv - for Liquid, Steam and Gas - Formulas and Online Calculators," 12 Feburary 2020. [Online]. Available: https://www.engineeringtoolbox.com/flow-coefficients-d_277.html.

106 T. V. Milligan, "What is the best fin shape for a model rocket?," *Peak of Flight,* 2 May 2017.

107 Featherweight Altimeters, *Raven4.* [Online]. Available: https://www.featherweightaltimeters.com/raven-altimeter.html. [Accessed 2020].

108 Freescale, *MPL3115A2 - I2C Barometric Pressure/Altitude/Temperature Sensor.* [Online]. Available: https://www.adafruit.com/product/1893. [Accessed 2020].

109 Adafruit, *Adafruit Ultimate GPS Breakout - 66 channel w/10 Hz updates - Version 3.* [Online]. Available: https://www.adafruit.com/product/746. [Accessed 2020].

110 Adafruit, *Adafruit 9-DOF LSM9DS1 Breakout Board - STEMMA QT / Qwiic.* [Online]. Available: https://www.adafruit.com/product/4634. [Accessed 2020].

111 GlobTek Incorporation, *Lithium Ion Polymer Battery - 3.7v 500mAh.* [Online]. Available: https://www.adafruit.com/product/1578. [Accessed 2020].

112 H. D. Curtis, *Orbital Mechanics for Engineering Students*, Fourth Edition, Cambridge: Butterworth-Heineman, 2020.

113 NIST, "NIST Chemical Kinetics Database," 2015. [Online]. Available: https://kinetics.nist.gov/kinetics/index.jsp.

114 Blue Origin, "New Glenn," Blue Origin, 2021. [Online]. Available: https://www.blueorigin.com/new-glenn/. [Accessed 2021].

115 A. Anis, "Cold gas propulsion system-an ideal choice for remote sensing small satellites," *Remote sensing-advanced techniques and platforms.*, 2012.

116 S. L. Nothnagel, "Development of a cold gas propulsion system for the Talaris hopper," MIT, 2011.

117 T. Benson, "Rocket Aerodynamic Forces," NASA GRC, 12 June 2014. [Online]. Available: https://www.grc.nasa.gov/www/k-12/UEET/StudentSite/dynamicsofflight.html. [Accessed 18 October 2020].

118 "Material Datasheets : Composites," Markforged, [Online]. Available: http://static.markforged.com/downloads/composites-data-sheet.pdf.

119 "Durable Plastic Nose Cone for High Power Rockets," Apogee Components, 2019. [Online]. Available: https://www.apogeerockets.com/Building_Supplies/Nose_Cones/High_Power_Nose_Cones/PNC_4_x_16-5_98mm#description.

120 O. B. George P. Sutton, *Rocket Propulsion Elements*, New York City: Wiley, 2010.

121 Merriam-Webster, "Merriam-Webster Dictionary," Merriam-Webster, [Online]. Available: https://www.merriam-webster.com/dictionary/phenolic. [Accessed 14 03 2021].

7 Appendices

7.1 Appendix A: Gantt Chart

The Gantt chart used for the project is significantly longer and more detailed than the one presented below. In order for ease of viewing, it was simplified to only show the first week of work done each term. All of the dates on the left column correlate with the general timeline of the tasks.

Team Gantt Chart

High Powered Rocket MQP

Project Start Date: 9/18/2020 (Friday) | Project Lead: | Display Week: 0

WBS	TASK	LEAD	START	END	DAYS	% DONE	WORK DAYS
1	**Full Team**						
0.1	**Introduction**	All	Fri 9/18/20	Sat 10/17/20	1	100%	21
0.0.1	Write First Draft of Introduction	All	Fri 9/18/20	Thu 10/08/20	21	100%	15
0.0.1	Peer Review	All	Thu 10/08/20	Tue 10/13/20	6	100%	4
0.0.1	Revise Introduction	All	Tue 10/13/20	Sat 10/17/20	5	100%	4
0.0.1	Final Edits to Introduction	All	Sat 10/17/20	Mon 10/19/20	3	100%	1
0.0.1	*Submit Introduction*	All	Tue 10/20/20	Tue 10/20/20	1	100%	1
0.1	**Methodology**	All	Wed 10/21/20	Tue 12/15/20	56	100%	40
0.0.1	Write First Draft of Methodology	All	Wed 10/21/20	Thu 12/03/20	44	100%	32
0.0.1	Peer Review Methodology	All	Fri 12/04/20	Mon 12/07/20	4	100%	2
0.0.1	Revise Methodology	All	Tue 12/08/20	Sat 12/12/20	5	100%	4
0.0.1	Final Edits to Methodology	All	Sun 12/13/20	Mon 12/14/20	2	100%	1
0.0.1	*Submit Methodology*	All	Tue 12/15/20	Tue 12/15/20	1	100%	1
0.1	**Results**	All	Thu 2/18/21	Fri 3/19/21	30	100%	22
0.0.1	Write First Draft of Results	All	Thu 2/18/21	Thu 3/11/21	22	100%	16
0.0.1	Peer Review Another Persons Paper	All	Fri 3/12/21	Sun 3/14/21	3	100%	1
0.0.1	Revise Results	All	Mon 3/15/21	Thu 3/18/21	4	100%	4
0.0.1	Final Edits to Results	All	Thu 3/18/21	Thu 3/18/21	1	100%	1
0.0.1	Finalize Results	All	Fri 3/19/21	Fri 3/19/21	1	100%	1
0.0.1	*Submit Results*	All	Sat 3/20/21	Sat 3/20/21	1	100%	0
0.1	***All Parts Ordered By***	All	Wed 11/11/20	Wed 11/11/20	1	100%	1
0.1	***MQP Completed***	All	Sat 3/20/21	Sat 3/20/21	1	100%	0
1	**ARS**						
0.1	**Task 1: Airframe Stress Distribution**		Thu 10/08/20	3/6/2021	150	100%	107
0.0.1	Design the airframe, bulkheads and full CAD assembly		Mon 10/12/20	Thu 2/18/21	130	100%	94
0.0.1	Completed Impact Simulation		Mon 3/01/21	Wed 3/10/21	10	100%	8
0.1	**Task 2: Grid Fin Aerodynamic Loads**		10/23/2020	10/25/2020	3	100%	1
0.0.1	CAD model of grid fin		Mon 9/28/20	10/7/2020	10	100%	8
0.0.1	Create FLUENT test matrix		Tue 10/20/20	10/24/2020	5	100%	4
0.0.1	Conduct FLUENT simulations for aerodynamic model		10/24/2020	11/2/2020	10	100%	6
0.0.1	3D print grid fin and verify FLUENT results in wind tunnel		Tue 10/20/20	11/8/2020	20	100%	14
0.0.1	Create control lookup tables from FLUENT results		11/8/2020	11/22/2020	15	100%	10
0.0.1	CAD model grid fin actuation mechanism		10/24/2020	11/2/2020	10	100%	6
0.1	**Task 3: Grid Fin Stress Distribution**		3/9/2021	3/28/2021	20	100%	14
0.0.1	Conduct ANSYS simulations to validate material choice and design		3/9/2021	3/28/2021	20	100%	14
3	**PTSS**						
0.1	**Task 1: Motor Performance Model**		10/21/2020	1/28/2021	100	100%	72
3.1.1	determine nozzle geometry, propellant properties, other inputs		10/21/2020	11/19/2020	30	100%	22
3.1.2	Formulate model connecting combustion chemistry and flow through nozzle		11/19/2020	12/13/2020	25	100%	17
3.1.3	Model Validation		12/13/2020	2/10/2021	60	100%	43
0.1	**Task 2: Temperature Distribution**		10/21/2020	1/28/2021	100	100%	72
3.2.1	CAD model motor, casing, nozzle, motor mount		10/21/2020	10/28/2020	8	100%	6
3.2.3	Determine Model Inputs		10/28/2020	12/16/2020	50	100%	36
3.2.4	run model in COMSOL		12/16/2020	3/5/2021	80	100%	58
0.1	**Task 3: Mechanical Separation System Model**		10/21/2020	12/2/2020	43	100%	31
3.3.1	Determine design requirements (size, weight, friction forces(experimentally))		10/21/2020	10/27/2020	7	100%	5
3.3.2	Complete CAD design of system		10/27/2020	12/16/2020	18	100%	37
3.3.3	Order materials		12/13/2020	3/2/2021	79	100%	57
3.3.4	Perform motion study		3/2/2021	3/13/2021	12	100%	9
3.3.5	Design bulkheads with ARS		3/2/2021	3/20/2021	8	100%	14
0.1	**Task 4: Cold Gas Descent Thruster Model**		9/20/2020	11/8/2020	50	100%	35
3.4.1	basic idealized model of flow through rocket nozzle		9/20/2020	10/10/2020	21	100%	15
3.4.2	flow model for pressure drop through valve		10/10/2020	10/23/2020	14	100%	10
3.4.3	size tank and components		10/23/2020	11/5/2020	14	100%	10
1	**FDA**						
0.1	**Task 1: Vehicle Dynamics and Performance Model**		Wed 10/21/20	Tue 3/09/21	140	100%	100
0.0.1	COM Simulator		Wed 10/21/20	Fri 12/11/20	51	100%	38
0.0.1	Euler Simulator		Sat 12/12/20	Sat 2/20/21	70	100%	50
0.0.1	Simulator Integration		Sun 2/21/21	Tue 3/09/21	16	100%	12
0.1	**Task 2: Vehicle Aerodynamic Loads – Simulation**		Wed 10/21/20	Tue 3/09/21	140	100%	100
0.0.1	Fin Design		Wed 10/21/20	Wed 11/04/20	15	100%	11
0.0.1	General Body Ascent Analysis		Thu 11/05/20	Tue 3/09/21	125	100%	89
0.0.0.1	Geometry		Thu 11/05/20	Fri 12/04/20	30	100%	22
0.0.0.1	Mesh		Sat 12/05/20	Fri 2/12/21	70	100%	50
0.0.0.1	Set Up		Sat 2/13/21	Sat 2/27/21	15	100%	10
0.0.0.1	Analysis		Sun 2/28/21	Tue 3/09/21	10	100%	7

7.2 Appendix B: Budget

Item	Quantity	Price ($)	Total Cost ($)
1inch actuators	2	37.99	75.98
3 inch actuators	2	36.99	73.98
Blue Tube	1	10.95	10.95
Battery	1	29.50	29.50
Battery Charger	1	7.95	7.95
Arduino Teensy	1	12.50	12.50
Header Pins	1	26.95	26.95
Threaded Inserts	1	5.95	5.95
TeleMega	1	461.54	461.54
Servo Motors	4	62.99	251.96
Total			1028.61

7.3 Appendix C: Cantera Model Input File

```
# CURRENT MODEL SETUP
# assume AP and Al are already in gaseous phase, not solid phase
# assume initial pressure is atmospheric pressure
# assume initial temperature is the autoignition temperature of AP @513.15
K
# assume no gas phase reactions



units(length = "cm", time = "s", quantity = "mol", act_energy = "J/mol")

ideal_gas(name = "gas",
      elements = """ O H N Cl Al""",
      species = "all",
      reactions = "all",
      # transport = "Multi",
      options=('allow_discontinuous_thermo',
              'skip_undeclared_elements'),
      initial_state = state(temperature = 513.15,
                       pressure = OneAtm,
                       mole_fractions = 'NH4CLO4:0.9,AL:0.1'))


#-----------------------------------------------------------------------
------
# Species data
#-----------------------------------------------------------------------
------


species(name = "NH4CLO4",
  atoms = " N:1 H:4 Cl:1 O:4",
  thermo = (
    NASA9( [ 100.00, 513.15],
           [-3.075344900E+03, -2.136506130E+02, 1.021583093E+01,
            1.659463617E-02, 1.665266832E-05, -2.306096672E-08,
            1.543657693E-11, 3.825767260E+04, -4.230254380E+01] ),
    NASA9( [ 513.15, 1000.00],
           [9.739327490E+06, -7.095035120E+04, 2.133435041E+02,
           -2.657628679E-01, 2.034168863E-04, -5.628229760E-08,
           0.000000000E+00, 3.485366440E+05, -1.304962218E+03] ),
    NASA9( [ 1000.00, 1500.00],
           [4.174684580E+06, -1.716642636E+04, 1.925955165E+01,
           3.499518910E-02, -6.264443390E-06, -1.494354361E-09,
           0.000000000E+00, 7.134752510E+04, -1.274562074E+02] )
      ),
  transport = gas_transport(
          geom = "nonlinear",
          diam =   4.77,
          well_depth =   547,
             dipole =      0.0,
```

```
            polar =   0.0,
            rot_relax =  1.00)
      # note = "Chase,1998 p766."
    )



species(name = "AL",
  atoms = " Al:1 ",
  thermo = (
    NASA9( [ 200.00, 933.61],
            [-6.251811430E+04, 6.343934350E+02, -7.131883820E-01,
            1.088725280E-02, -1.458741820E-05, 9.961160880E-09,
            -1.774928010E-12, -3.985439320E+03, 6.561100200E+00] ),
    NASA9( [ 933.61, 6000.00],
            [ 0.000000000E+00, 0.000000000E+00, 3.818625510E+00,
            0.000000000E+00, 0.000000000E+00, 0.000000000E+00,
            0.000000000E+00, -9.576323160E+01, -1.752553420E+01]),
    NASA9( [ 6000.00, 20000.00],
            [-5.040682320E+08, 3.802322650E+05, -1.082347159E+02,
            1.549444292E-02, -1.070103856E-06, 3.592110900E-11,
            -4.696039394E-16, -2.901050501E+06, 9.491883160E+02])
       ),
  transport = gas_transport(
            geom = "atom",
            diam =   2.6175,
            well_depth =   2995.6,
               dipole =      0.0,
            polar =   0.0,
            rot_relax =  0.00)
  # note = "J 6/83"
    )

species(name = "AL2O3",
  atoms = " Al:2 O:3 ",
  thermo = (
    NASA9( [ 200.000, 500.000],
            [ -5.391549970E+06, 1.036676983E+05, -8.173229150E+02,
              3.388258720E+00, -7.512400360E-03, 8.659248820E-06,
             -4.066085670E-09, -6.660134650E+05, 4.235502230E+03] ),
    NASA9( [ 500.000, 1200.000],
            [ -6.042087868E+05, 0.000000000E+00, 1.475480816E+01,
              8.272285438E-04, 0.000000000E+00, 0.000000000E+00,
              0.000000000E+00, -2.079235447E+05, -8.136029480E+01]),
    NASA9( [ 1200.000, 2327.000],
            [ 0.000000000E+00, 0.000000000E+00, 1.293774378E+01,
             1.992781294E-03, 0.000000000E+00, 0.000000000E+00,
             0.000000000E+00, -2.060787581E+05, -6.966603728E+01]),
    NASA9( [ 2327.000, 6000.000],
            [ 0.000000000E+00, 0.000000000E+00, 1.959225499E+01,
             0.000000000E+00, 0.000000000E+00, 0.000000000E+00,
             0.000000000E+00, -2.027701571E+05, -1.108590952E+02])
       ),
  transport = gas_transport(
```

```
           geom = "nonlinear",
           diam =   2.6175,
           well_depth =   2995.6,
              dipole =      0.0,
           polar =   0.0,
           rot_relax =  0.00)
  # note = "J 6/83"
    )


species(name = "H2O",
  atoms = " H:2 O:1 ",
  thermo = (
    NASA( [ 200.00, 1000.00], [ 4.198640560E+00, -2.036434100E-03,
        6.520402110E-06, -5.487970620E-09,  1.771978170E-12,
        -3.029372670E+04, -8.490322080E-01] ),
    NASA( [ 1000.00, 6000.00], [ 2.677037870E+00,  2.973183290E-03,
        -7.737696900E-07,  9.443366890E-11, -4.269009590E-15,
        -2.988589380E+04,  6.882555710E+00] )
       ),

  transport = gas_transport(
           geom = "nonlinear",
           diam =   2.605,
           well_depth =   572.400,
              dipole =      1.844,
           polar =   0.0,
           rot_relax =  4.00)
  # note = "L 8/89"
    )


species(name = "OH",
  atoms = " O:1 H:1 ",
  thermo = (
    NASA( [ 200.00, 1000.00], [ 3.992015430E+00, -2.401317520E-03,
        4.617938410E-06, -3.881133330E-09,  1.364114700E-12,
        3.615080560E+03, -1.039254580E-01] ),
    NASA( [ 1000.00, 6000.00], [ 2.838646070E+00,  1.107255860E-03,
        -2.939149780E-07,  4.205242470E-11, -2.421690920E-15,
        3.943958520E+03,  5.844526620E+00] )
       ),
  transport = gas_transport(
           geom = "linear",
           diam =   2.750,
           well_depth =   80.000,
              dipole =      0.0,
           polar =   0.0,
           rot_relax =  0.00)
  # note = "TPIS78"
    )

species(name = "N2",
```

```
  atoms = " N:2 ",
  thermo = (
    NASA( [ 200.00, 1000.00], [ 3.531005280E+00, -1.236609870E-04,
        -5.029994370E-07,  2.435306120E-09, -1.408812350E-12,
        -1.046976280E+03,  2.967474680E+00] ),
    NASA( [ 1000.00, 6000.00], [ 2.952576260E+00,  1.396900570E-03,
        -4.926316910E-07,  7.860103670E-11, -4.607553210E-15,
        -9.239486450E+02,  5.871892520E+00] )
       ),
  transport = gas_transport(
           geom = "linear",
           diam =   3.621,
           well_depth =   97.530,
              dipole =      0.0,
           polar =   1.760,
           rot_relax =  4.00)
  # note = "TPIS78"
    )

species(name = "HCL",
  atoms = " H:1 Cl:1 ",
  thermo = (
    NASA( [ 300.00, 1000.00], [ 3.524817100E+00,  2.998486200E-05,
        -8.622189100E-07,  2.097972100E-09, -9.865819100E-13,
        -1.215050900E+04,  2.408923590E+00] ),
    NASA( [ 1000.00, 5000.00], [ 2.766588400E+00,  1.438188300E-03,
        -4.699300000E-07,  7.349940800E-11, -4.373110600E-15,
        -1.191746800E+04,  6.471506290E+00] )
       ),
  transport = gas_transport(
           geom = "linear",
           diam =  3.339,
           well_depth =   344.7,
              dipole =      0.0,
           polar =   0.0,
           rot_relax =  1.00)
  # note = "J 9/64"
    )


species(name = "CL",
  atoms = " Cl:1 ",
  thermo = (
    NASA( [ 200.00, 1000.00], [ 2.260624800E+00,  1.541543990E-03,
        -6.802836220E-07, -1.599729750E-09,  1.154166360E-12,
        1.385529860E+04,  6.570207990E+00] ),
    NASA( [ 1000.00, 6000.00], [ 2.946583580E+00, -3.859854080E-04,
        1.361393880E-07, -2.170329230E-11,  1.287510250E-15,
        1.369703270E+04,  3.113301360E+00] )
       ),
  transport = gas_transport(
           geom = "atom",
           diam =  3.613,
           well_depth =   130.8,
```

```
             dipole =      0.0,
          polar =   0.0,
          rot_relax =  1.00)
  # note = "J 6/82"
    )

species(name = "O2",
  atoms = " O:2 ",
  thermo = (
    NASA( [ 200.00, 1000.00], [ 3.782456360E+00, -2.996734150E-03,
        9.847302000E-06, -9.681295080E-09,  3.243728360E-12,
        -1.063943560E+03,  3.657675730E+00] ),
    NASA( [ 1000.00, 6000.00], [ 3.660960830E+00,  6.563655230E-04,
        -1.411494850E-07,  2.057976580E-11, -1.299132480E-15,
        -1.215977250E+03,  3.415361840E+00] )
       ),
  transport = gas_transport(
          geom = "linear",
          diam =  3.458,
          well_depth =   107.400,
             dipole =      0.0,
          polar =   1.600,
          rot_relax =  3.80)
  # note = "TPIS89"
    )


#-----------------------------------------------------------------------
------
# Reaction data
#-----------------------------------------------------------------------
------
```

7.4 Appendix D: Cantera Model MATLAB Code

```
gas = Solution('prop1 transport.cti')
```

```
  gas:

       temperature          513.15  K
          pressure          101325  Pa
           density         2.57525  kg/m^3
   mean mol. weight         108.438  amu

                          1 kg            1 kmol
                       -----------     ------------
           enthalpy    -2.1024e+06      -2.28e+08     J
    internal energy    -2.1417e+06     -2.322e+08     J
            entropy         2414.6      2.618e+05     J/K
     Gibbs function    -3.3414e+06     -3.623e+08     J
   heat capacity c_p         1518.8      1.647e+05     J/K
   heat capacity c_v         1442.2      1.564e+05     J/K

                           X               Y          Chem. Pot. / RT
                      -------------   ------------    ------------
```

```
% gas = Solution('prop2.cti')


AL = speciesIndex(gas,'AL');
AP = speciesIndex(gas,'NH4CLO4');

mole_fraction = moleFractions(gas);

AL_mol = mole_fraction(AL,1)
```

```
AL_mol = 0.1000
```

```
AP_mol = mole_fraction(AP,1)
```

```
AP_mol = 0.9000
```

```
% Option 1 Define major species mass fraction as > 5% and change nothing
% Option 2 Define major species mass fraction as > 1% and add O and NO
% species to products

set(gas, 'T', 558.15, 'P', 101325, 'X', 'NH4CLO4:0.9,AL:0.1')
% set(gas, 'T', 558.15, 'P', 101325, 'X', 'NH4CLO4:1.0')
mf_init = moleFractions(gas)
```

```
mf_init = 9×1
    0.9000
    0.1000
         0
         0
         0
         0
         0
         0
         0
```

```
equilibrate(gas, 'UV')
```

```
  gas:

       temperature         2460.39  K
          pressure     1.71393e+06  Pa
           density         2.36763  kg/m^3
   mean mol. weight         28.2591  amu

                          1 kg           1 kmol
                       -----------     ------------
           enthalpy    -1.3512e+06      -3.819e+07     J
    internal energy    -2.0752e+06      -5.864e+07     J
           entropy         9113.5       2.575e+05     J/K
     Gibbs function    -2.3774e+07      -6.718e+08     J
  heat capacity c_p         1587.7       4.487e+04     J/K
  heat capacity c_v         1293.5       3.655e+04     J/K

                          X              Y          Chem. Pot. / RT
                     -------------  ------------  ------------
                  AL  3.02477e-12   2.88802e-12       -32.2069
                AL2O3    0.0130301     0.0470136       -106.452
                  H2O     0.358632       0.22863       -38.4384
                   OH    0.0126105    0.00758946       -26.2256
                   N2     0.117271       0.116251       -26.9744
```

```
% mu = viscosity(gas) % Pa-s
% kT = thermalConductivity(gas)

O2 = speciesIndex(gas,'O2');
H2O = speciesIndex(gas,'H2O');
N2 = speciesIndex(gas,'N2');
HCL = speciesIndex(gas,'HCL');
CL = speciesIndex(gas, 'CL');
AL = speciesIndex(gas,'AL');
AL2O3 = speciesIndex(gas,'AL2O3');
AP = speciesIndex(gas,'NH4CLO4');
OH = speciesIndex(gas, 'OH');

mole_fraction = moleFractions(gas);

O2_mol = mole_fraction(O2,1);
H2O_mol = mole_fraction(H2O,1);
OH_mol = mole_fraction(OH,1);
N2_mol = mole_fraction(N2,1);
HCL_mol = mole_fraction(HCL,1);
CL_mol = mole_fraction(CL,1);
AL2O3_mol = mole_fraction(AL2O3,1);
AL_mol = mole_fraction(AL,1);
AP_mol = mole_fraction(AP,1);

Z_mole = [O2_mol;H2O_mol;OH_mol;N2_mol;HCL_mol;CL_mol;AL2O3_mol];

T_equil = temperature(gas)
```

```
T_equil = 2.4604e+03
```

```
P_chamb = pressure(gas)
```

```
P_chamb = 1.7139e+06
```

```
rho_chamb = density(gas)
```

```
rho_chamb = 2.3676
```

```
Cp = cp_mass(gas)
```

```
Cp = 1.5877e+03
```

```
Cv = cv_mass(gas)
```

```
Cv = 1.2935e+03
```

```
Mm = meanMolecularWeight(gas)
```

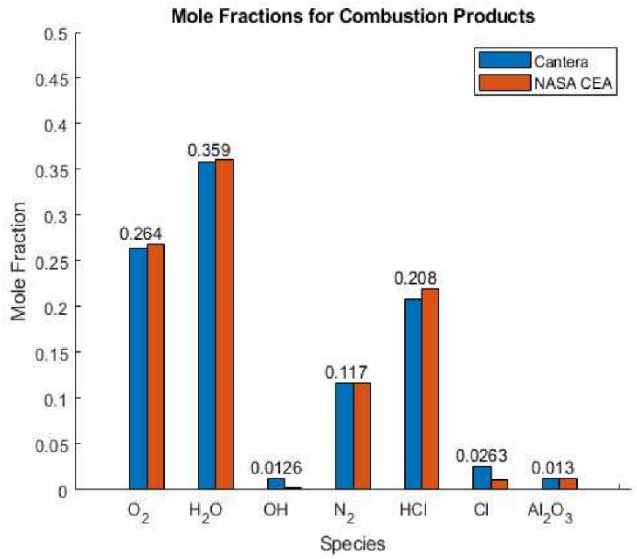
```
Mm = 28.2591
```

```
k = Cp/Cv
```

```
k = 1.2275
```

```
cea_res = [0.26894; 0.36102; 0.00282; 0.11631; 0.22025; 0.01044; 0.01309];
hold on
bar((1:length(Z_mole)),Z_mole, 0.25)
bar((1:length(cea_res))+0.25, cea_res, 0.25)
xlabel('Species');
ylabel('Mole Fraction');
title('Mole Fractions for Combustion Products')
ylim([0 0.5])

name = {'O_2';'H_2O';'OH';'N_2';'HCl';'Cl';'Al_2O_3'; ''};
set(gca,'xticklabel',name)
legend('Cantera', 'NASA CEA')
text(1:7, [cea_res(1:2); Z_mole(3:4); cea_res(5); Z_mole(6:7)], num2str(Z_mole(1:7),3),'vert','bottom','horiz','center');
```




```
R = 8314.5; %Gas Constant (J/kg-K)
g = 9.81;
P_a = 101325;
Ae_At = 1.25^2 / .455^2;
exit_radius = 0.03175/2;
A_e = pi*exit_radius^2;

[M_exit, T, P, rho, area] = flowisentropic(k, Ae_At, 'sup');

T_e = T*T_equil %Exit Temperature (K)
```

```
T_e = 1.1567e+03
```

```
P_e = P*P_chamb %Exit Pressure (Pa)
```

```
P_e = 2.9190e+04
```

```
rho_e = rho*rho_chamb %Exit density (kg/m^3)
```

```
rho_e = 0.0858
```

```
u_e = sqrt((2*k/(k-1))*(R/Mm)*T_equil*(1-P)^((k-1)/k)) %Exit Velocity (m/s)
```

```
u_e = 2.7907e+03
```

```
mdot_cantera = rho_e*u_e*A_e %Mass flow rate (kg/s)
```

```
mdot_cantera = 0.1895
```

```
Thrust = mdot_cantera*u_e + (P_e - P_a)*A_e %Thrust (N)
```

```
Thrust = 471.7305
```

```
Isp = Thrust/(mdot_cantera*g) % Specific Impulse (s)
```

```
Isp = 253.7528
```

```
Tact = 405.151;
Ispact = 184.4;

Terr = 100*(Tact - Thrust)/Tact
```

```
Terr = -16.4333
```

```
Isperr = 100*(Ispact - Isp)/Ispact
```

```
Isperr = -37.6100
mdot_comsol = 0.1795
mdot_err = 5.2782
```

7.5 Appendix E: Equations in COMSOL Model

Condition	Equation	Variables
Heat Transfer in Solids	$\rho C_p \frac{\partial T}{\partial t} + \rho C_p \boldsymbol{u} \cdot \nabla T + \nabla \cdot \boldsymbol{q} = Q + Q_{ted}$ $\boldsymbol{q} = -k\nabla T$	ρ – solid density (kg/m^3) C_p – solid heat capacity at constant pressure (J/kg-K) k – solid thermal conductivity (W/m-K) **u** – velocity field (m/s) Q – heat source (W/m^3) Q_{ted} – thermoelastic damping (W/m^3)
Heat Transfer in Fluids	$\rho C_p \frac{\partial T}{\partial t} + \rho C_p \boldsymbol{u} \cdot \nabla T + \nabla \cdot \boldsymbol{q} = Q + Q_p + Q_{vd}$ $\boldsymbol{q} = -k\nabla T$	Q_p – pressure work (W/m^3) Q_{vd} – viscous dissipation (W/m^3) See 'Heat Transfer in Solids' for other variable definitions
Symmetry & Thermal Insulation Boundary	$-\boldsymbol{n} \cdot \boldsymbol{q} = 0$	q – heat flux (W/m^2) n – normal vector
Convective Heat Flux Boundary	$-\boldsymbol{n} \cdot \boldsymbol{q} = q_0$ $q_0 = h \cdot (T_{ext} - T)$	q – heat flux (W/m^2) n – normal vector h – heat transfer coefficient (W/m^2-K) T_{ext} – external temperature (K) T – domain temperature (K)
Temperature Boundary	$T = T_0$	T_0 – boundary temperature (K)
Weakly Compressible Laminar Flow	$\rho \frac{\partial \boldsymbol{u}}{\partial t} + \rho(\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = \nabla \cdot [-p\boldsymbol{I} + \boldsymbol{K}] + \boldsymbol{F}$ $\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0$ $\boldsymbol{K} = \mu(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T) - \frac{2}{3}\mu(\nabla \cdot \boldsymbol{u})\boldsymbol{I}$	**I** – Identity matrix **u** – velocity vector (m/s) ρ – solid density (kg/m^3) p – pressure (Pa) **F –** external force (N) μ – dynamic viscosity (Pa-s)
Wall Boundary (no slip)	$\boldsymbol{u} = 0$	**u** – fluid velocity (m/s)

Inlet Boundary	$\boldsymbol{u} = \boldsymbol{u_0}$	$\mathbf{u_0}$ – initial velocity field (m/s)
Outlet Pressure Boundary	$[-p\boldsymbol{I} + \boldsymbol{K}]\boldsymbol{n} = -\hat{p}_0\boldsymbol{n}$ $\boldsymbol{K} = \mu(\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^T) - \frac{2}{3}\mu(\nabla \cdot \boldsymbol{u})\boldsymbol{I}$ $\hat{p}_0 \leq p_0$	$\mathbf{I}$ – Identity matrix $\mathbf{u}$ – velocity vector (m/s) p_0 – absolute pressure (Pa) μ – dynamic viscosity (Pa-s) $\mathbf{n}$ – normal vector

7.6 Appendix F: Dynamical Simulator Code

7.6.1 Aerodynamics

```
classdef Aerodynamics
  %AERODYNAMICS Aerodynamics functions class.

  properties
    aoa;  % Angle of attack values from CFD.
    C_d;  % Coefficient of Drag polyfit parameters.
    C_l;  % Coefficient of Lift polyfit parameters.
    C_m;  % Coefficient of Moment polyfit parameters.
  end

  methods
    function obj = Aerodynamics()
      %AERODYNAMICS Instantiates the object.
      load("CFD.mat", "aoa", "C_d", "C_l", "C_mx", "C_my", "C_mz");

      obj.aoa = aoa;
      obj.C_d = C_d;
      obj.C_l = C_l;
      obj.C_m = [C_mx C_my C_mz];
    end

    function F_a = Forces(obj, rok, env, Q, V, time)
      %FORCES   Comptues aerodynamic forces on the vehicle.
      %  rok:  Rocket object.
      %  env:  Environment object.
      %  Q:   Attiude of the vehicle as a quaternion.
      %  V:   Velocity in the inertial frame in m/s.
      %  time:  Simulation time in seconds.
      %  RETURNS: Aerodynamic force in inertial frame.
      if all(0 == V)
        vel_dir = zeros(3,1);
      else
        vel_dir = V / norm(V);
      end
      % Force in body frame
      alpha = Aerodynamics.AngleOfAttack(Q, V);
      Cd = Aerodynamics.Cubic(alpha, obj.C_d);
      f_a = - Cd * obj.Area(rok, obj.AngleOfAttack(Q, V))...
        * env.rho * norm(V)^2 / 2;
      F_a = f_a * vel_dir;
    end
```

```
  function M_net = Moments(obj, rok, env, Q, V, time)
    %FORCES   Comptues aerodynamic moments on the vehicle.
    %  rok:  Rocket object.
    %  env:  Environment object.
    %  Q:   Attiude of the vehicle as a quaternion.
    %  V:   Velocity in the inertial frame in m/s.
    %  time:  Simulation time in seconds.
    %  RETURNS: Aerodynamic moment in inertial frame.
    if all(0 == V)
      M_net = [0; 0; 0];
    else
      uq = QuaternionRotate(Q, [1 0 0])';
      uv = V / norm(V);

      a_v = asin(uv(3)/norm(uv));
      a_q = asin(uq(3)/norm(uq));

      alpha = -(a_v - a_q);

      b_v = atan(uv(2)/uv(1));
      b_q = atan(uq(2)/uq(1));
      if isnan(b_v)
        b_v = b_q;
      end
      beta = b_v - b_q;

      attack = obj.AngleOfAttack(Q, V);
      CoefM = Aerodynamics.Cubic(attack, obj.C_m(:,3));
      t = Aerodynamics.CoefToMoment(env, rok, V, attack, CoefM);
      M_net = [0; t*alpha; t*beta];
      Qinv = QuaternionInvert(Q);
      M_net = QuaternionRotate(Qinv, M_net');
    end
  end
end

methods(Static)
  function A = Area(rok, aoa)
    %AREA Estimates the crossectional area of a vehicle given its
    %angle of attack. This is done by assuming the crossection is
    %a rectangle that widens as aoa increases.
    h = rok.d + rok.len * sin(aoa);
    A = h * rok.d;
  end

  function aoa = AngleOfAttack(Q, V)
    %ANGLEOFATTACK Computes the angle of attack given attitude as a
    %quaternion and velocity.
    uq = QuaternionRotate(Q, [1 0 0])';
    uv = V / norm(V);
    aoa = acos(dot(uq, uv)/vecnorm(uq));
    if ~isreal(aoa) || isnan(aoa)
      aoa = 0;
    end
  end

  function Q = DynamicPressure(env, V)
    %DYNAMICPRESSURE Computes the dynamic pressure given an
    %environment and velocity.
    Q = env.rho * vecnorm(V)^2 / 2;
```

```
    end

    function M = CoefToMoment(env, rok, V, aoa, C_m)
      %COEFTOMOMENT Converts C_m, a moment coefficient, to a real
      %moment given the current environment, the rocket, the
      %velocity, and the angle of attack.
      M = C_m * Aerodynamics.DynamicPressure(env, V) * ...
        Aerodynamics.Area(rok, aoa) * rok.len;
    end

    function y = Cubic(x, p)
      % CUBIC Computes a cubic function at x for coefficient vector p
      % Designed to work with polyfit()
      y = p(1)*x^3 + p(2)*x^2 + p(3)*x + p(4);
    end
  end
end
```

Published with MATLAB® R2020b

7.6.2 DefinedMoments

```
classdef DefinedMoments < Aerodynamics
  %DEFINEDMOMENTS A subclass of Aerodynamics that applies manually
  %  deffined moments rather than aerodynamic ones.
  %  Calculates moments due to gravity from a specified COM.

  properties
    moments;
    COM;
  end

  methods
    function obj = DefinedMoments(Moments, COM)
      % Defines an instance of the class.
      %  Moments: An array of 5 long row vectors. Elements 1 and
      %  two are the start and endtimes while elements 3 through 5
      %  are the components.
      %  COM: Location of the center of mass relative to the origin.
      %  If non zero, gravity exerts a moment.
      obj.moments = Moments;
      obj.COM = COM;
    end

    function F_a = Forces(obj, rok, env, Q, V)
      % Exerts no forces.
      F_a = zeros(3,1); % No aerodynamic forces
    end

    function M_net = Moments(obj, rok, env, Q, V, time)
      % Applies the defined moments on the vehicle.
      M_net = zeros(3,1);
      [nummoments,~] = size(obj.moments);
      for n = 1:nummoments
        if time > obj.moments(n,1) && time < obj.moments(n,2)
          m = obj.moments(n,3:5);
          M_net = M_net + m';
        end
      end
```

```
      F_g = QuaternionRotate(QuaternionInvert(Q),...
        [0 0 -env.g*rok.m_s]);
      M_g = cross(obj.COM, F_g);
      M_net = M_net + M_g';
    end
  end
end
```

Published with MATLAB® R2020b

7.6.3 Environment

```
classdef Environment
  %ENVIRONMENT Container class for environmental properties.

  properties
    g = 9.80665;      % m/s^2 Standard Gravitational Acceleration (per ISO 80000)
    rho = 1.225;      % kg/m^3
    V_w;          % m/s  Wind velocity vector
  end

  methods
    function obj = Environment(V_w, rho, g)
      %ENVIRONMENT Construct an instance of this class
      obj.V_w = V_w;
      if nargin > 1
        obj.rho = rho;
      end
      if nargin > 2
        obj.g = g;
      end
    end
  end
end
```

Published with MATLAB® R2020b

7.6.4 Rocket

```
classdef Rocket
  % ROCKET Represents a rocket.
  %  Used to hold vehicle parameters and methods.

  properties
    Isp;    % s, Specific impulse
    m_pi;    % kg, Initial propellant mass
    m_s;    % kg, Structural mass
    dm;     % kg, Mass flow rate over time
    tb;     % s, Motor burn time.
    c_mp;    % m, COM of propellant along x axis
    c_ms;    % m, COM of structure along x axis
%     c_p;    % m, COP of airframe without fins along x axis
%     c_pfx;   % m, COP of fins along x axis
%     c_pfr;   % m, COP of fins along radial
    d;     % m, Airframe diameter
    len;    % m, Airframe length
  end
```

```
  methods
    function [Ix, Iy, Iz] = MomentOfInertia(obj)
      %MOMENTOFINTERTIA Returns the moments of inertia about the
      %         primary axis of the rocket.
      Ix = 3;
      Iy = 15;
      Iz = 15;
    end

    function [dmass] = Control(obj, state, t)
      %CONTROL  Computes control variables
      %  state: 14 long current state vector
      %  t:   Current time in seconds
      if t < obj.tb && state(14) > 0
        index = floor(t / (obj.tb/length(obj.dm))) + 1;
        dmass = obj.dm(index);
      else
        dmass = 0;
      end
    end

    function F_T = Thrust(obj, Q, dm, env)
      %THRUST   Computes the thrust of the vehicle in the inertial
      %      frame.
      %  Q: Attitude of the vehicle as a quaternion.
      %  dm:     Rate of propellant mass loss in kg/s.
      %  env:Environment object.
      attitude = QuaternionRotate(Q, [1 0 0])';
      T = -dm * obj.Isp * env.g; % Force in body frame
      F_T = T * attitude;     % Force in inertial frame
    end
  end

  methods(Static)
    function Ip = MomentOfInertiaPointMass(m, p)
      %MOMENTOFINERTIAPOINTMASS  Computes the moment of inertia of
      %              a point mass.
      %  m: Mass of the point mass in kg.
      %  p: Position vector relative to point of rotation.
      Ip = [
        m*(p(2)^2+p(3)^2)  -m*p(1)*p(2)    -m*p(1)*p(3)
        -m*p(1)*p(2)    m*(p(1)^2+p(3)^2)  -m*p(2)*p(3)
        -m*p(1)*p(3)    -m*p(2)*p(3)    m*(p(1)^2+p(2)^2)
        ];
    end
  end
end
```

Published with MATLAB® R2020b

7.6.5 QuaternionRotate

```
function v_prime = QuaternionRotate(q, v)
%ROTATEFRAME Applies a quaternion rotation to a row vector
%    Uses quaternion form w + i + j + k -> [w i j k]
qinv = QuaternionInvert(q);
v_prime = QuaternionComposition(QuaternionComposition(q,[0 v]), qinv);
v_prime = v_prime(2:4);
```

```
end
```

Published with MATLAB® R2020b

7.6.6 QuaternionInvert

```
function qinv = QuaternionInvert(q)
%QUATERNIONINVERT Inverts a quaternion
%  q: Quaternion to invert.
qinv = [q(1) -q(2:4)];
end
```

Published with MATLAB® R2020b

7.6.7 QuaternionFromVector

```
function q = QuaternionFromVector(vec)
%QuaternionFromVector Computes a quaternion representing a rotation from the
%  vector [1 0 0] to a given vector.
%  vec: Vector to solve for.

% Basis Vectors
i_ = [1 0 0];
j_ = [0 1 0];
k_ = [0 0 1];

% Yaw and Pitch
vec = vec/norm(vec);
theta = acos(vec(1)/(sqrt(vec(2)^2+vec(1)^2))); % Rotation about z axis
phi = atan(vec(3)/sqrt(vec(2)^2 + vec(1)^2)); % Rotation about y axis

if vec(2) > 0
  theta = -theta;
end

% Rotation Matrix
R = [
  cos(theta)  sin(theta)  0
  -sin(theta)  cos(theta)  0
  0  0  1
] * [
  cos(phi)  0  -sin(phi)
  0      1  0
  sin(phi)  0  cos(phi)
];

% Compute Direction Cosine Matrix
I_ = R * i_';
J_ = R * j_';
K_ = R * k_';

Q = [
  I_'
  J_'
  K_'
];

% Convert DCM to Quaternion
K3 = ...
```

```
[Q(1,1)-Q(2,2)-Q(3,3), Q(2,1)+Q(1,2), Q(3,1)+Q(1,3), Q(2,3)-Q(3,2);
 Q(2,1)+Q(1,2), Q(2,2)-Q(1,1)-Q(3,3), Q(3,2)+Q(2,3), Q(3,1)-Q(1,3);
 Q(3,1)+Q(1,3), Q(3,2)+Q(2,3), Q(3,3)-Q(1,1)-Q(2,2), Q(1,2)-Q(2,1);
 Q(2,3)-Q(3,2), Q(3,1)-Q(1,3), Q(1,2)-Q(2,1), Q(1,1)+Q(2,2)+Q(3,3)]/3;

[eigvec, eigval] = eig(K3);

[~,index] = max(diag(eigval));

q = eigvec(:,index);
q = [q(4) q(1:3)'];
end
```

Published with MATLAB® R2020b

7.6.8 QuaternionComposition

```
function res = QuaternionComposition(p, q)
%QUATERNIONCOMPOSITION Computes the composition p x q
%       Note, this is a snynonym for multiplication.

res(1) = q(1)*p(1)-dot(q(2:4), p(2:4));
res(2:4) = q(1)*p(2:4) + p(1)*q(2:4) + cross(p(2:4), q(2:4));

end
```

Published with MATLAB® R2020b

7.6.9 Import CFD

```
% Imports data from CFD.xlsx as CFD.mat to be used by the aerodynamics
% solver.
clc; clear;
table = open("CFD.xlsx");
aoa = table.data(:,1);
C_l = polyfit(aoa, table.data(:,2), 3);
C_d = polyfit(aoa, table.data(:,3), 3);
C_mx = polyfit(aoa, table.data(:,4), 3);
C_my = polyfit(aoa, table.data(:,5), 3);
C_mz = polyfit(aoa, table.data(:,6), 3);
mx = table.data(:,7);
my = table.data(:,8);
mz = table.data(:,9);
l = table.data(:,10);
d = table.data(:,11);
save("CFD.mat", "aoa", "C_l", "C_d", "C_mx", "C_my", "C_mz", ...
  "mx", "my", "mz", "l", "d");
```

Published with MATLAB® R2020b

7.6.10 BuildAndromeda

```
function mqpRocket = BuildAndromeda()
%BUILDANDROMEDA Builds an instance of Andromeda for the simulator.
mqpRocket = Rocket;
mqpRocket.Isp = 184;
mqpRocket.m_pi = .708;
```

```
mqpRocket.m_s = 6.9691;
mqpRocket.dm = [2.08];
mqpRocket.tb = 3.4;
mqpRocket.c_mp = .1;
mqpRocket.c_ms = .625;
% mqpRocket.c_p = .3;
% mqpRocket.c_pfx = .15;
% mqpRocket.c_pfr = .12;
mqpRocket.d = 0.1016;
mqpRocket.len = 1.757;
end
```

Published with MATLAB® R2020b

7.6.11 Simulator

```
classdef Simulator
  %SIMULATOR Primary simulator object

  properties
    initialState double;          % Initial State
    rok (1,1);                % Current Rocket
    aero (1,1);               % Aerodynamics
    env (1,1);                % Environment
    ignoreForces;              % Do not solve COM equations
    ignoreMoments;             % Do not solve Euler equations
    gusts;                 % Wind gusts
    endtime = 500;             % Maximum Simulation Length
    gustForce = false;           % Apply gust as force
                        %   (rather than Velocity)
  end

  methods
    function obj = Simulator(initialState, ascentRocket, ...
        aerodynamics, environment, ignoreForces, ...
        iqnoreMoments, endtime, gusts, gustForce)
      % SIMULATOR Construct an instance of this class
      %  initialState: A 14 long column vector of initial
      %    conditions.
      %  ascentRocket: A rocket object for ascent.
      %  aerodynamics: An aerodynamic solver object.
      %  environemt: An environment object.
      %  ignoreForces: If true, does not solve for COM location.
      %  ignoreMoments: If true, does not solve euler equations.
      %  endtime: The maximum simulated flight time. [optional]
      %  gusts: Defines wind gusts. Passed as an array of 5 long
      %      row vectors. Elements 1 and 2 are start and end
      %      times while elements 3 through 5 are the cartesian
      %      components in m/s. [optional]
      %  gustForce: If true, applies gusts as a direct force in N
      %       rather than a difference in airspeed. [optional]
      obj.initialState = initialState;
      obj.rok = ascentRocket;
      obj.aero = aerodynamics;
      obj.env = environment;
      obj.ignoreForces = ignoreForces;
      obj.ignoreMoments = iqnoreMoments;
      if nargin > 6
        obj.endtime = endtime;
      end
      if nargin > 7
```

```
      obj.gusts = gusts;
    end
    if nargin > 8
      obj.gustForce = gustForce;
    end
  end

  function [time, state] = Execute(obj, initialState)
    %EXECUTE Runs the simulation.
    %  initialState: The 14 long initial state vector.
    %  STATE VECTOR
    %  1:3 -> p
    %  4:7 -> Q
    %  8:10 -> V
    %  11:13 -> w
    %  14 -> m_p
    Opt = odeset('Events', @obj.landing);
    [time, state] = ode45(@(t, y) obj.StateDerivative(t, y), ...
      [0 obj.endtime], initialState, Opt);
  end

  function dstate = StateDerivative(obj, time, state)
    %STATEDERIVATIVE Derivative of the state vector that is passed
    %        to ode45() to solve the problem.
    %  time:  Simulation time in seconds
    %  state: Current 14 long state vector.
    V = state(8:10) + obj.env.V_w;

    if obj.ignoreMoments
      if ~any(any(isnan(V))) && ~any(any(V == 0))
        Q = QuaternionFromVector(V);
      else
      Q = state(4:7)';
      end
    else
      Q = state(4:7)';
    end
    dmass = obj.rok.Control(state, time);

    dmass = -dmass;

    [numgusts,~] = size(obj.gusts);

    F_W = zeros(3,1);

    for n = 1:numgusts
      if time >= obj.gusts(n,1) && time <= obj.gusts(n,2)
        if ~obj.gustForce
          V = V + obj.gusts(n,3:5)';
        else
          F_W = F_W + obj.gusts(n,3:5)';
        end
      else
        F_W = F_W + zeros(3,1);
      end
    end
    if obj.ignoreForces
      dP = zeros(3,1);
      dV = zeros(3,1);
    else
      F_A = obj.aero.Forces(obj.rok, obj.env, Q, V);
```

```
        F_G = [0; 0; -(obj.env.g*(state(end) + obj.rok.m_s))];
        F_T = obj.rok.Thrust(Q, dmass, obj.env);

        F_net = F_A + F_G + F_T + F_W;

        dP = state(8:10);
        dV = F_net / (state(end) + obj.rok.m_s);
      end
      if obj.ignoreMoments
        dQ = zeros(4,1);
        dw = zeros(3,1);
      else
        w = state(11:13)';

        dQ = QuaternionComposition(Q, [0 w]/2)';
        [Ix, Iy, Iz] = obj.rok.MomentOfInertia();

        M = obj.aero.Moments(obj.rok, obj.env, Q, V, time);

        dw = [
          M(1)/Ix-(Iz-Iy)*w(2)*w(3)/Ix
          M(2)/Iy-(Ix-Iz)*w(1)*w(3)/Iy
          M(3)/Iz-(Iy-Ix)*w(2)*w(1)/Iz
        ];
      end
      dstate = [

        dP

        dQ

        dV

        dw

        dmass

      ];

    end
  end

  methods (Static)

    function [value, isterminal, direction] = landing(T, Y)
      %LANDING  Determines the end of the simulation
      %      Triggers on hitting ground.
      value    = (Y(3) < 0);
      isterminal = 1;  % Stop the integration
      direction = 0;
    end
  end
end
```

Published with MATLAB® R2020b

7.6.12 COMPlots

```
function COMPlots(state, time)
%EULERPLOTS Plots data from the COM simulator.
%  state: n by 14 matrix of state over time.
%  time: n long vector of time stamps.
  figure(2)
  subplot(311)
  plot(time, state(:,1));
  title("Position over Time");
  xlabel("Time [s]");
  ylabel("X position [m]");
  subplot(312)
  plot(time, state(:,2));
  xlabel("Time [s]");
  ylabel("Y position [m]");
  subplot(313)
  plot(time, state(:,3));
  xlabel("Time [s]");
  ylabel("Z position [m]");

        figure(4)
  subplot(411)
  plot(time, state(:,8));
  title("Velocity over Time");
  xlabel("Time [s]");
  ylabel("X [m/s]");
  subplot(412)
  plot(time, state(:,9));
  xlabel("Time [s]");
  ylabel("Y [m/s]");
  subplot(413)
  plot(time, state(:,10));
  xlabel("Time [s]");
  ylabel("Z [m/s]");
  subplot(414)
  plot(time, vecnorm(state(:,8:10)'));
  xlabel("Time [s]");
  ylabel("Total [m/s]");
  figure(6)
  plot(time, state(:,14));
  title("Propellant Mass over Time");
  xlabel("Time [s]");
  ylabel("Mass [kg]");

  figure(1)
  hold on
  plot3(state(:,1),state(:,2), state(:,3),"b");
  plot3([(min(state(:,1))-max(state(:,1))  *  .3)  (max(state(:,1))+max(state(:,1))  *
.3)],[0 0],[0 0], 'k--');
  plot3([0 0],[(min(state(:,2))-max(state(:,2)) * .3) (max(state(:,2))+max(state(:,2))
* .3)],[0 0], 'k--');
  plot3([min(state(:,1))  max(state(:,1))],[min(state(:,2))  max(state(:,2))],[0  0],
'b:');
  title("Flight Path");
  daspect([1 1 1]);
  xlim([(min(state(:,1))-max(state(:,1))  *  .3)  (max(state(:,1))+max(state(:,1))  *
.3)]);
  ylim([(min(state(:,2))-max(state(:,2))  *  .3)  (max(state(:,2))+max(state(:,2))  *
.3)]);
  zlim([0 (max(state(:,3))+max(state(:,3)) * .1)]);
  hold off
```

```
end
```

Published with MATLAB® R2020b

7.6.13 EulerPlots

```
function EulerPlots(state,time)
%EULERPLOTS Plots data from the Euler simulator.
%  state: n by 14 matrix of state over time.
%  time: n long vector of time stamps.
  pitchyawroll = quat2eul(quaternion(state(:,4:7)), "ZYX");
  pitchyawroll = rad2deg(pitchyawroll)';
  figure(3)
  subplot(311)
  plot(time, pitchyawroll(1,:));
  title("Attitude over Time");
  xlabel("Time [s]");
  ylabel("Z [deg]");
  subplot(312)
  plot(time, pitchyawroll(2,:));
  xlabel("Time [s]");
  ylabel("Y [deg]");
  subplot(313)
  plot(time, pitchyawroll(3,:));
  xlabel("Time [s]");
  ylabel("X [deg]");

  figure(5)
  subplot(411)
  plot(time, rad2deg(state(:,11)));
  title("Rotational Velocity over Time");
  xlabel("Time [s]");
  ylabel("X [deg/s]");
  subplot(412)
  plot(time, rad2deg(state(:,12)));
  xlabel("Time [s]");
  ylabel("Y [deg/s]");
  subplot(413)
  plot(time, rad2deg(state(:,13)));
  xlabel("Time [s]");
  ylabel("Z [deg/s]");
  subplot(414)
  plot(time, vecnorm(rad2deg(state(:,11:13))'));
  xlabel("Time [s]");
  ylabel("Total [deg/s]");

  i_ = [1 0 0];
  j_ = [0 1 0];
  k_ = [0 0 1];
  figure(12);
  fprintf("Press enter to continue...\n");
  pause;
  for n = 2:length(time)
    hold off;
    plot3([0 i_(1)],[0 i_(2)], [0 i_(3)], "r");
    hold on;
    plot3([0 j_(1)],[0 j_(2)], [0 j_(3)], "r");
    plot3([0 k_(1)],[0 k_(2)], [0 k_(3)], "r");
    xlim([-1 1]);
    ylim([-1 1]);
    zlim([-1 1]);
```

```
    I_ = QuaternionRotate(state(n,4:7), i_);
    J_ = QuaternionRotate(state(n,4:7), j_)/2;
    K_ = QuaternionRotate(state(n,4:7), k_)/2;

    v = .75 * state(n,8:10)/vecnorm(state(n,8:10));
    plot3([0 v(1)],[0 v(2)], [0 v(3)], "m");

    plot3([0 I_(1)],[0 I_(2)], [0 I_(3)], "b--");
    plot3([0 J_(1)],[0 J_(2)], [0 J_(3)], "b--");
    plot3([0 K_(1)],[0 K_(2)], [0 K_(3)], "b--");
    text(.75,0,-.3, sprintf("%0.1f sec", time(n)));

    text(i_(1), i_(2), i_(3), "   x");
    text(I_(1), I_(2), I_(3), "   x'");
    text(j_(1), j_(2), j_(3), "   y");
    text(J_(1), J_(2), J_(3), "   y'");
    text(k_(1), k_(2), k_(3), "   z");
    text(K_(1), K_(2), K_(3), "   z'");
    text(v(1), v(2), v(3), "   V");
    title("Vehicle Attitude");
    pause(time(n)-time(n-1));
  end
end
```

Published with MATLAB® R2020b

7.6.14 Case 1

```
%%CASE 1: Singular Forces [COM]
%Simulates a point mass moving in space. It is acted on by two arbitrary forces. The
%mass has the properties of the vehicle, but does not have propellant and is not %affected
by gravity.
% Initialize Rocket
clc; clear global; clear; close all; clf;
ascentRocket = BuildAndromeda();


ascentRocket.m_pi = 0;


environment = Environment(zeros(3,1));


environment.g = 0;


moments = [


  0 0 0 0 0


];


aerodynamics = DefinedMoments(moments, [0 0 0]);


initialState = [


  zeros(3,1)              % Launch Position
  compact(quaternion([15 -85 0], ... % Launch Attitude
      "eulerd", "ZYX", "frame"))'
  zeros(3,1)              % Initial Velocity
```

```
  zeros(3,1)              % Initial Rotational Vel
  ascentRocket.m_pi           % Initial Propellant Mass
];
simulator = Simulator(initialState, ascentRocket, aerodynamics, ...
  environment, false, true, 10, [1 2 2 2 3; 3 4 0 0 -3], true);
tic

[time, state] = simulator.Execute(initialState);

toc

if ~simulator.ignoreForces
  COMPlots(state, time);
end

if ~simulator.ignoreMoments
  EulerPlots(state, time);
end
```

Published with MATLAB® R2020b

7.6.15 Case 2

```
%CASE 2: Simple Flight [COM]
%Simulates the vehicle as a point mass as it flies along a typical model rocket flight.
%clc; clear global; clear; close all; clf;
ascentRocket = BuildAndromeda();

environment = Environment(zeros(3,1));

aerodynamics = Aerodynamics();

initialState = [

  zeros(3,1)              % Launch Position
  compact(quaternion([15 -88 0], ... % Launch Attitude
      "eulerd", "ZYX", "frame"))'
  zeros(3,1)              % Initial Velocity
  zeros(3,1)              % Initial Rotational Vel
  ascentRocket.m_pi           % Initial Propellant Mass
];
simulator = Simulator(initialState, ascentRocket, ...
  aerodynamics, environment, false, true);
tic

[time, state] = simulator.Execute(initialState);

toc

if ~simulator.ignoreForces
  COMPlots(state, time);
end

if ~simulator.ignoreMoments
  EulerPlots(state, time);
end
```

Published with MATLAB® R2020b

7.6.16 Case 3

```
%%CASE 3: Single Moment [Euler]
%Simulates the vehicle as a rigid body with no propellent nor gravity. A single %rolling
moment is applied for 5 seconds starting at t = 5s.
clc; clear global; clf;
ascentRocket = BuildAndromeda();

environment = Environment([100; 0; 0]);

moments = [

  5 10 5 0 0

];

aerodynamics = DefinedMoments(moments, [0 0 0]);

initialState = [

  zeros(3,1)              % Launch Position
  compact(quaternion([0 -45 0], ...  % Launch Attitude
      "eulerd", "ZYX", "frame"))'
  zeros(3,1)
  zeros(3,1)              % Initial Rotational Vel
  ascentRocket.m_pi           % Initial Propellant Mass
];
simulator = Simulator(initialState, ascentRocket, ...
  aerodynamics, environment, true, false, 10);
tic

[time, state] = simulator.Execute(initialState);

toc

if ~simulator.ignoreForces
  COMPlots(state, time);
end

if ~simulator.ignoreMoments
  EulerPlots(state, time);
end
```

Published with MATLAB® R2020b

7.6.17 Case 4

```
%%CASE 4: Precession [Euler]
%The rocket is fixed at its base, pointed off axis, and given an initial roll. The %force
of gravity causes it to both precess and nutate.
clc; clear global; clf;
ascentRocket = BuildAndromeda();
```

```
environment = Environment([100; 0; 0]);

moments = [

  5 10 0 0 0

];

COM = [ascentRocket.c_ms 0 0];

aerodynamics = DefinedMoments(moments, COM);

initialState = [

  zeros(3,1)              % Launch Position
  compact(quaternion([0 -70 0], ...  % Launch Attitude
      "eulerd", "ZYX", "frame"))'
  zeros(3,1)
  60                  % Initial Rotational Vel
  0
  0
  ascentRocket.m_pi          % Initial Propellant Mass
];
simulator = Simulator(initialState, ascentRocket, ...
  aerodynamics, environment, true, false, 15);
tic

[time, state] = simulator.Execute(initialState);

toc

if ~simulator.ignoreForces
  COMPlots(state, time);
end

if ~simulator.ignoreMoments
  EulerPlots(state, time);
end
```

Published with MATLAB® R2020b

7.6.18 Case 5

```
%%CASE 5: Full Integrated Flight [COM] [Euler]
%Full integrated simulation of the vehicle which solves for position and attitude over
%the flight. Conditions resemble a real launch with a slightly off vertical launch rail
%and an 8 m/s south westerly wind.
clc; clear global; clear; close all; clf;
ascentRocket = BuildAndromeda();

environment = Environment([-8/sqrt(2); -8/sqrt(2); 0]);

aerodynamics = Aerodynamics();

initialState = [
```

```
  zeros(3,1)              % Launch Position
  compact(quaternion([45 -85 0], ... % Launch Attitude
      "eulerd", "ZYX", "frame"))'
  zeros(3,1)              % Initial Velocity
  zeros(3,1)              % Initial Rotational Vel
  ascentRocket.m_pi           % Initial Propellant Mass
];
simulator = Simulator(initialState, ascentRocket, ...
  aerodynamics, environment, false, false);
tic

[time, state] = simulator.Execute(initialState);

toc

if ~simulator.ignoreForces
  COMPlots(state, time);
end

if ~simulator.ignoreMoments
  EulerPlots(state, time);
end
```

Published with MATLAB® R2020b

7.7 Appendix G: Dynamical Simulator Case 5 Graphs

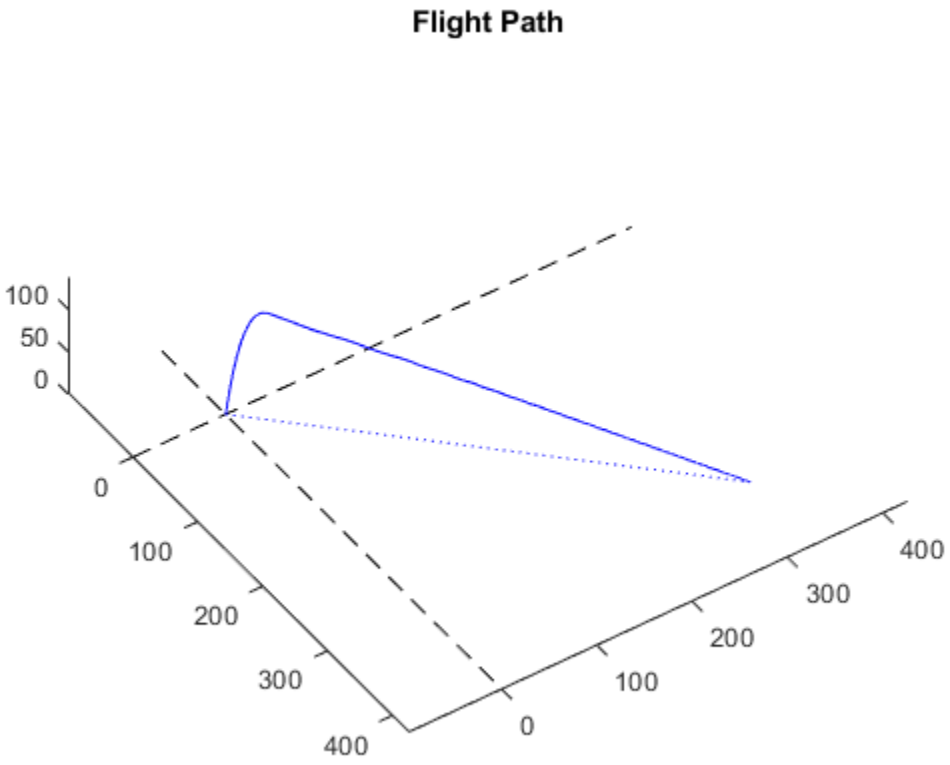


Figure 7-1 Case 5 Flight Path

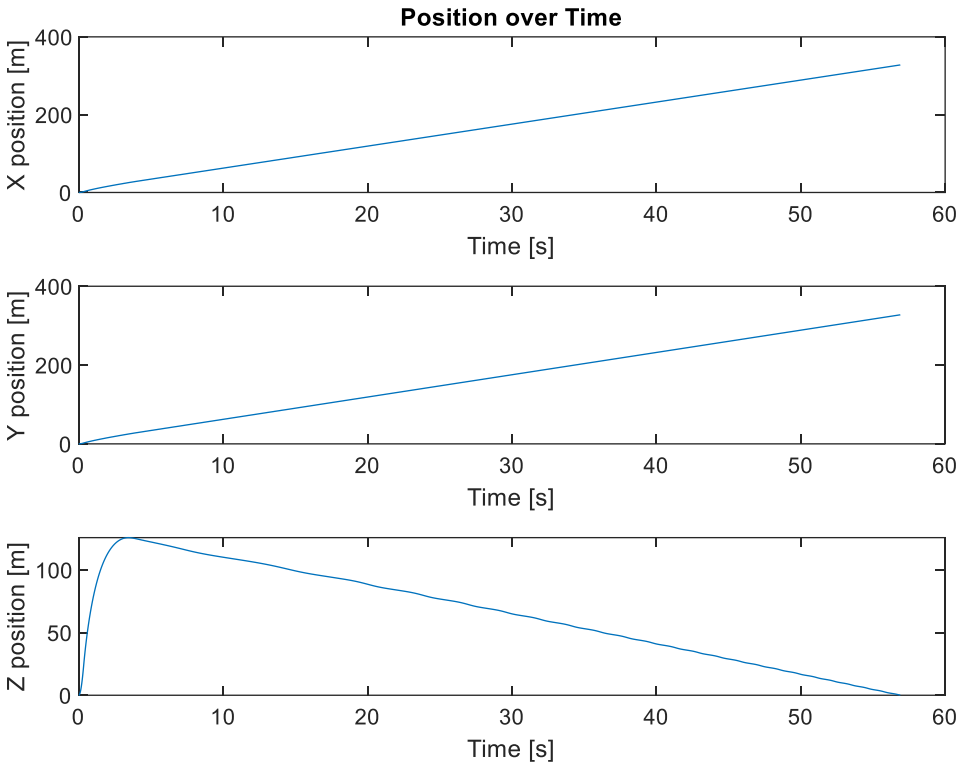


Figure 7-2 Case 5 Position over Time

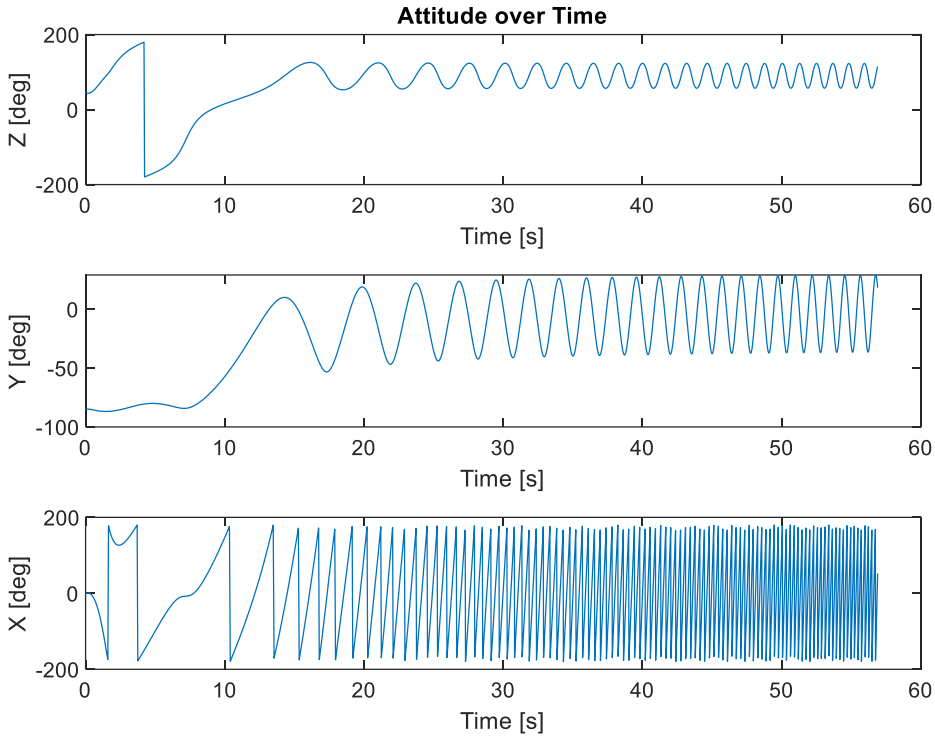


Figure 7-3 Case 5 Attitude over Time

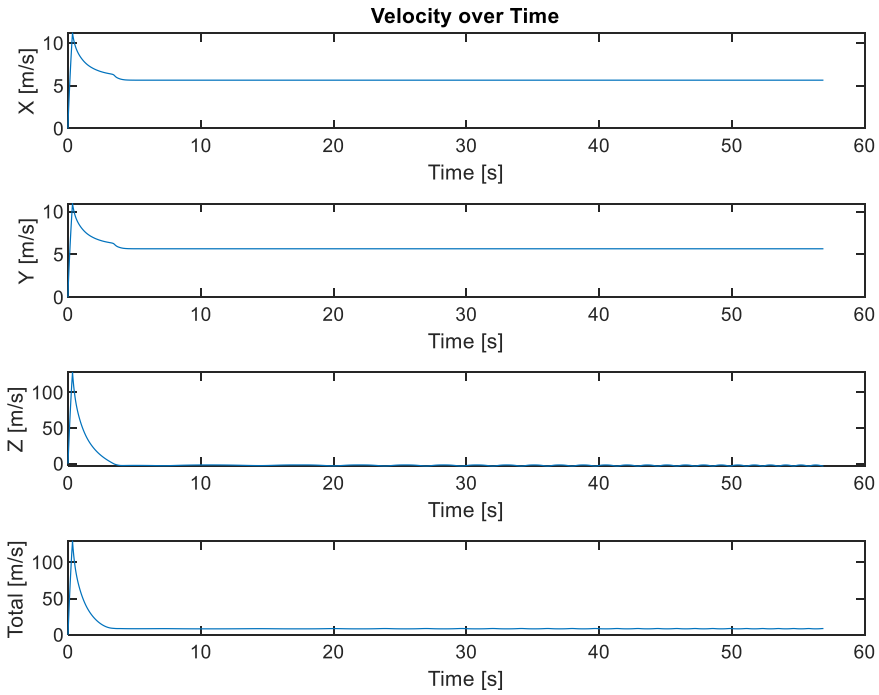


Figure 7-4 Case 5 Velocity over Time

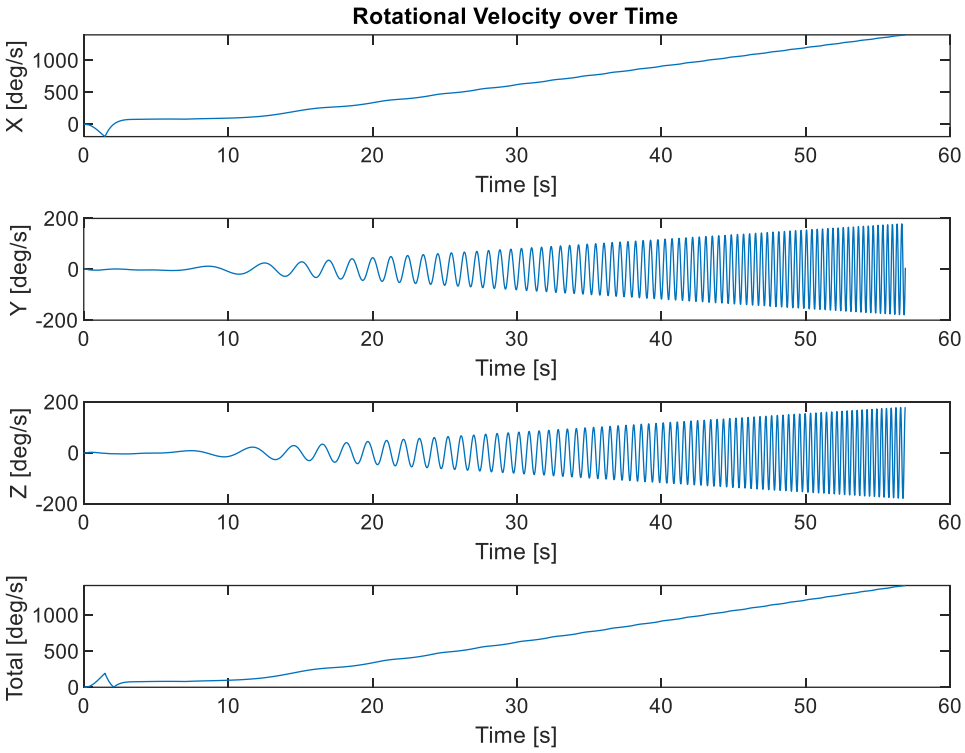


Figure 7-5 Case 5 Rotational Velocity over Time

7.8 Appendix H: Ansys Fluent Aerodynamic Load Plots

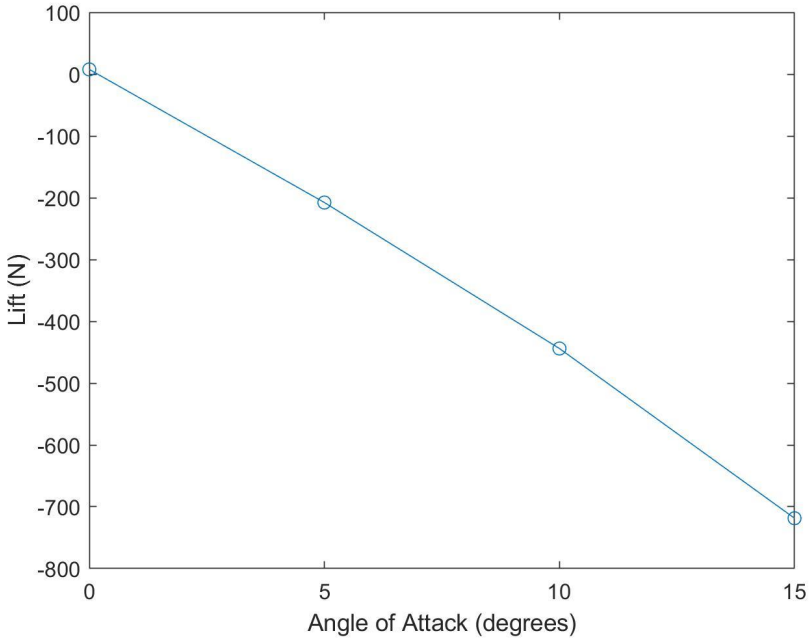


Figure 7-6 Plot of Lift as a Function of Angle of Attack

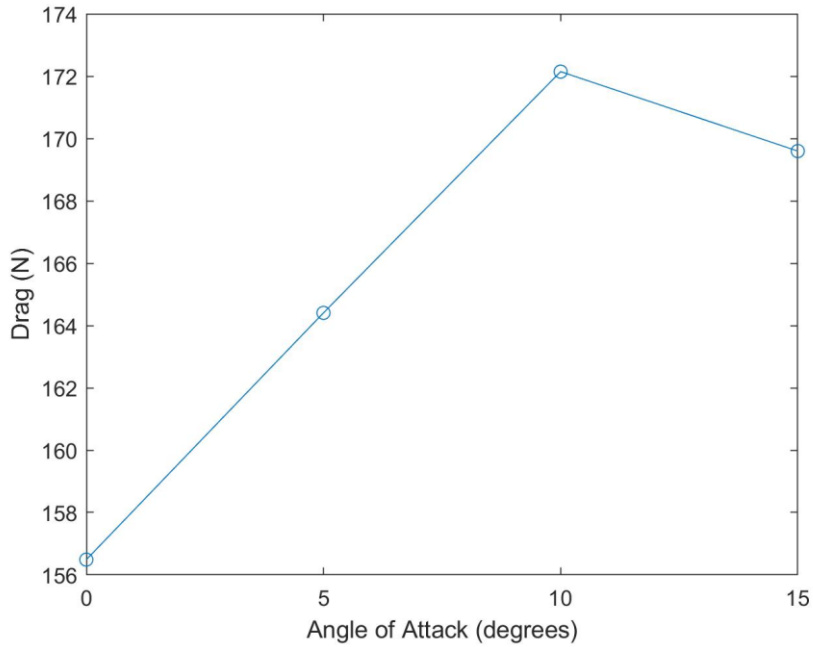


Figure 7-7 Plot of Drag as a Function of Angle of Attack

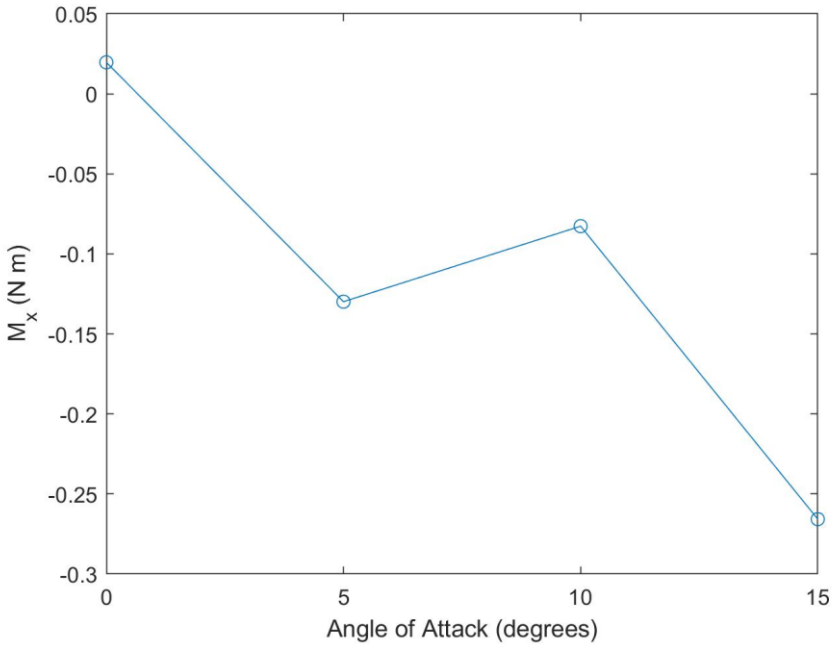


Figure 7-8 Plot of X Moment as a Function of Angle of Attack

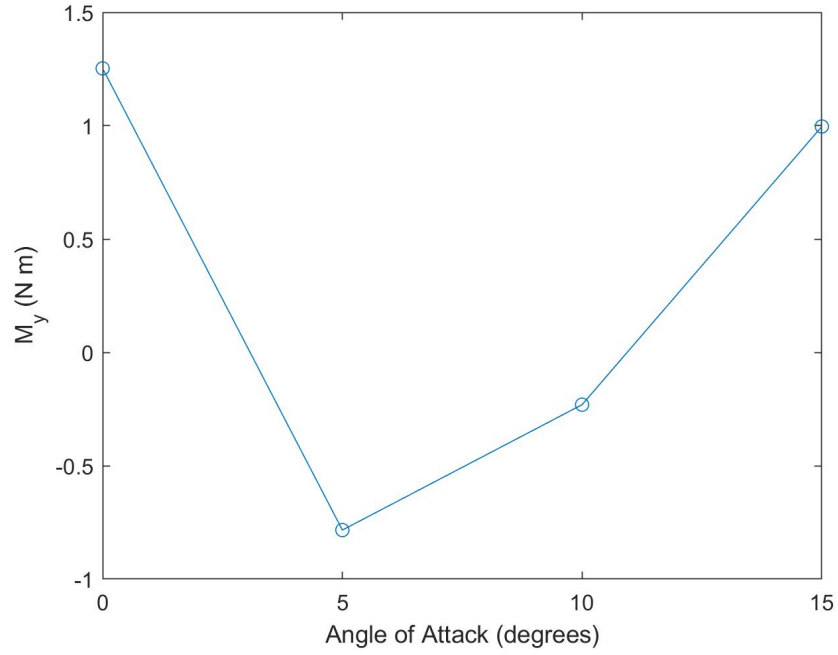


Figure 7-9 Plot of Y Moment as a Function of Angle of Attack

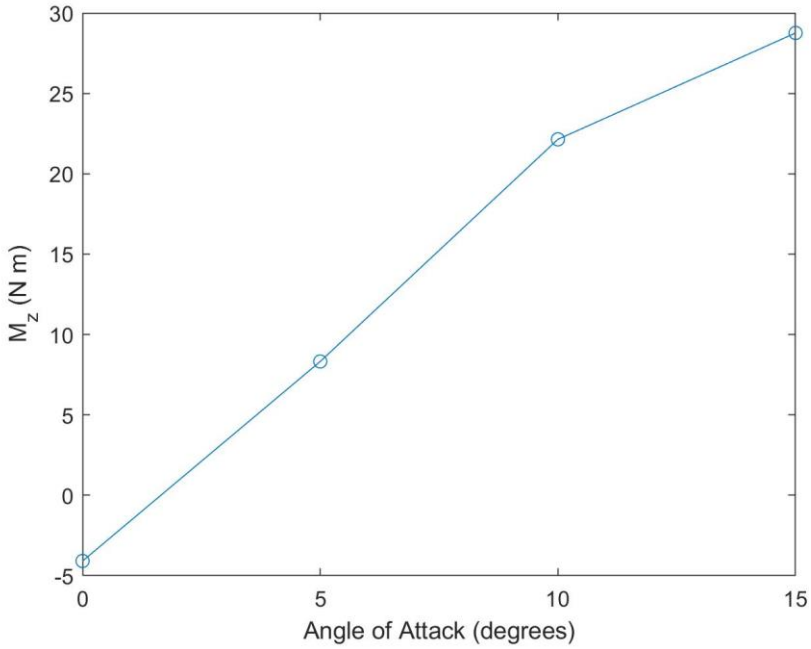


Figure 7-10 Plot of Z Moment as a Function of Angle of Attack

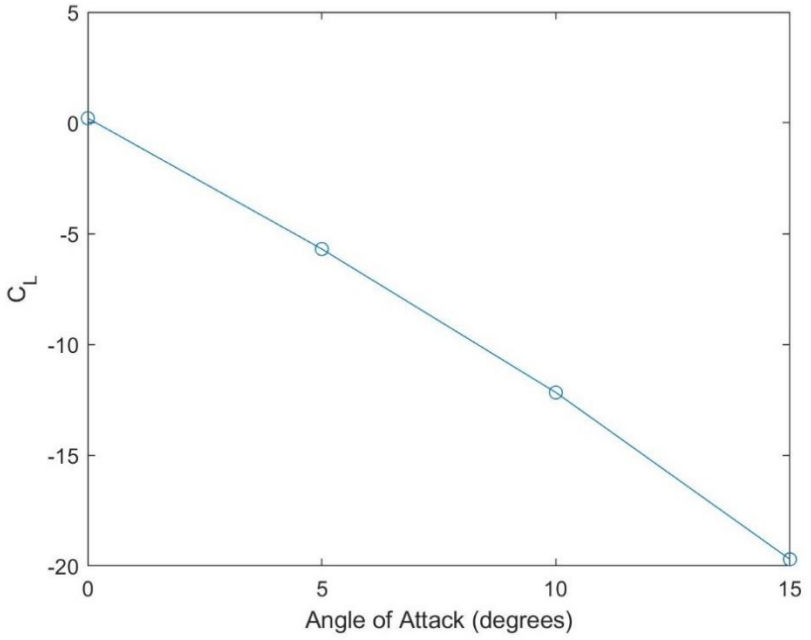


Figure 7-11 Plot of Lift Coefficient as a Function of Angle of Attack

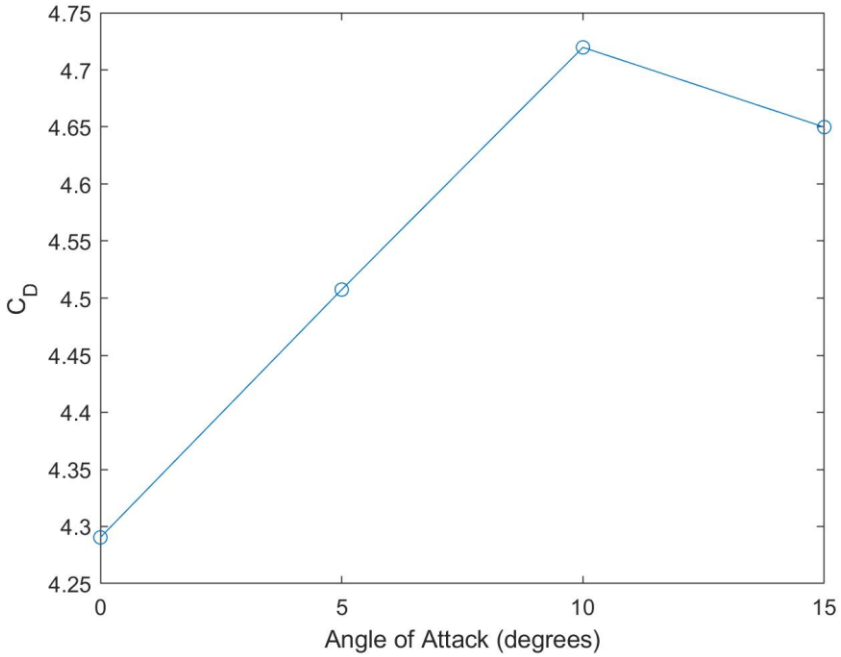


Figure 7-12 Plot of Drag Coefficient as a Function of Angle of Attack

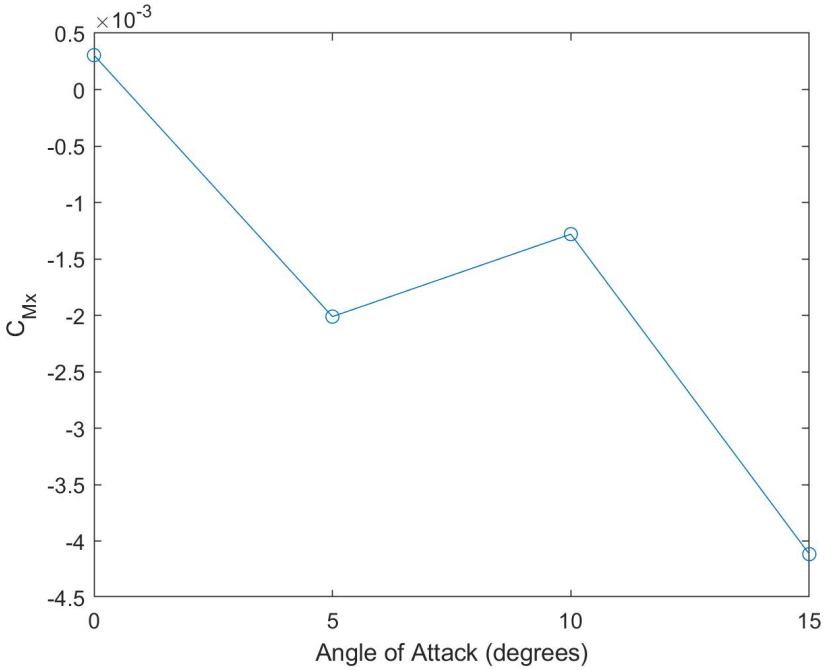


Figure 7-13 Plot of X Moment Coefficient as a Function of Angle of Attack

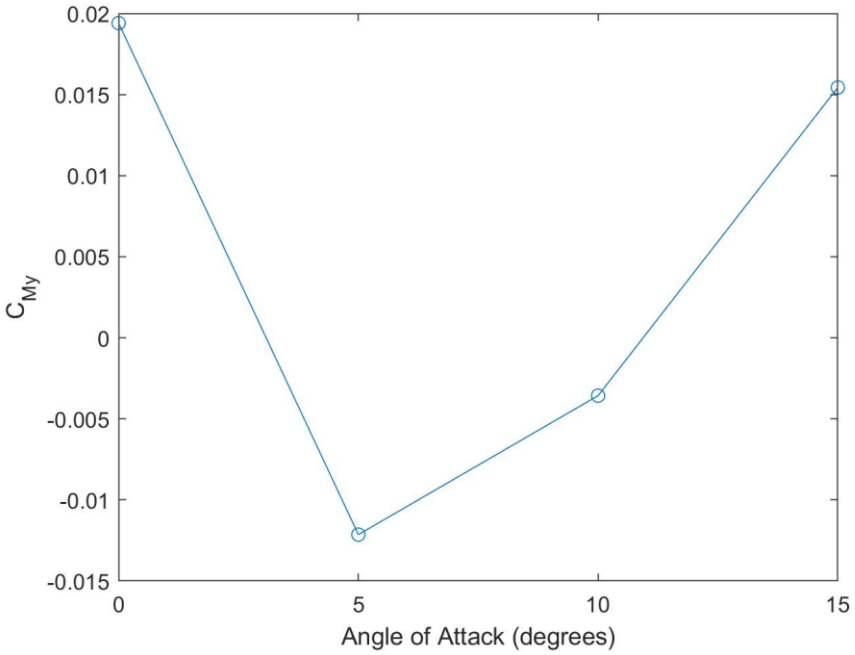


Figure 7-14 Plot of Y Moment Coefficient as a Function of Angle of Attack

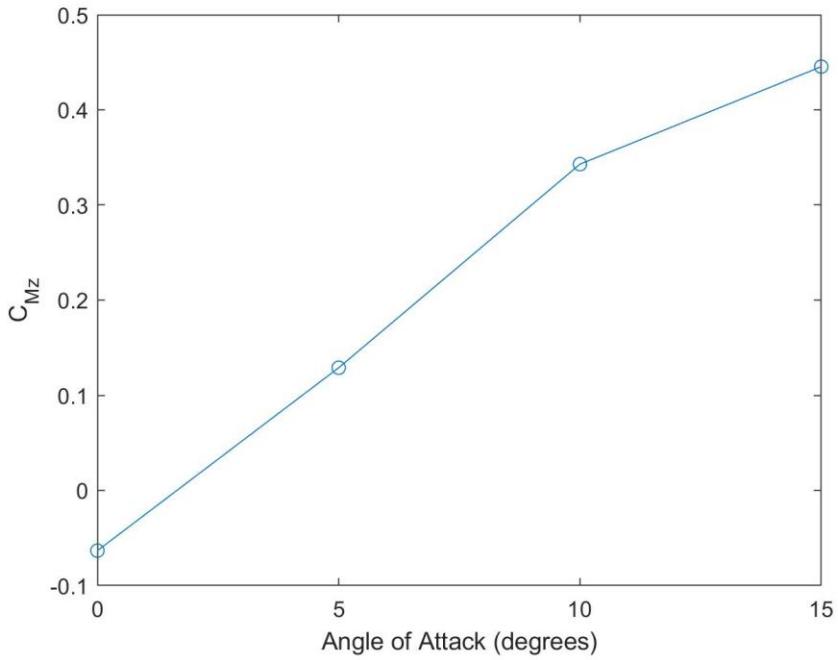


Figure 7-15 Plot of Z Moment Coefficient as a Function of Angle of Attack

7.9 Appendix I: Ansys Fluent Dynamic Pressure Contours

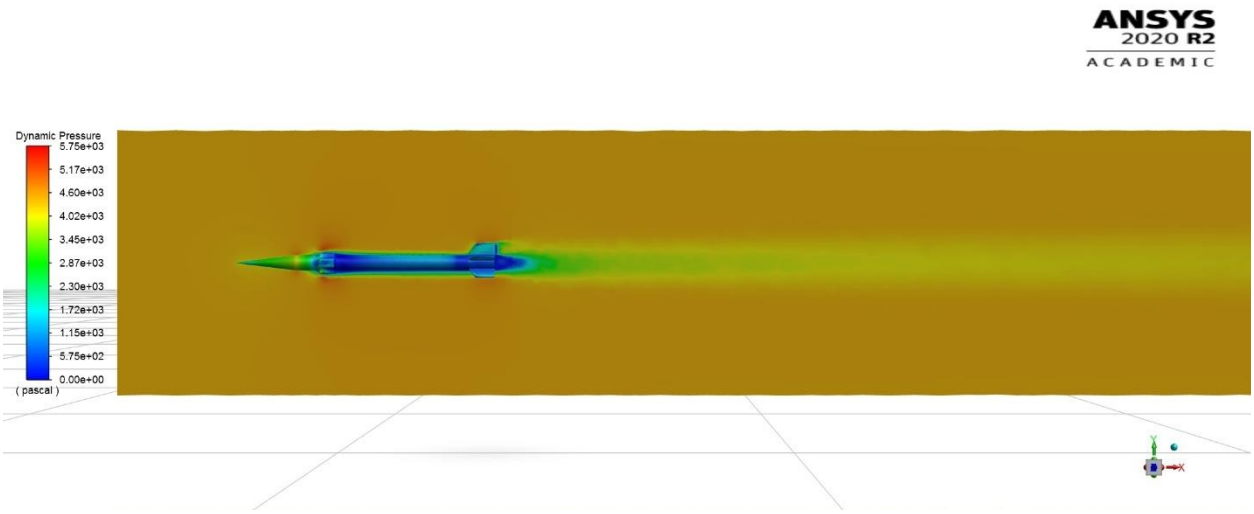


Figure 7-16 Dynamic Pressure Contour at AOA 0 deg

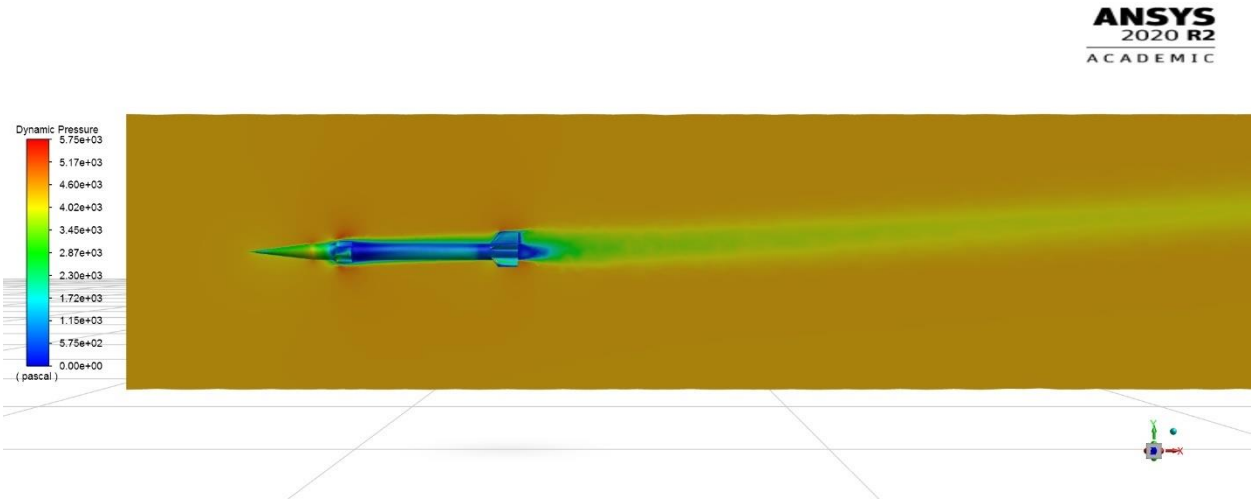


Figure 7-17 Dynamic Pressure Contour at AOA 5 deg

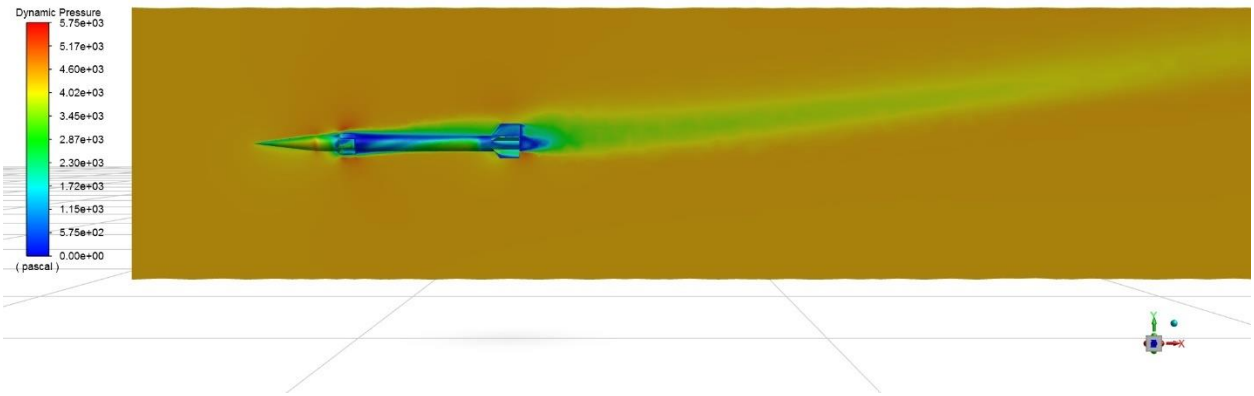


Figure 7-18 Dynamic Pressure Contour at AOA 10 deg

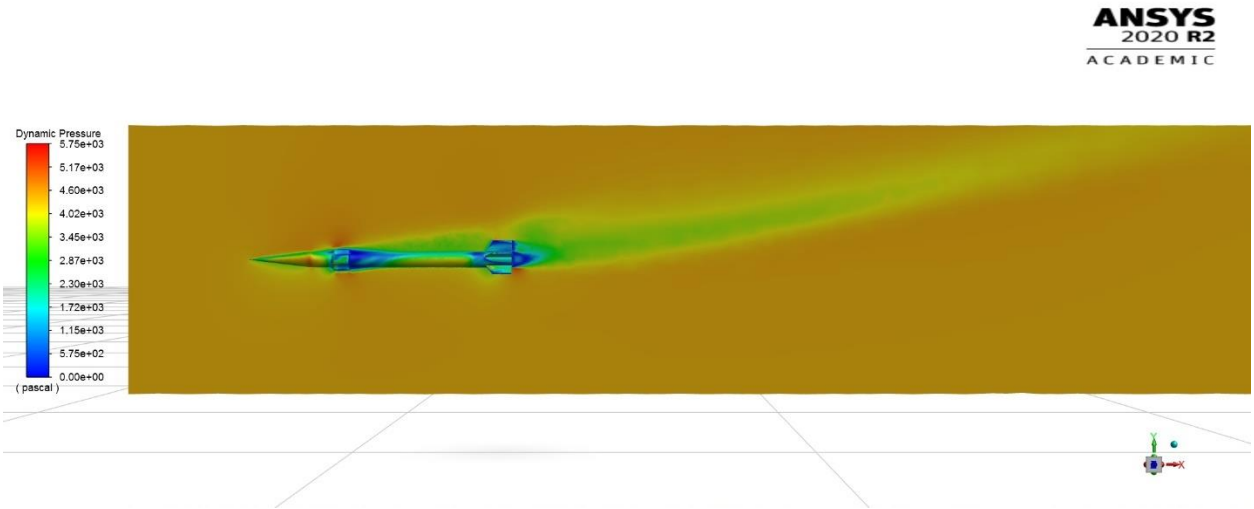


Figure 7-19 Dynamic Pressure Contour at AOA 15 deg