Launch Vehicle Analysis
Launch Vehicle Design II

Yevhenii Kovryzheko 903980521
Christian Bookout 903914427
Jackson Treese 903942098
Grant Turner 903923486
Carter Davis 903897002
Jacob Lee 903845980

AERO 4720
Group 1
Auburn University

April 13, 2021

Contents

Nomenclature

ΔV ..... delta v / burnout velocity
γ ..... flight path angle or propellant mixture ratio
μ ..... mass ratio
ϕ ..... latitude
ρ ..... density
ρ_0 ..... initial/reference density
σ ..... structural ratio
σ_H ..... hoop stress
σ_L ..... longitudinal stress
A ..... area
A_e ..... exit area
C_D ..... drag coefficient
D ..... drag
g ..... acceleration due to gravity
g_0 ..... initial/reference gravitational acceleration
h ..... altitude/height
h_0 ..... reference altitude/initial height
$I_{sp,\ eff}$ ..... effective specific impulse
$I_{sp,\ SL}$ ..... specific impulse at sea level
$I_{sp,\ vac}$ ..... specific impulse in vacuum (theoretical)
I_{sp} ..... specific impulse
L ..... Lagrange multiplier or length
l ..... length
m ..... mass
m_0 ..... initial mass
m_f ..... final or b/o mass
m_p ..... propellant mass
m_s ..... structure mass
m_{pl} ..... payload mass
P ..... pressure
P_0 ..... initial/reference pressure
P_∞ ..... free stream pressure
r ..... radius

R_E ..... radius of Earth
S_{ref} ..... reference area (for aerodynamic calculations)
T ..... thrust
t ..... time
T ..... mass flow rate
V ..... volume
v ..... velocity
v_e ..... exhaust velocity
v_f ..... final/burnout velocity

List of Figures

Concept Design and Design Philosophy

Overview and Vehicle Design Approach

Blue Oregano has a goal of developing, designing, and constructing a launch vehicle capable of sending up to 50 metric tons of un-crewed payload into low Earth orbit. Blue Oregano's mission as a company is to provide the world with a cost-effective and successful launch vehicle to aid in supplying future space travelers with goods that will enhance their quality of life. Blue Oregano hopes to get ahead of the industry by shipping luxury items into space. Blue Oregano hopes to be the primary launch provider in the newly developed space trade system and spur on a modern day silk road. In order to achieve this goal, the team at Blue Oregano has begun work towards a new program coined Advanced Rocket Trade Enterprise: Mars International Space Station (ARTEMISS). The ARTEMISS program will bring around a new launch vehicle designed to spark the beginning of the space fairing trade industry. This new launch vehicle will be called Genesis 1. This launch vehicle will consist of two stages. Both of these stages will be powered by liquid rocket engines that provide ample thrust to get the vehicle payloads into orbit. The approach to this design was to maximize efficiency by minimizing mass and optimizing the ascent trajectory of the vehicle

Requirements Compliance Matrix

Requirement Number	Requirement	Means of Compliance
1	The launch vehicle shall be capable of placing a 50 metric ton un-crewed payload into low Earth orbit (LEO) at an equatorial orbit of a minimum of 800km altitude.	The Genesis I consists of two stages. The first stage will consist of nine Raptor engines, and the second stage will consist of 7 RL10C-1 engines. Each Raptor engine can produce 2 MN of thrust and gimbal, and each RL10C engine can produce 102 kN of thrust and gimbal if necessary. Thus, both stages provide the necessary means to propel the launch vehicle into LEO.
2	The vehicle cargo module shall be capable of docking with the NASA International Docking Adapter.	The Genesis I will be capable of attaching a docking system that will be able to dock with the NASA International Docking Adapter. Customers will be allowed to provide their own payload adapter and payload separation system. To comply with necessary protocols, payload adapters will be required to consist the major components of a Payload Separation Ring (PSR), Launch Vehicle Adapter, and a Payload Separation System.
3	The vehicle design process shall complete the design, analysis, and test flight stage no later than 4th Quarter, FY 2036 and your design shall have a complete GANTT chart of the anticipated design, testing, and flight schedule	By meeting scheduled testing and flight dates, the Genesis I shall be mission capable by FY 2036. The GANTT chart incorporated within this report will ensure the progress of the Genesis I's design and gradual completion.
4	The vehicle shall be designed such that current infrastructure can be utilized with minimal or no modifications. No stand-alone or new large scale facilities can be built as a result of the design.	The Genesis I's testing, development, and construction will not need new large scale facilities for completion of design. The aforementioned abilities will take place at pre-existing NASA facilities and NASA partnered corporations. Additionally, the Genesis I will launch out at the Kennedy Space Center.
5	The vehicle shall be designed to meet human-rating requirements in NPR 8705.2C and design constraints set forth in the Commercial Crew Transportation System Certification Requirements dated 12/9/2010.	The Genesis I will follow the follow the four functions of the Commercial Crew Transportation System Certification Requirements of 1) validation of the technical and performance requirements/standards; 2) verification of compliance with those requirements/standards; 3) consideration of relevant operational experience; and 4) acceptance of residual technical risk due to hazards, waivers, non-compliances, etc. Moreover, all design pieces of the launch vehicle will be arranged so that ratification of the agreement milestone, statement of work, contract requirements, engineering and operations plans etc. completed by the NASA Program Manager will be done ethically and swiftly. The vehicle will also be designed to follow NASA STD 3001, FAA HFDS, MIL STD 1472, NASA STD 3000 Vol I-II, JSC requirements, NPD, NPR, ANSI, IEEE, and further systems engineering requirement sets to meet human-rating requirements in NPR 8705.2C.

Table 1: Requirements Compliance Matrix

Concept of Operations (CONOPS)

The launch vehicle flight path will be primarily determined by the gravity turn maneuver executed at an altitude of 1400 meters with a small deflection of 2.3° from the vertical using the on-board thrust vectoring control system. To minimize the losses due to drag and unnecessary thrust vectoring, the vehicle will primarily follow the natural path of the gravity turn maneuver to the initial low Earth orbit. The first stage, with nine Raptor engines, is used to escape earth's atmosphere, and is going to absorb all aerodynamic effects and provide the majority of the orbital velocity required to reach the final destination. Moreover, both steps of the vehicle are burned sequentially to reach the orbital velocity. A small fraction of the propellant on the second stage, with seven RL-10C engines, will be used to establish an initial, close-to-circular orbit at approximately 350 km to 550 km altitude and a second orbital transfer maneuver will then be executed to reach the final orbit at 800 km to 900 km altitude using the leftover propellant on the second stage. This process can be visualized in Figure 2

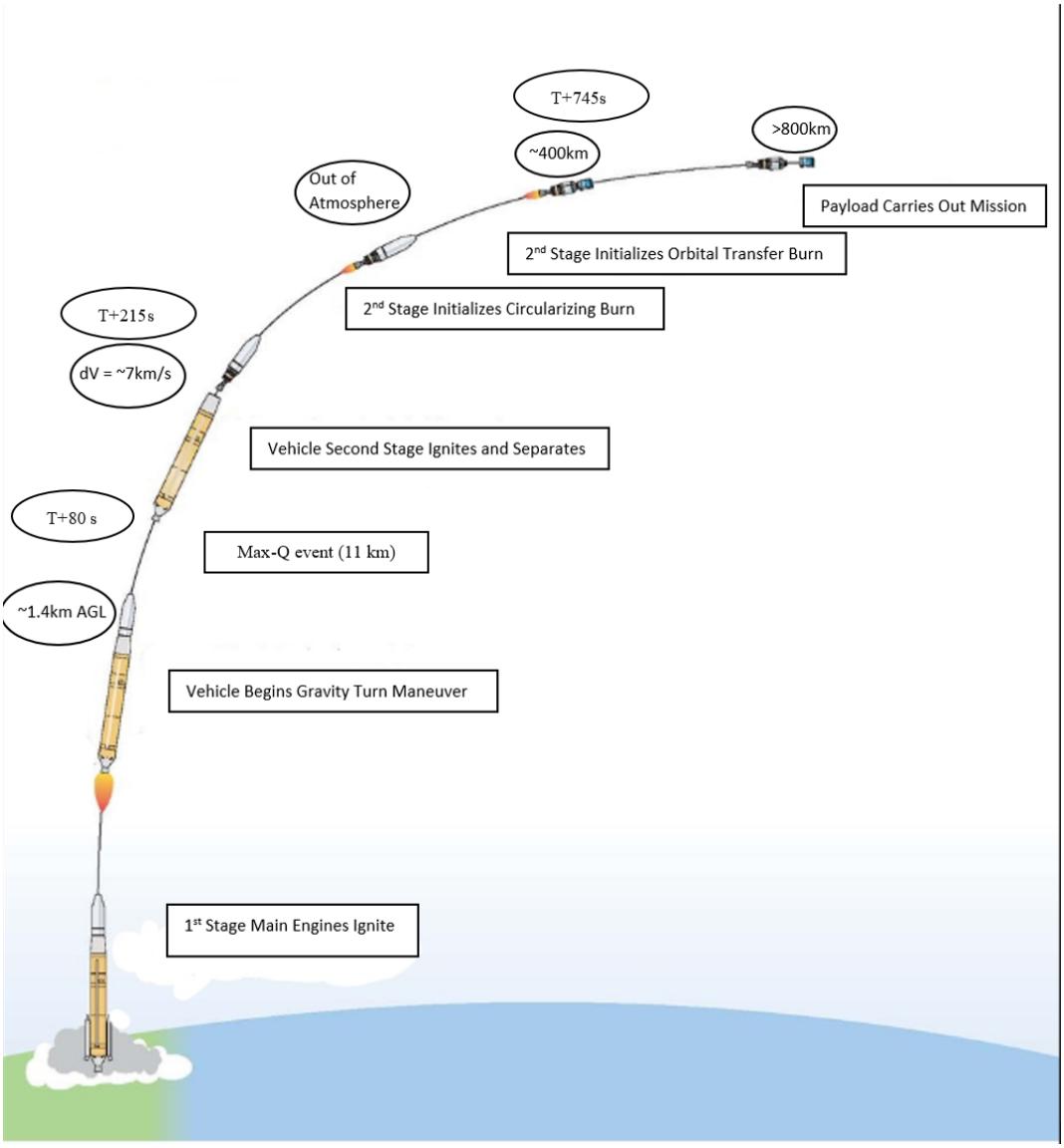

Table 2: Vehicle CONOPS

Human rating certification process

NASA has a number of key points regarding human safety ratings within the documents; NASA-Standard-3000 Volume I-11, Man-Systems Integration Standards; NASA-Standard-3001 Volume 1, Space Flight Human Systems: Crew Health;FAA HFDS-Human Factors Design Standard; and MIL-STD-1472, Department of Defense Design Criteria Standard-Human Engineering [1]. These human ratings are required for any vehicle on which humans may fly and is measured by the PLOC metric or possibility of loss of crew metric.

To get a Human Rating Certification a Vehicle would have to get rated on a scale of failures being "1-in-a certain number of launches" however, this vehicle will not be human rating certified due to its lack of manned power and payload delivering nature. If the vehicle would require certification, the NASA chain of command including the Program Manager, Associate Administrator for Responsible Mission Directorate, Technical Authorities, and the Director of the NASA facility at which certification is occurring would be required to get the process started. However to complete the certification both the NASA administrator and associate administrator would need to endorse the certification[1] as shown in Fig 1 below.

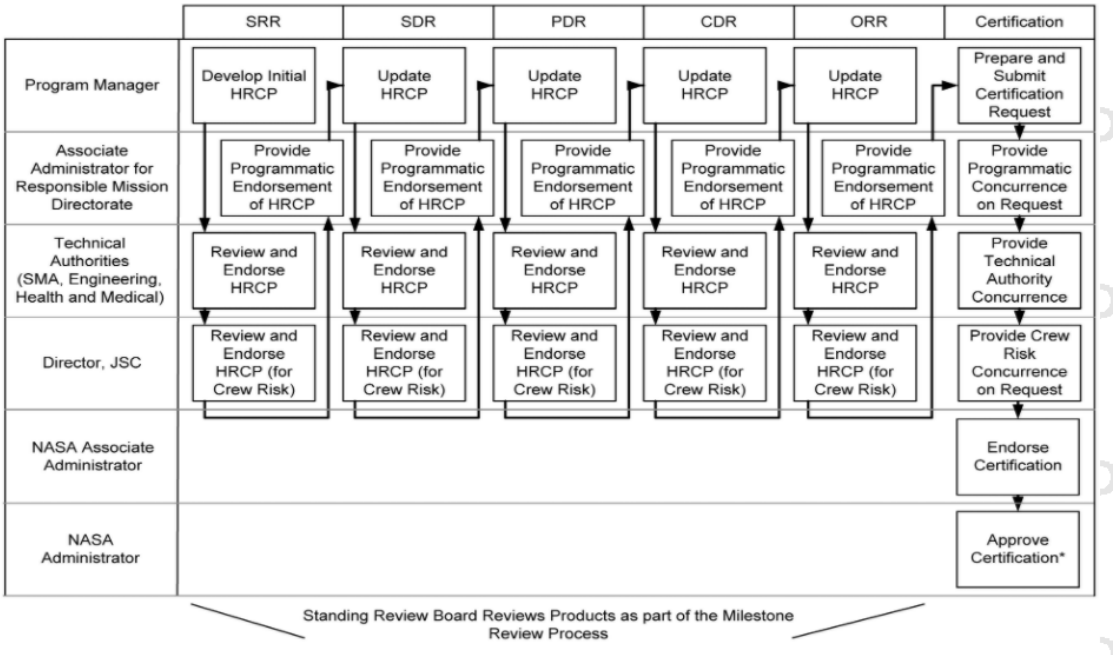

Figure 1: Human Rating Certification Chain of Command Requirements

Budget

For this preliminary cost analysis, 20% of NASA's FY2019 budget was assumed to be available to spend each year for the duration of the program. According to Lambright [2], NASA was granted $19,892.2 billion for FY2019. The program was assumed to start in 2021 and complete vehicle design and testing by the year 2036, which gives 15 years for the entire program and will be used for cost and budget estimation.

Advanced Mission Cost Model and Budget Estimation

To estimate the total available budget over the span of 15 years, the total budget has to be adjusted for inflation. Using the US Bureau of Labor Statistics inflation formula, inflated cost can be calculated in the following manner:

$$C_{2036} = C_{2021} \times R_{inf}^{(2036-2021)}, \tag{1}$$

where R_{inf} is the inflation rate and C_N is cost in US \$ in N_{th} year.

Using the previous equation (1) and official data from US Bureau of Labor Statistics, the inflation rates for years 2019-2020 were calculated to be 2.5%. Using this inflation rate, the total available projected funding in 2036 US\$ can be calculated in the following manner:

$$Total\ budget = \sum_{i=1}^{15} 0.2 \times 19,892.2 \times 1.362005^{i} = \$73,064.30\ billion \tag{2}$$

To estimate the actual cost of the the program, the Advanced Missions Cost Model (AMCM) was used. AMCM is a single equation (3) which uses the dry mass of the vehicle M (in lbs), quantity of development and production units Q, mission type S, number of design generations B, technical difficulty D and year of initial operation capability IOC to estimate the total cost for DDT&E and production.

$$Cost = 5.65 \times 10^{-4} Q^{0.59} M^{0.66} 80.6^{S} (3.81 \times 10^{-550})^{1/(IOC-1900)} B^{-0.36} 1.57^{D} [3] \tag{3}$$

According to Owens[4] and Jones [3], the specification value S for a launch vehicle is 1.93. AMCM parameters were summarized in Table 4 and the following list provides a basic breakdown of each parameter:

- Quantity Q: The number of development and production units produced, including test articles and spares.
- Mass M: System dry mass in pounds. This parameter can be converted for input in kilograms to measure mass, and therefore a factor of 2.2 lb/kg needs to be applied for calculation if needed [4].
- Specification S: The type of mission to be flown. This input has to be chosen according to the type of system. System types and their specification values are listed in Table 3.
- Initial Operational Capability IOC: The first year of system operations.
- Block B: The system's block number, or level of design inheritance. For example, a completely new design has a block number equal to 1, while modifications to existing designs have block numbers of 2 or more.
- Difficulty D: A subjective input describing the expected programmatic and technical difficulty of producing the system. Difficulty ranges from -2.5 to 2.5 in increments of 0.5, where -2.5 indicates "extremely easy"and 2.5 indicates "extremely difficult."

Assuming a rough dry weight of the launch vehicle to be 120 metric tons (264,000 lbs.) 6 vehicles for testing and validation and medium mission difficulty the AMCM cost estimate in 1999 US dollars is as follows:

$$Cost_{LV} = 5.65 \times 10^{-4} \times 6^{0.59} \times (120,000 \times 2.2)^{0.66} \times 80.6^{1.93} \times (3.81 \times 10^{-55})^{1/(2036-1900)} \times 1^{-0.36} \times 1.57^{0.5} = 14,670.76 billion \tag{4}$$

Specification	S **Value**
Planetary Lander	2.46
Planetary	2.39
Human Reentry	2.27
Rovers	2.14
Human Habitats	2.13
Launch Vehicle	1.93

Table 3: Advanced Mission Cost Model Summary

	Launch Vehicle	**Transfer Vehicle**	**Lander**	**Rover**	**CRV**
Quantity Q	6	0	0	0	0
Dry Mass M (kg)	120,000	0			
Mission Type S	1.93	2.39	2.46	2.14	2.27
Year IOS	2036	2036	2036	2036	2036
Block B	1	1	1	1	1
Difficulty D	0.5	1.0	2.5	2.0	1.0
Cost billions of $US	33,155.92	0	0	0	0

Table 4: Advanced Mission Cost Model Summary

The AMCM formula provides an estimate cost of the entire program in billions of 1999 dollars [5] and has to be adjusted for inflation using average inflation rate for years 1999 through 2021 which is 2.05%. Therefore, the total cost of the program is 22, 926.55 billion of 2021 US dollars or 33, 155.92 billion of 2036 US dollars, assuming a constant rate of inflation of 2.49% for 2021-2036 period.

In summary, the estimated cost of the program is $33, 155.92 billion with $73064.30 of available funding.

Timeline

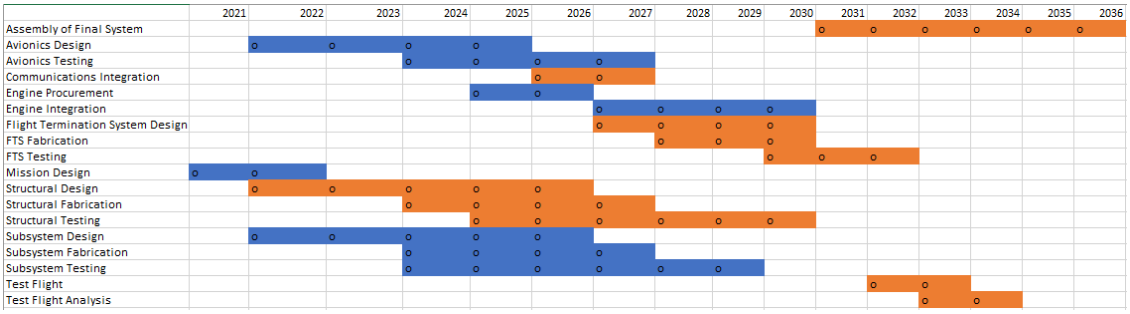

Figure 2: GANTT Chart by Year

Overall vehicle specifications

Flysheet

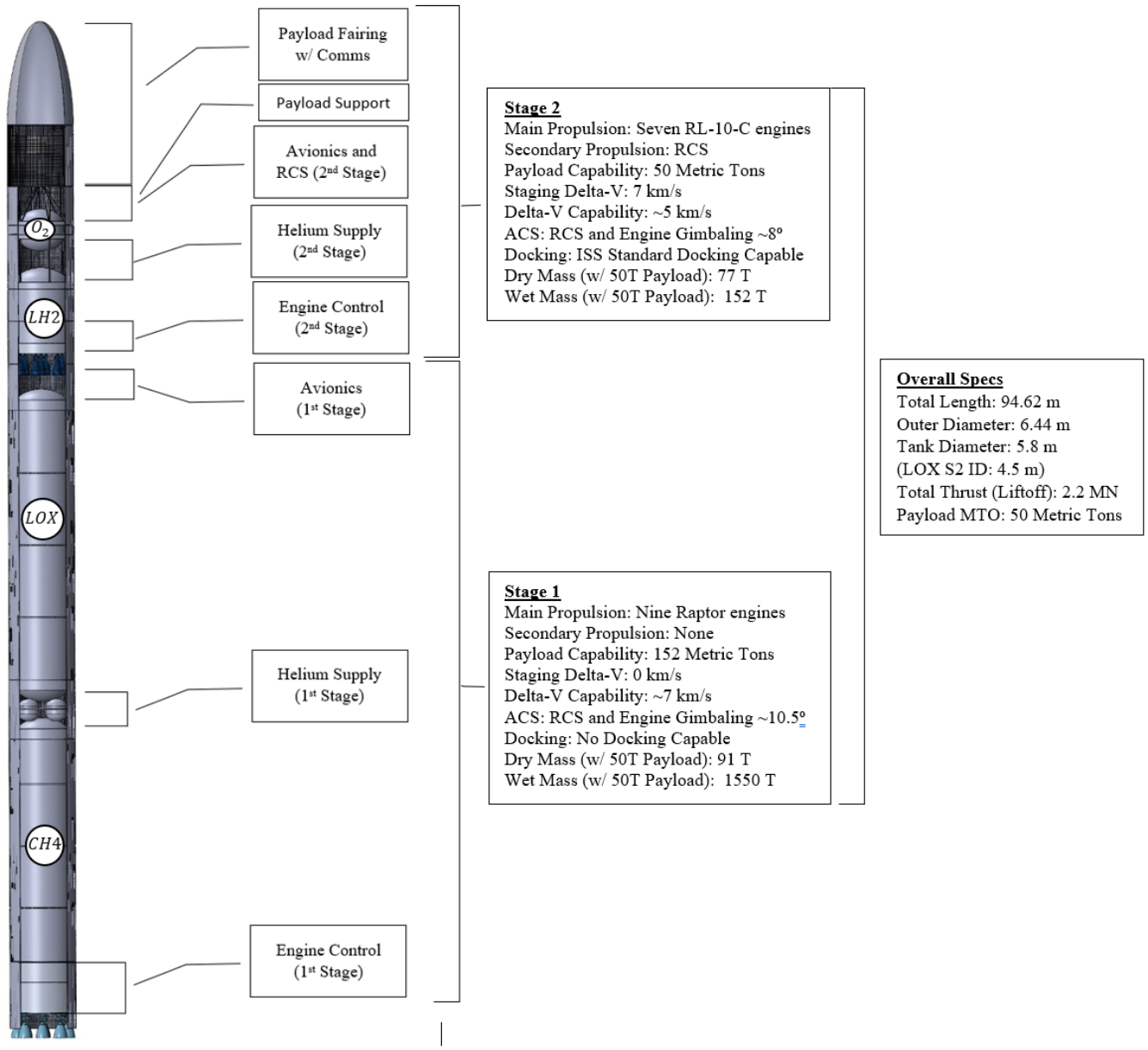

Figure 3: Vehicle Flysheet

Delta V Requirements

The mission profile requires that the vehicle reach an altitude of no less than 800km with the specified payload. In order to determine the necessary delta V for such an operation, a MATLAB script was written to find total delta V given a particular launch site.

Launch Location

In order to select a launch location, the three main factors were taken into consideration. These factors were availability, efficiency, and most importantly, safety. The first of these factors, availability, is extremely important when considering a launch site. The site needs to be rated for a vehicle of this size, and open for launch. Additionally, in order to avoid international negotiations and fees, a launch site in America is preferable.

An efficient launch site would be one such where the additional delta V can be minimized as much as possible. A site very far from the equator will result in a large amount of delta V lost to return to an equatorial orbit, while a launch site closer to the equator will minimize this, and increase the amount of initial velocity granted by the rotation of the earth. Thus, a launch site very close to the equator is preferred.

Should the vehicle suffer any catastrophic failure, it is important to minimize the amount of casualties and damage inflicted. Thus, conducting a gravity turn over land, especially highly populated areas, is not ideal. A launch site which can provide a downrange area with little to no risk of casualties or inflicting damage in the event of a catastrophic failure is necessary.

Given the importance of all these criteria, Launch Complex 39A at the Kennedy Space Center was chosen as the launch location for our launch vehicle. As far as availability, it is a launch site in America with a maximum weight capacity of well over 2500 Metric tons, much more than our vehicle will need [6]. As far as efficiency is concerned, this launch site is extremely close to the equator at a latitude of 28.58 degrees, minimizing the amount of delta V to cancel and maximizing the boost from the Earth's rotation. Where safety is concerned, a flight path from this location would see the vehicle flying directly over open ocean, which is ideal as opposed to the risks inherent with flying over land.

Delta V Calculations

First, for a given launch location, the delta V gained by the earth's rotation can be determined using the earth's rotational velocity at the equator as well as the latitude of the launch site.

$$V_i = V_{eq} cos(\phi) = 465.1 cos(28.58) = 408.45 m/s \tag{5}$$

Next, the necessary delta V to reach a circularized orbit outside of the atmosphere, 160km, was found by taking the difference between the orbital velocity at that altitude from the previously calculated initial velocity.

$$\Delta V_{160km} = \sqrt{\frac{\mu_E}{r_E + h}} - V_i = \sqrt{\frac{3.986e14}{6378e3 + 160e3}} - 408.45 = 7808.1 m/s \tag{6}$$

Additional considerations needed to be made on the burn into orbit for drag and gravity losses. Since at this time losses due to drag were unknown, the gravity losses were estimated to be 2.5 times the drag losses

$$\Delta V_{grav} = -\frac{\sqrt{2g_o h}}{\left(1+\frac{h}{r_E}\right)} = -\frac{\sqrt{2(9.81)(160e3)}}{\left(1+\frac{160e3}{6378e3}\right)} = -1750.0m/s \tag{7}$$

$$\Delta V_{drag} = \frac{\Delta V_{grav}}{2.5} = \frac{1.75e3}{2.5} = -700.0m/s \tag{8}$$

Once the spacecraft is in orbit, a circular Hohmann transfer, consisting of two burns, will be conducted to reach the altitude of 900km.

$$\Delta V_{H,1} = \frac{1}{r_1}\sqrt{2\mu_E}\sqrt{\frac{r_1 r_2}{r_1+r_2}} - V_{160km} = 206.4m/s \tag{9}$$

$$\Delta V_{H,2} = V_{900km} - \frac{1}{r_2}\sqrt{2\mu_E}\sqrt{\frac{r_1 r_2}{r_1+r_2}} - = 200.92m/s \tag{10}$$

Thus, the total delta V of the spacecraft, with a 7% factor of safety, was determined to be 10.975km/s.

Staging

To simplify the design process and after studying the historical data on launch vehicles, it was determined to pursue a two-stage launch vehicle. Two, vertically stacked steps are going to be used for delivering the required payload mass to the final orbit.

To begin the sizing process of the vehicle, mass fractions for each stage had to be determined using an optimization Matlab routine. The formulation presented in this section in based on chapter 5 of [7] and section 8.2 of [8].

Structural mass ratio σ_i and mass ratio μ_i for each step i of an N stage vehicle are defined by equations (11) and (12):

$$\mu_i = \frac{m_{0i}}{m_{0i} - m_{pi}} \tag{11}$$

$$\sigma_i = \frac{m_{si}}{m_{si} - m_{pi}} \tag{12}$$

Using the efficiency of each step, defined by I_{sp} value (in seconds) of the propulsion system and acceleration of gravity of Earth sea level ($g_0 = 9.80665\ (m/s^2)$), the final speed provided by a multi-stage vehicle without taking into account gravity and aerodynamic losses is:

$$v_f = \sum_{i=1}^{N} g_0 I_{sp}\ ln\mu_i \tag{13}$$

Payload ratio is also defined as the total initial mass of the vehicle over the payload mass $f(\sigma, \mu) = \frac{m_{01}}{m_{pl}}$ which has to be minimized. By taking a logarithm of the payload ratio and expressing it as a summation of mass fractions and structural fractions for each step yields the following equation:

$$ln\frac{m_{01}}{m_{pl}} = \sum_{i=1}^{N} \left[ln\mu_i + ln(1-\sigma_i)\ ln(1-\mu_i\sigma_i)\right], \tag{14}$$

which is subject to the constraint (13), rearranged to yield zero when given the required velocity v_f:

$$g(\sigma, \mu) = \sum_{i=1}^{N} g_0 I_{sp}\ ln\mu_i - v_f = 0 \tag{15}$$

It is also necessary to introduce *Lagrange multipliers*, L_i, for each step i to a function f. Using the constraint expressed by (15), variable L and payload ratio f, the final equation has the form of $h = f + Lg$. Function $h(\sigma, \mu)$ that needs to be minimized can is expressed as such:

$$h(\sigma, \mu) = ln\frac{m_{01}}{m_{pl}} \sum_{i=1}^{N} \left[(ln\mu_i\ ln(1-\sigma_i) - ln(1-\mu_i\sigma_i)) + L_i\left(g_0 I_{sp,i}\ ln\mu_i - v_f\right)\right] \tag{16}$$

Equation (16) was differentiated with respect to mu_i to yield N equations of the form:

$$\frac{1}{\mu_i} + \frac{\sigma_i}{1-\mu_i\sigma_i} + L\frac{g_0\ I_{sp,i}}{\mu_i} = 0$$

which can be solved for the mass ratio:

$$\mu_i = \frac{1 + L_i \; g_0 \; I_{sp,i}}{L_i \; g_0 \; I_{sp,i} \; \sigma_i} \tag{17}$$

By substituting the mass ratio expressed by (17) into the definition of the burnout speed for a multistage vehicle (13) to obtain:

$$v_{f,i} - \sum_{i=1}^{N} g_0 \; I_{sp,i} \; ln\left(\frac{1 + L_i \; g_0 \; I_{sp,i}}{L_i \; g_0 \; I_{sp,i} \; \sigma_i}\right) = 0 \tag{18}$$

Where (18) is the minimum problem to be iteratively solved for L_i with the selected structural ratios σ_i, the propulsion system is characterized by the specific impulse I_{sp}, gravitational acceleration at Earth sea-level g_0, and required burnout speed $v_{f,i}$ for each step i. More details on this derivation are presented in section 5.9 of [7]. Note, to solve a problem for a multistage vehicle, a separate Lagrange multiplier L has to be used and solved separately for each step individually to match the allocated burnout speed.

The total required burnout speed v_{design} was determined by adding the required orbital speed at the first low-altitude circular orbit ($\approx 160 \; km$) with the Δv required for the Hohmann transfer to the final circular orbit at 800 km to 900 km. Moreover, extra Δv was accounted for losses due to drag during ascent, losses due to gravity, and Δv gain due to the rotation of Earth according to the launch site location.

The final step to begin the optimization routine was to incorporate the total loses Δv_{loss} and the total burnout Δv_{bo} after the vehicle has used all of its **usable** propellant into the optimization model. This requirement was formulated by introducing two parameters: β_i, which allocates how much of the total losses Δv_{loss} each step is absorbing, and α_i to control how much of the total burnout speed Δv_{bo} is provided by each step. The total design speed Δv_{design} for the entire vehicle can be expressed in terms of the intermediate requirements for each step of the launch vehicle such that:

$$\Delta v_{design} = \sum_{i=1}^{N} \left(\alpha_i \; \Delta v_f + \beta_i \Delta v_{loss}\right), \tag{19}$$

where α_i and β_i are defined such that:

$$\begin{gathered} \alpha_i, \;\; \beta_i \geq 0 \\ \sum_{i=1}^{N} \alpha_i = 1 \\ \sum_{i=1}^{N} \beta_i = 1 \end{gathered} \tag{20}$$

Definitions in (20) ensure that $\Delta v_{design} = \Delta v_{bo} + \Delta v_{loss}$ always holds. The optimization routine was then started by solving for L_i and corresponding mass ratios using equation (18) where $v_{f,i}$ is v_{design} given by (19). Using historical data presented in Figure 4 an initial estimate for the structural ratio of the first step step was picked to be $\sigma_1 = 0.06$ and for the second step was $\sigma_2 = 0.08$. Different propulsion systems were characterised by their

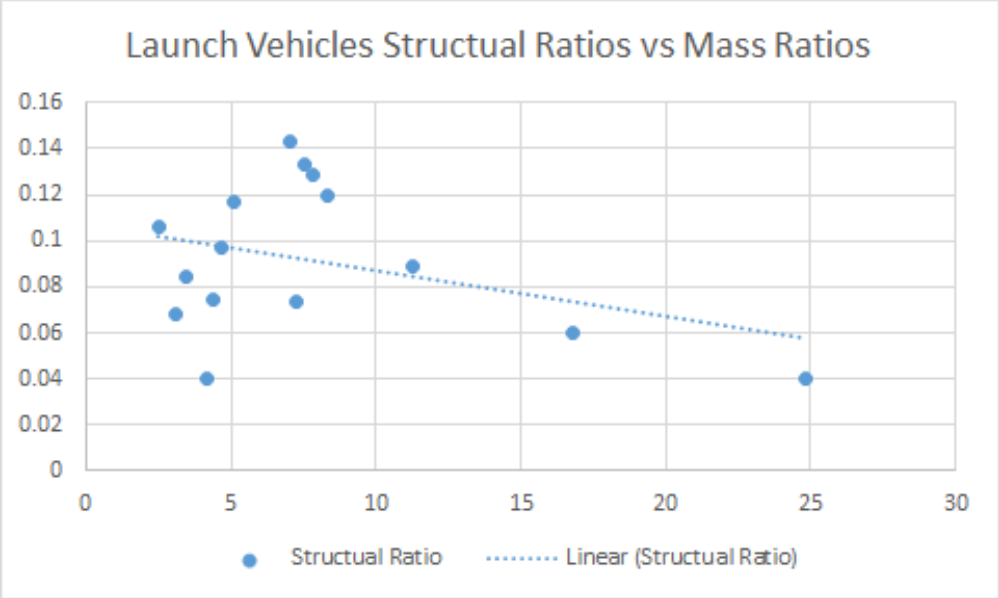

Figure 4: Historical data on structural ratios and mass fraction for two stage launch vehicles.

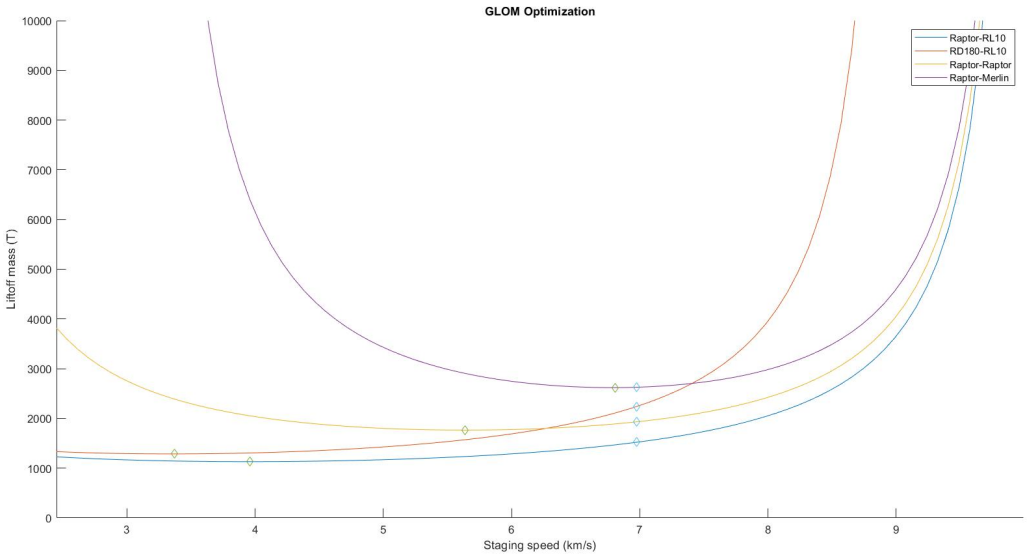

Figure 5: Launch vehicle gross liftoff mass optimization results for different propulsion types.

respective I_{sp} values. However, for the first step the ascent portion the specific impulse was corrected using both sea-level and vacuum characteristics:

$$I_{sp\ eff} = \frac{I_{sp\ SL} + 2I_{sp\ vac}}{3} \tag{21}$$

The results of the gross liftoff mass optimization routine for the launch vehicle are presented in Figure 5. It was assumed that all the aerodynamic losses were absorbed during the operation of the first stage by applying $\beta_1 = 1$ and $beta_2 = 0$ and the allocation of the total

burnout velocity provided by each step α was iteratively solved by cycling through all the options for the first step $\alpha_1 = [0 : 1]$. Thus, the second step had $\alpha_2 = 1 - \alpha_1$ of burnout velocity allocated for each iteration.

Note the marked minimums for the total liftoff mass of each of the propulsion type: using methane (raptor) for the first stage, combined with the $LOxLH_2$ (RL-10) second stage yields the lowest liftoff mass. The preliminary results for the allocation of mass are presented in Table 5

Stage	Mass (T)	Raptor-RL10	RD180-RL10	Raptor-Raptor	Raptor-Merlin
First	Liftoff	1130.7	1286.2	1765.0	2619.6
	Propellant	758.3	833.7	1401.8	2231.8
	Final (b/o)	372.4	452.4	363.2	387.8
	Payload	324.0	399.2	273.7	245.4
Staging Speed (km/s):		3.96	3.37	5.63	6.81
Second	Liftoff	324.0	399.2	273.7	245.4
	Propellant	252.1	321.2	205.8	179.7
	Final (b/o)	71.9	77.9	67.9	65.6
	Payload	50.0	50.0	50.0	50.0

Table 5: Summary of minimum liftoff masses for each propulsion selection.

Since it was assumed that the first stage was going to provide enough velocity to get the vehicle into low-Earth orbit, a higher staging speed of approximately 7 km/s was selected (see Figure 5) and required mass fractions extracted. These sub-optimal values were used for the sizing process and the initial mass values are presented in Table 6. Note, these are not the final masses, but only preliminary estimates which were later adjusted to accommodate the extra weight of the systems and structural components.

Stage	Mass (T)	Raptor-RL10	RD180-RL10	Raptor-Raptor	Raptor-Merlin
First	Liftoff	1521.3	2241.1	1932.0	2627.2
	Propellant	1306.4	1983.0	1659.1	2256.1
	Final (b/o)	214.9	258.1	272.9	371.1
	Payload	131.5	131.5	167.0	227.1
Staging Speed (km/s):		6.98	6.98	6.98	6.98
Second	Liftoff	131.5	131.5	167.0	227.1
	Propellant	75.0	75.0	107.6	162.9
	Final (b/o)	56.5	56.5	59.4	64.1
	Payload	50.0	50.0	50.0	50.0

Table 6: Summary of minimum liftoff masses for each propulsion selection.

Estimated values from Table 6 for Raptor-RL10 proposition combination were used as initial mass requirements for each stage. However, the second stage structural mass was later increased as explained and validated in the following sections.

First Stage Overview

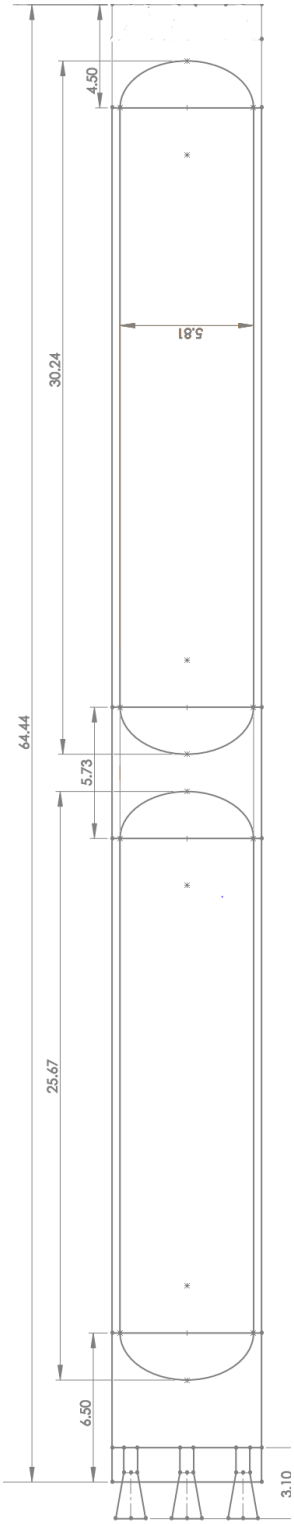

Figure 6: First step primary sizing and overall layout.

Stage overall design layout

Using the mass requirement derived earlier, $LOxLCH_4$ as propellant selection and after accounting for extra losses, the primary dimensions of the first step were derived and are presented in Figure 6. With the outer diameter of 6.44 m, the approximate length of the first step is 65 m.

Mass and volume requirements and upper limits

The maximum liftoff mass of the vehicle is 1550 metric tons, which includes everything: propellant mass, structure, systems and payload. The total usable propellant mass required for the first stage is 1307 metric tons, which does not include the extra propellant required for the engine start-up sequence, boil-off, etc. The maximum payload mass for the first stage is 152 metric tons, which includes the actual payload of 50 metric tons and the transfer vehicle. Required propellant volume for the first stage was estimated to be 1443.0 cubic meters, including the losses. The volume requirement for the liquid methane is 677.6 m^3 and 764.7 m^3 for liquid oxygen.

First Stage Propulsion

Primary Propulsion The primary propulsion for the first stage will consist of nine SpaceX Raptor engines each producing approximately 2.2 MN of thrust at sea level.

Secondary Propulsion None

Mass and volume as to be built

Structural and Payload mass fractions Theoretical structural fraction used in the optimization for the first stage was 0.06. After accounting for materials, propulsion, extra systems and incorporating a safety factor, the actual structural fraction for the first stage is 0.0651. The payload mass fraction, assuming 1550 metric tons launch vehicle and 152 metric tons second stage as payload is 0.0981 and the overall payload ratio, assuming final payload mass of 50 metric tons is 0.0323

Wet and Dry mass Dry mass of the first step, excluding the payload and the second step is estimated to be 91 metric tons. This includes the structural components, unused propellant, systems, pressurizing tanks and liquids not used as propellant.

The mass of the fully loaded first step, without the second step and no payload is 1398 metric tons.

Second Stage Overview

Stage overall design layout

Using the mass requirement for the second stage, LOx LH_2 as propellant selection and after accounting for extra losses, the primary dimensions of the first step were derived and are presented in Figure 7. With the same outer diameter of 6.44 m as the first stage, the approximate length of 30.2 m.

Mass and volume requirements and upper limits

The mass of the entire stage is 152 metric tons, which includes everything: propellant mass, structure, systems and payload. The total usable propellant mass required for the first stage is 75 metric tons, which does not include the extra propellant required for the engine start-up sequence, boil-off, etc. The payload mass for the stage is 50 metric tons, which includes just the mission required payload. Required propellant volume for the stage was estimated to be 202.5 cubic meters, including the losses. The volume requirement for the liquid hydrogen is 154.4 m^3 and 48.1 m^3 for liquid oxygen.

First Stage Propulsion

Primary Propulsion The primary propulsion for the first stage will consist of seven RL-10C engines each producing approximately 102 KN of thrust in vacuum.

Secondary Propulsion There will be onboard RCS systems to assist in course correcting in space.

Mass and volume as to be built

Structural and Payload mass fractions Theoretical structural fraction used in the optimization for the second stage was 0.08. However, after accounting for materials, propulsion, extra systems and incorporating a safety factors, the actual structural fraction for the stage increased to 0.2647. The payload mass fraction, assuming 152 metric tons launch vehicle and final payload mass of 50 metric tons is 0.3289

Wet and Dry mass The second step structural (dry) mass, excluding the payload, is estimated to be 27 metric tons. This includes the structural components, unused propellant, systems, pressurizing tanks and liquids not used as propellant. The dry weight had to be significantly increased from the preliminary calculated values derived from the optimized mass ratios since it was not feasible to include all the structure mass and systems with only 6.5 metric tons of structural mass for everything.

The mass of the fully loaded second step, without payload, is 102 metric tons.

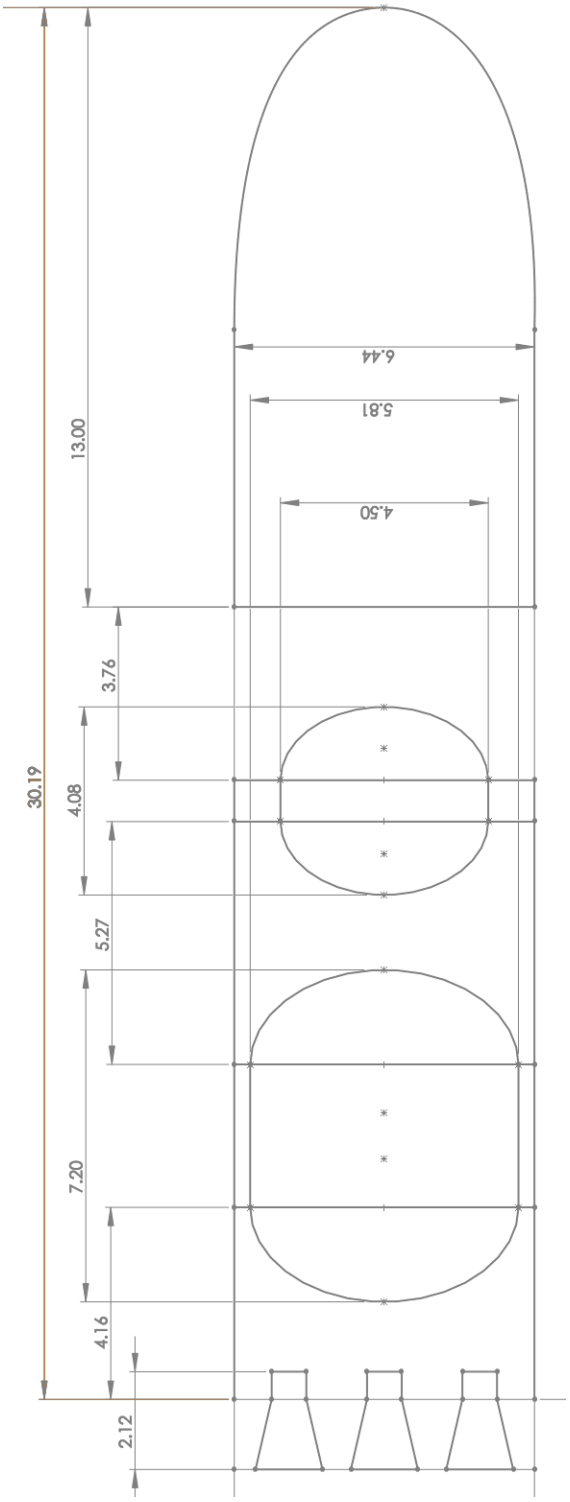

Figure 7: Second Stage primary sizing and overall layout.

Cargo Module

Structural Layout

Since the launch vehicle's second stage is the cargo module, all the structural systems for the cargo module consist of the second stage structural systems described in their respective sections.

Any cargo on board the launch vehicle will need to be properly secured in the the uppermost section to avoid any potential hazards relating to cargo becoming loose during flight, in turn causing damage to both the cargo and the launch vehicle. Regardless of the payload, it will be the responsibility of the organization utilizing Genesis as a payload launch system to provide a secure payload which fits within the dimensional specifications of the cargo hold, not exceeding 50 metric tons. Once the payload is secured to itself, on a titanium pallet or other structural system, the payload structure will be bolted and secured to a flat titanium plate on the bottom of the cargo hold, preventing the payload from causing any damage to the launch vehicle during the launch or maneuvering procedures. A rendering of the structural layout of the 2nd stage payload support structure can be seen in Figure 8.

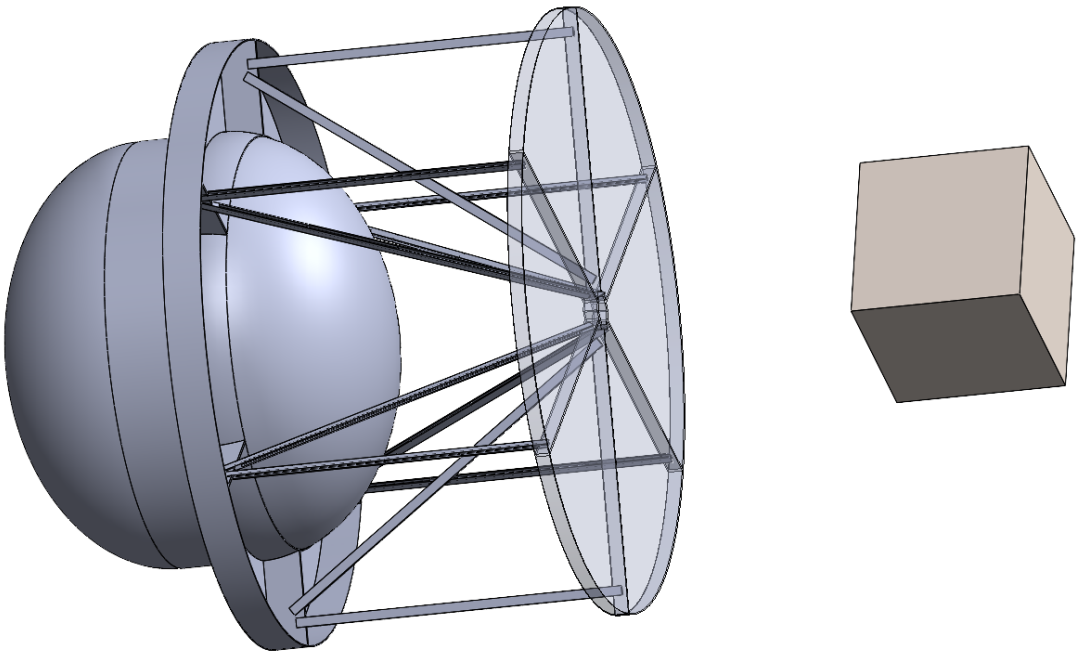

Figure 8: Cargo Support Structure

Primary Propulsion

Since the launch vehicle's second stage is the cargo module, all the primary propulsion systems for the cargo module consist of the second stage primary propulsion systems described in their respective sections.

Secondary Propulsion

Since the launch vehicle's second stage is the cargo module, all the secondary propulsion systems for the cargo module consist of the second stage secondary propulsion systems described in their respective sections.

Docking Method and Equipment

The Genesis will be equipped with an International Docking System Standard (IDSS) compliant mechanism. The mechanism will be able to dock onto other spacecraft such as the International Space Station (ISS) and, ultimately, unload the Genesis's payload. The main docking system will follow an automated rendezvous and docking/capture approach. The docking system will be equipped with numerous sensors ranging from radio frequency (RF) communication-based ranging capabilities to visible/infrared wavelength electromagnetic (EM) spectrum-based sensors. To begin with, RF sensors connected to the Genesis will include clusters of radars, communication equipment, and Global Positioning System (GPS) receivers. This technology will allow the Genesis and its target spacecraft to coordinate relative GPS positioning through GPS state differencing and relative GPS positioning, which have been successful techniques utilized by the Progress, Soyuz, Dragon, and Cygnus vehicles [9]. In addition to RF sensors, the Genesis will also be equipped with visible and infrared wavelength electromagnetic (EM) spectrum-based sensors such as infrared cameras and light, detection and ranging (LIDAR) systems [9]. Similar to the SpaceX's Dragon, the infrared cameras will offer the Genesis the ability to detect and track its target spacecraft such as the ISS without the need of outstanding lighting conditions [9]. Additionally, the infrared camera will provide relative navigation filter range data. In conjunction with on board infrared cameras, the Genesis will utilize LIDARs to provide range and bearing data to the ground control team during docking procedures [9]. By amalgamating these sensor capabilities, the Genesis will employ the infrared cameras to enhance the LIDARs effectiveness during rendezvous and docking operations and be used as FDIR cross-checks [9]. Once the Genesis is attached, the spacecraft will be designed to stay within the attached phase for time frames longer than 6 months, if necessary [10]. In all, the Genesis's main docking system will primarily utilize EM-based sensors to interact with the on board capture system for final attachment through an automated rendezvous and docking approach. Moreover, in order to maximize redundancy, the Genesis will be equipped with RF-based sensors and, if needed, will allow a ground control team to manually takeover the spacecraft in order to allow the attachment of the spacecrafts.

Because of the aforementioned docking requirements, the Genesis will be equipped with a NASA docking system. Currently, the NASA docking system is still under development; however, the technology should be serviceable by the required launch date of the Genesis. A 3D model of the NASA docking system can be seen in Figure 9.

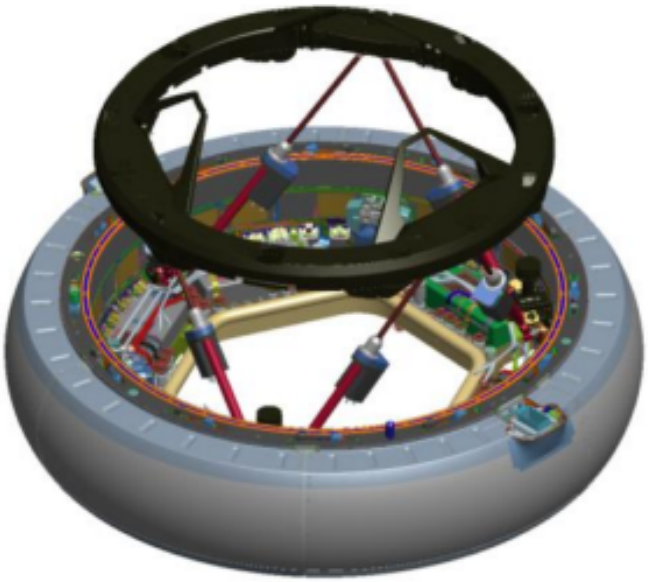

Figure 9: NASA Docking System

The NASA docking system provides an effective solution toward providing redundancy in its capability to be autonomously docked, manually docked, and, if need be, "berthed" to other spacecraft [11]. Additionally, if the Genesis were to ever become a human rated vehicle, the NASA docking system provides an already certified system for the launch vehicle. Thus, the Genesis will employ the NASA docking system as its main capture system.

Main Propulsion (Stage 1)

The first stage of the launch vehicle will be powered by nine SpaceX Raptor engines that produce approximately 2.2 MN of thrust each. This will give the launch vehicle an approximate thrust to weight ratio of 1.25.

Power Cycle (Stage 1)

The Raptor engine hosts a full-flow staged combustion cycle that utilizes its fuel and oxidizer to its fullest potential. The full-flow staged combustion cycle utilizes a state-of-the-art turbopump system that allows lossless propellant usage throughout its flight. This system is extremely versatile as well, being fully restartable after shutdown and hosting a minimum throttle capability of 40 percent. This can allow maximum variability in the total thrust output of the first stage for optimization purposes. A diagram of the Raptor engine combustion cycle can be seen in Figure 10

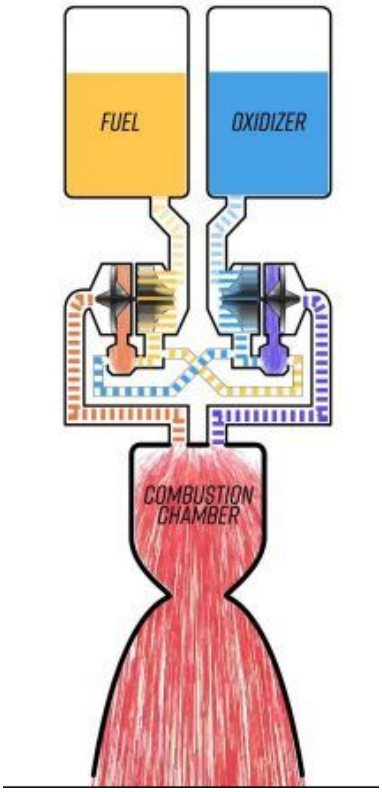

Figure 10: Full Flow Staged Combustion Cycle.

Engine Characteristics

The Raptor engine is fairly efficient for its thrust capabilities, boasting an ISP of 330 seconds. These engines will be burning for approximately 214.5 seconds before the first separation event. The characteristic velocity of the Raptor engine is not charted or released publicly by SpaceX, but the specific exhaust velocity was calculated to be 3237.3 meters per second. The Total impulse for one engine is 471.9 MN-s.

Propellant type

The Raptor engine utilizes liquid methane (CH4) as its fuel and liquid oxygen (LOX) as its oxidizer. It uses an oxidizer to fuel ratio of 3.55 to 1.

Mass and volume of each engine

Each engine weighs approximately 1500 kg. The volume of each engine is approximately 2.5 cubic meters.

Propellant flow Rate

The total propellant flow rate for a Raptor engine is approximately 650 kg/s. This equates to an approximate LOX flow rate of 510 kg/s and an approximate CH4 flow rate of 140 kg/s.

Main Propulsion Feed Assembly

The Raptor engines are fed propellant form the propellant tanks into two independent turbo pumps that increase their pressure significantly before they are sent into the combustion chamber. Each of these turbo pumps is fed a fuel and an oxidizer line. One turbo pump is run fuel rich while the other pump is run oxidizer rich. The resulting exhaust from these turbo pumps is then fed into the combustion chamber where they are remixed and reignited before being converted into thrust. To prevent pogo oscillations within the propellant lines, the oxidizer and fuel lines will have various bellows integrated to assist in damping propellant oscillations. Through testing, the success of this system will be validated. If the bellow system fails to prevent enough pogo oscillation, there is an alternative plan. Before engine ignition, the system may need to be primed with helium to assist in the damping of the pogo oscillations. This was a technique used on the Saturn V. Inside the thrust structure, there will be stacks of solenoids with various propellants and helium running through them to control pressurization of all of the thrust systems. These solenoids will be mounted to pre-allocated fixtures and will be powered by on board electrical signals. This will allow control of every fluid running through the system. Inside each propellant tank will be eight perforated plates arranged in an octagonal shape that will allow propellant to pass through to the feed lines below, but will mitigate sloshing within the tanks during flight. A diagram of the engine feed assembly turbopumps can be seen in Figure 11. An example of slosh baffles and bellows can be seen in Figure 12.

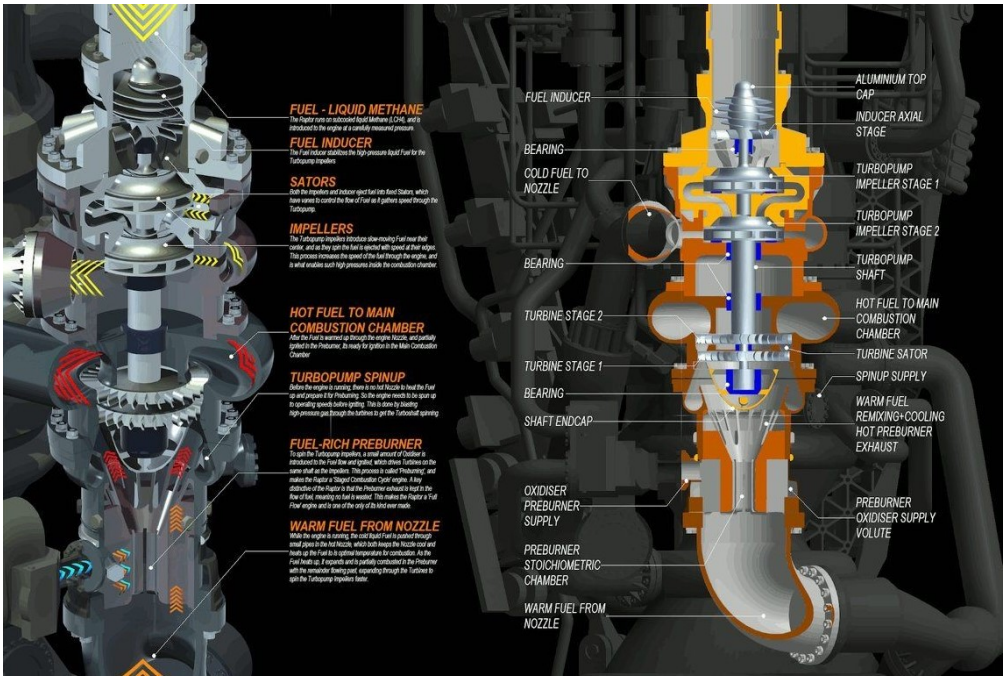

Figure 11: Cross-section view of SpaceX Raptor Propellant Feed.

Figure 12: Example Slosh Baffles (left) and Bellows (right)

Thrust vector control:

Type The Raptor engine will be equipped with dual mounted thrust vectoring systems. The gimballing of these engines will be done by two electrohydraulic actuators per gymballed engine. To successfully command the rocket with this thrust vector control scheme, alternating engines, including the center engine, will be able to be gimballed. This will help

prevent engines from striking each other during heavy or complicated maneuvers. A layout of the nine first stage raptor engines can be seen in Figure 13

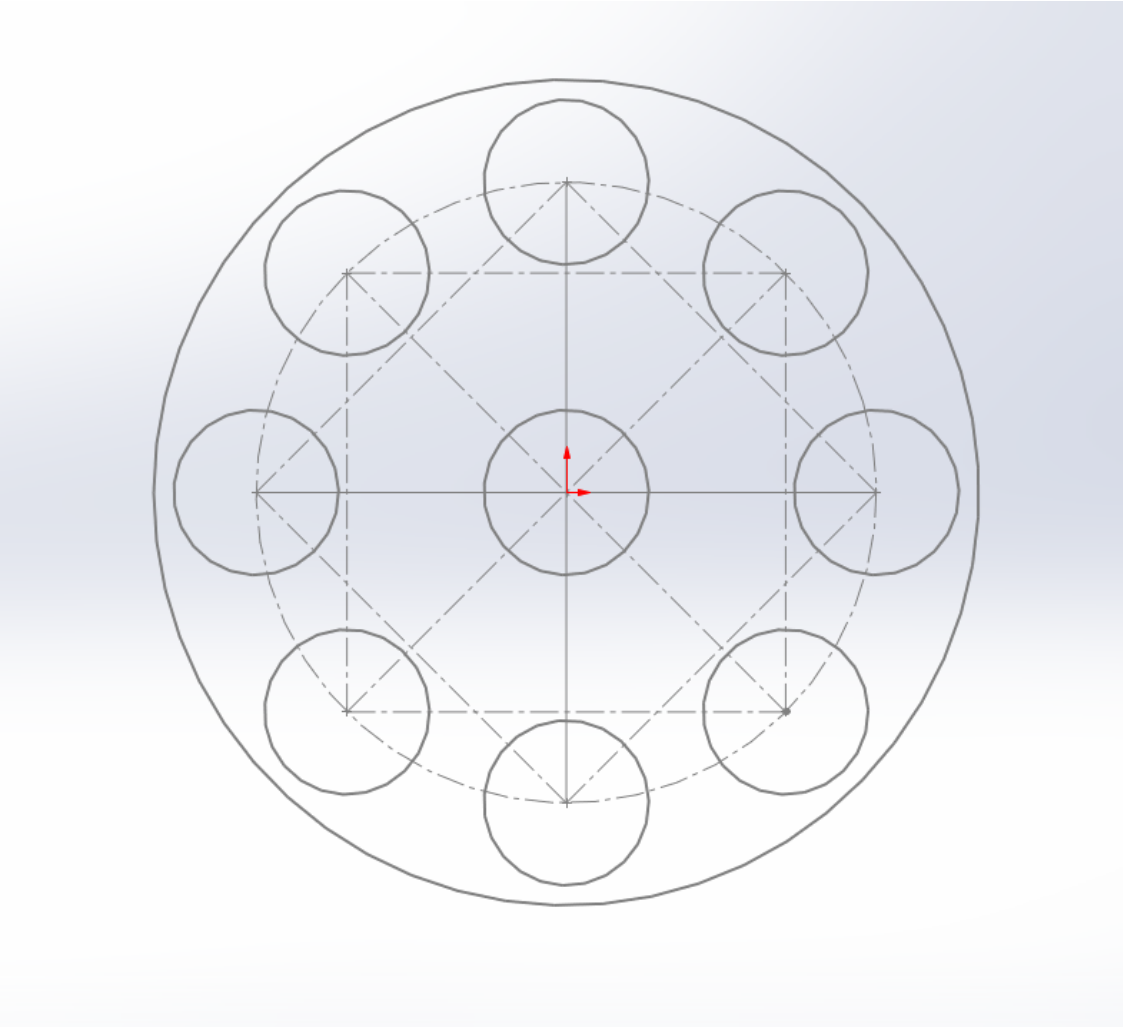

Figure 13: Layout of the Raptor Engines.

Control Requirements The approximate gimballing response rate for these engines will be 10 degrees per second. The maximum deflection of each engine will be approximately 10.5 degrees from vertical.

Effective Deflection Arc Fan The deflection fan of the first stage can be seen in Figure 14.

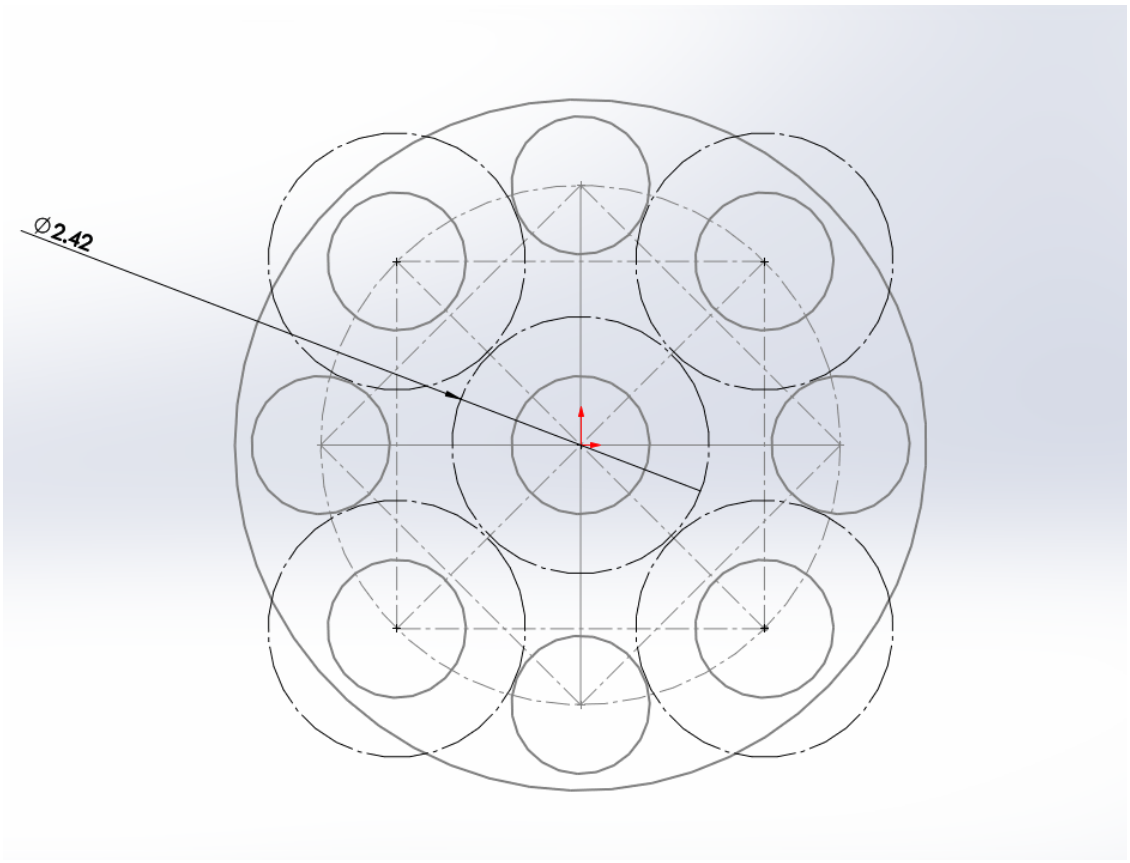

Figure 14: Layout of the Raptor Engines with Maximum Arc Fan.

Applicable trade studies: To adequately select the Genesis-1 propulsion systems, eight engines were analyzed using a trade study. The eight engines analyzed were the Merlin, RL-10, Raptor, BE-4, Vinci, RS-25, F-1, and RD-180. These engines were selected for their applicability in space missions, launch vehicles, and for their historical use, excluding the Raptor. The engines were evaluated on their Technical Readiness Level (TRL), max thrust, ISP, weight, specific exhaust velocity, thrust to weight, propellant density, and availability. By quantifying and evaluating these engine parameters, the first stage propulsion source was determined. Table 7 displays the initial rocket engine values.

Actual Values								
Item	Merlin	RL-10	Raptor	BE-4	Vinci	RS-25	F-1	RD-180
TRL (1-9)	9	9	8	8	7	9	9	9
Max Thrust (N)	8.45E+05	1.02E+05	1.96E+06	2.40E+06	1.80E+05	1.86E+06	6.77E+06	3.83E+06
ISP (sec)	282	450.1	330	350	465	366	263	311
Weight (kg)	470	190.51	1500	3000	550	3177	8400	5,480
Specific Exhaust Velocity	2766.42	4415.48	3237.30	3433.50	4561.65	3590.46	2580.03	3050.91
Thrust/Weight	1797.87	535.40	1306.67	800.00	327.27	585.46	805.95	698.91
Propellant Density (avg) (kg/m^3)	1131.95	372.44	910.61	913.79	369.31	377.89	1116.58	1139.56
Availability	Y	Y	Y	Y	Y	Y	N	Y

Table 7: Propulsion Trade Study with Actual Values.

The values from Table 7 were determined from published data and test results. Though availability may seem as rather odd parameter, its importance is founded upon the fact that

the F-1 engine is no longer in service and the soon to be discontinuation of the RD-180. In order to continue the weighted trade study, the engines were normalized to a single metric so that comparisons can be made. The factors, determining the normalization of these values, were determined on fundamental aspects of rocket propulsion and space vehicle travel. Table 8 shows the normalization values of the engines on a single metric.

Normalization								
Item	1	2	3	4	5	6	7	8
TRL (1-9)	2	3	4	5	6	7	8	9
Max Thrust (N)	<1E+05	<5E+05	<1E+06	<1.9E+06	<2.9E+06	<4E+06	<5E+06	>5E+06
ISP (sec)	>100	>150	>200	>250	>300	>350	>400	>450
Weight (kg)	>8000	>6000	>4000	>3000	>1750	>750	>400	<400
Thrust/Weight	>2E+02	>3E+02	>4E+02	>5E+02	>6E+02	>7E+02	>9E+02	>1E+03
Fuel Density	>200	>300	>400	>500	>600	>700	>900	>1100
Availability	N							Y

Table 8: Propulsion Trade Study with Normalization Values.

By utilizing the real and normalization values, first stage weighting factors were assigned to each propulsion parameter. Table 9 displays the unweighted values of the rocket engines.

Un-Weighted								
Item	Merlin	RL-10	Raptor	BE-4	Vinci	RS-25	F-1	RD-180
TRL (1-9)	8	8	7	7	6	8	8	8
Max Thrust (N)	3	2	5	5	2	4	8	6
ISP (sec)	4	8	5	5	8	6	4	5
Weight (kg)	7	8	6	4	7	4	1	3
Thrust/Weight	8	4	8	6	2	4	6	6
Fuel Density	8	2	7	7	2	2	8	8
Availability	9	9	9	9	9	9	1	9

Table 9: Propulsion Trade Study with Unweighted Values.

Table 10 shows a system of weighted factors based on the total importance of each variable to the first stage of the Genesis. Max thrust was given the highest weight due to the importance of the engine being selected as the first stage of the launch vehicle. Without a significant thrust, the amount of engines necessitated to power the launch vehicle will drive unreasonable amount of weight and overall cost of the system. Availability plays a critical role in the selection of the engine sources in that a few of the engine evaluated may not be available during the future launch date. Concerning the lowest weighted fields, ISP and thrust to weight parameters do not provide an extensive driver in the decision of the first stage engine choice.

Weighing Factors (1st Stage)		
Item	Weight	Reason
TRL (1-9)	6	Engine must be proven reliable before launch date
Max Thrust (N)	8	Vehicle needs significant thrust for 1st stage
ISP (sec)	3	Efficiency is not necessarily the highest priority for first stages
Weight (kg)	4	Lowering weight aids in efficiency.
Thrust/Weight	3	Higher thrust to weight lead to more thrust for structure and payload
Fuel Density	5	Higher fuel densities lead to smaller propellant structures and less mass
Availability	7	Engine must be available for use

Table 10: Propulsion Trade Study with Weighting Factors for 1st Stage.

After Table 10 was completed, the values from the table could be multiplied by the unweighted values from Table 8 to generate the final table, Table 11. Table 11 shows the weighted values for each of the 8 different types of engines. After the table has been weighted, the columns need to be totaled, and the column with the greatest value is the value that will be chosen as the ideal engine. After summing up the columns, the Raptor engine was determined to be ideal first stage engine for the payload mission. Table 11 displays the weighted values of the engines.

Weighted (1st Stage)								
Item	Merlin	RL-10	Raptor	BE-4	Vinci	RS-25	F-1	RD-180
TRL (1-9)	48	48	42	42	36	48	48	48
Max Thrust (N)	24	16	40	40	16	32	64	48
ISP (sec)	12	24	15	15	24	18	12	15
Weight (kg)	28	32	24	16	28	16	4	12
Thrust/Weight	24	12	24	18	6	12	18	18
Fuel Density	40	10	35	35	10	10	40	40
Availability	63	63	63	63	63	63	7	63
Totals	239	205	243	229	183	199	193	244

Table 11: Propulsion Trade Study with Weighted Values for 1st Stage.

Table 11 reveals that why Raptor engine was chosen as the propulsion source for the first stage of the Genesis. By applying the normalization values and weighting factors, the most mission ready of the engines has been deduced with the trade study. It is important to note that the RD-180 engine had overall totals that were very close to the Raptor; however,the RD-180 engine will not be available due to its discontinuation. All in all, the weighted trade study and computational analysis that was applied to the first stage engine was successful in determining the most applicable engine in accomplishing the mission. This success was further validated by the staging optimization analysis performed and demonstrated in Figure 5

Main Propulsion (Stage 2)

The second stage of the launch vehicle will be powered by 7 Pratt Wittney RL-10C engines. RL-10C engines that produce approximately 102 kN of thrust each. This will give the second stage of the launch vehicle an approximate initial acceleration of 5 m/s.

Power Cycle

The RL-10 engine utilizes an expander combustion cycle that utilizes its fuel and oxidizer very efficiently. The expander cycle supplies itself with propellant in a somewhat unique way, as it utilizes the natural thermodynamic expansion of gasses to spin its turbo pumps. Expander cycle engines need to utilize very specific fuels in order to produce the effect necessary for turbo pump operation. The supercooled fuel is routed around the nozzle of the rocket engine to heat and expand it rapidly through a turbine. This turbine is connected to the driveshafts of the turbopumps. After being run through the turbine, the fuel is then routed into the combustion chamber for combustion. These engines are very reliable because there is very little damage to the turbines during operation. This is because the fuel temperature is considerably lower during the expansion. This cycle has also proven to be extremely resistant to fuel imperfections. A diagram of the RL-10 engine combustion cycle can be seen in Figure 15

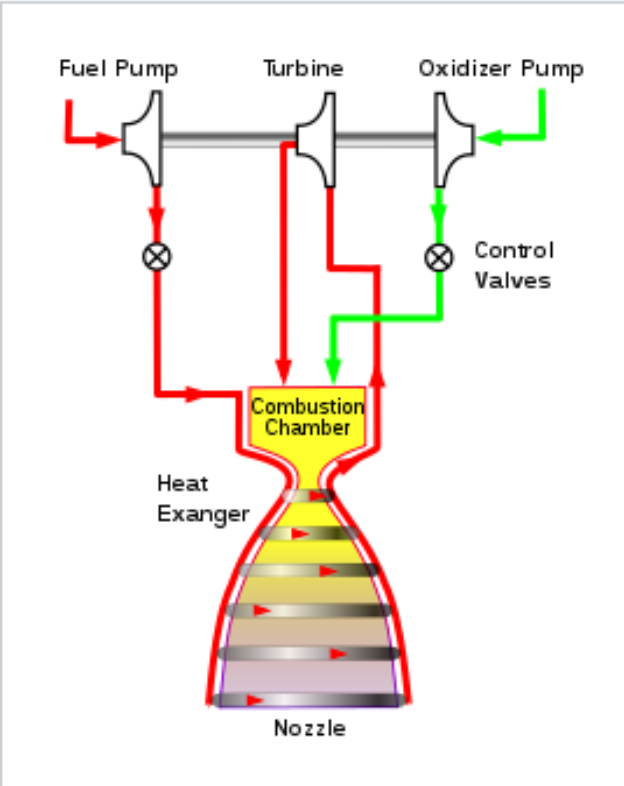

Figure 15: Expander Combustion Cycle.

Engine Characteristics

The RL-10 engine is very efficient for its thrust capabilities, boasting an ISP of 450 seconds. These engines will be burning for approximately 12 minutes before the mission is completed. The characteristic velocity of the RL-10 engine is not charted or released publicly, but the specific exhaust velocity was calculated to be 4415.5 meters per second. The total impulse for one engine is 56.15 MN-s.

Propellant type

The RL-10 engine utilizes liquid hydrogen (LH2) as its fuel and liquid oxygen (LOX) as its oxidizer. It uses an oxidizer to fuel ratio of 5.88 to 1.

Mass and volume of each engine

Each engine weighs approximately 190 kg. The volume of each engine is approximately 2.2 cubic meters.

Propellant Flow Rate

The total propellant flow rate for a RL-10 engine is approximately 16 kg/s. This equates to an approximate LOX flow rate of 13.7kg/s and an approximate CH4 flow rate of 2.32 kg/s.

Main Propulsion Feed Assembly

The RL-10 engines are fed propellant from the propellant tanks into the pump assemblies before being sent into the combustion. To prevent pogo oscillations within the propellant lines, the oxidizer and fuel lines will have various bellows integrated to assist in damping propellant oscillations just like the Raptors. Through testing, the success of this system will be validated. If the bellow system fails to prevent enough pogo oscillation, the alternative plan discussed before will be implemented. Since the acceleration of the upper stage will be minimal, it is not anticipated that pogo will be a large issue. Inside the thrust structure, there will be stacks of solenoids with various propellants and helium running through them to control pressurization of all of the thrust systems. These solenoids will be mounted to pre-allocated fixtures and will be powered by on board electrical signals. This will allow control of every fluid running through the system. Inside each propellant tank will be eight perforated plates arranged in an octagonal shape that will allow propellant to pass through to the feed lines below, but will mitigate sloshing within the tanks during flight.A diagram of the engine feed assembly turbopumps can be seen in Figure 16. An example of slosh baffles and bellows can be seen in Figure12.

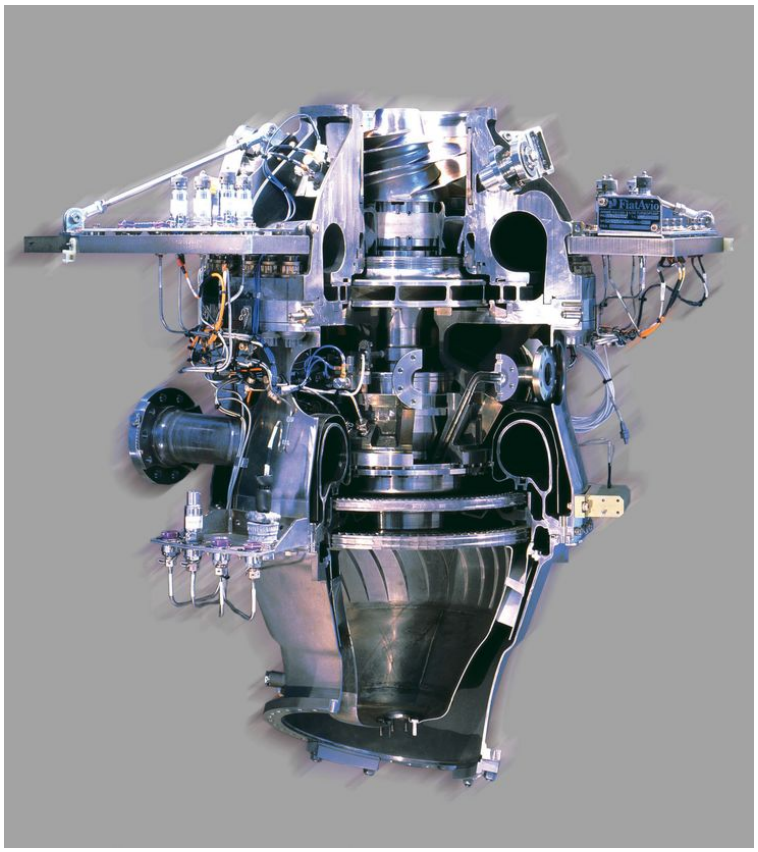

Figure 16: Cross-section view of Ariane 5 first stage LOx turbopump used for RL10 engines.

Thrust vector control:

Type The RL-10 engine will be equipped with dual mounted thrust vectoring systems. The gimballing of these engines will be done by two electrohydraulic actuators per gymballed engine. To successfully command the rocket with this thrust vector control scheme, all engines will be able to be gimballed. This will allow the vehicle to be able to perform strong

maneuvers with its limited thrust per engine. A layout of the nine second stage RL-10 engines can be seen in Figure 17.

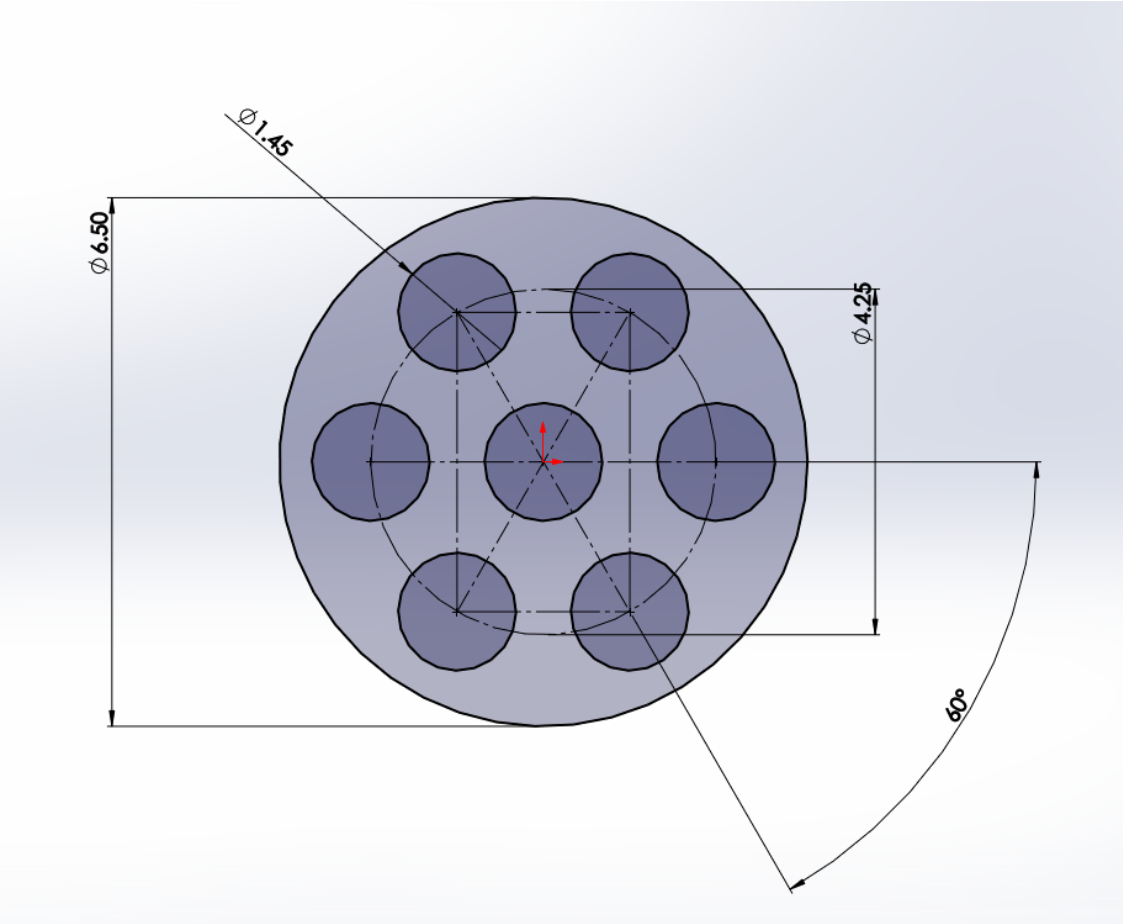

Figure 17: Layout of the RL10 Engines.

Requirements (response rate (deg./sec.), maximum-commanded deflection (deg.)) The approximate gimballing response rate for these engines will be 10 degrees per second. The maximum deflection of each engine will be approximately 8 degrees from vertical.

Effective deflection arc fan The deflection fan of the first stage can be seen in Figure 18.

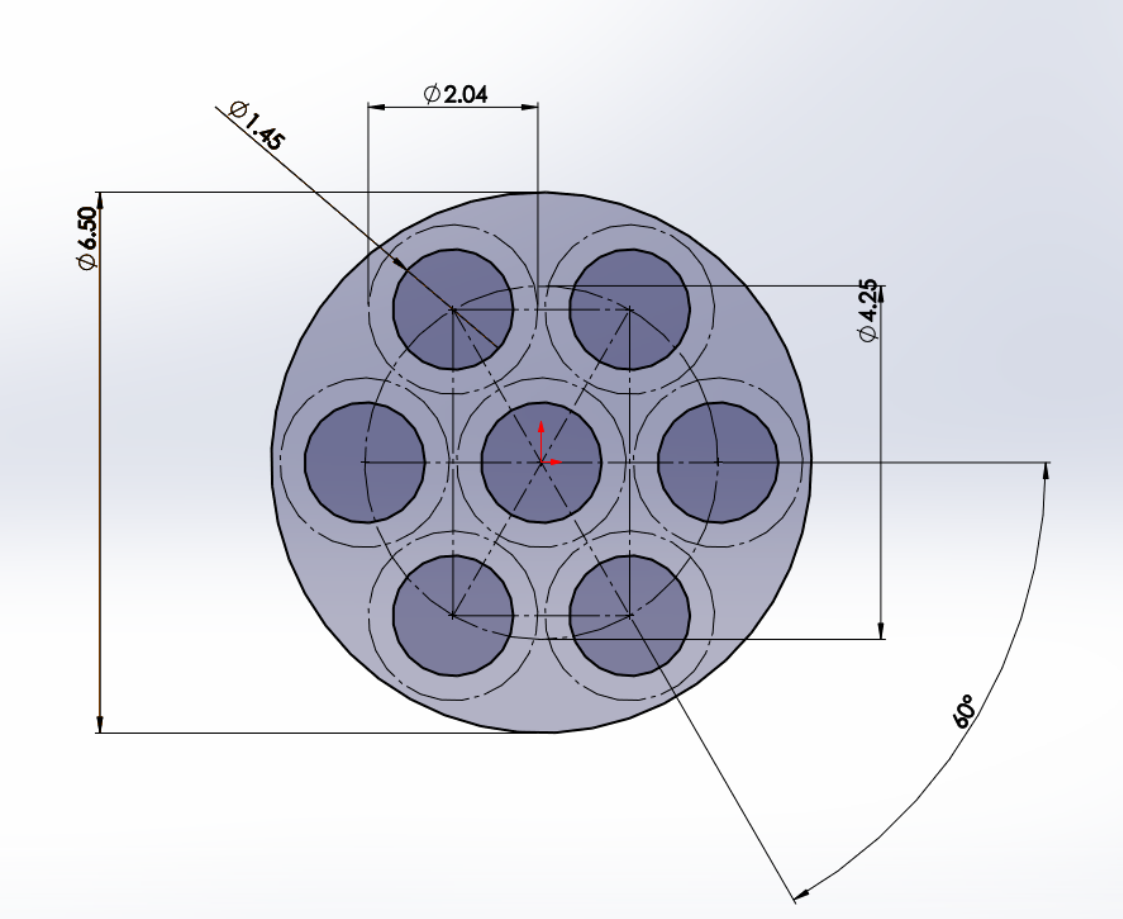

Figure 18: Layout of the RL10 Engines with Maximum Arc Fan.

Applicable trade studies Similar to the first stage, the second stage propulsion source must also be determined with a weighted trade study. The engines selected for the second stage trade study are the same engine evaluated within the first stage. The key difference between studies is the revaluation of the weighting factors. This was done in order to assign more applicable measures of the parameters selected that were suitable for a second stage of a launch vehicle.

As mentioned before, the only difference between the second stage and first stage trade studies were the values of the weighting factors for each propulsion parameter. Thus, Table 12 displays the new weighting factors.

Weighing Factors (2nd Stage)		
Item	Weight	Reason
TRL (1-9)	6	Engine must be proven reliable before launch date
Max Thrust (N)	3	Vehicle does not need as much thrust for 2nd stage
ISP (sec)	7	Efficiency is quite important for upper stage engines
Weight (kg)	6	Lowering weight lowers fuel consumption
Thrust/Weight	2	Thrust to weight is not too important when most of the mass is the payload
Fuel Density	3	Higher fuel densities lead to smaller propellant structures and less mass
Availability	7	Engine must be available for use

Table 12: Propulsion Trade Study with Weighted Factors for 2nd Stage.

The key differences between the weighting factors of the second and first stage were the switch in importance of max thrust and ISP. Due to the change in role, ISP and weight played a more critical role in the decision process of the second stage. Continuing the study, the same process that was done for the first stage trade study was applied to the second stage which resulted in Table 13 showing the weighted values for each of the 8 different types of engines. After summing up the columns, the RL-10 engine was determined to be ideal first stage engine for the payload mission. Table 11 displays the weighted values of the engines.

Weighted (2nd Stage)								
Item	Merlin	RL-10	Raptor	BE-4	Vinci	RS-25	F-1	RD-180
TRL (1-9)	48	48	42	42	36	48	48	48
Max Thrust (N)	9	6	15	15	6	12	24	18
ISP (sec)	28	56	35	35	56	42	28	35
Weight (kg)	42	48	36	24	42	24	6	18
Thrust/Weight	16	8	16	12	4	8	12	12
Fuel Density	24	6	21	21	6	6	24	24
Availability	63	63	63	63	63	63	7	63
Totals	230	235	228	212	213	203	149	218

Table 13: Propulsion Trade Study with Weighted Values for 2nd Stage.

Table 13 reveals that the RL-10 engine should be chosen as the propulsion source for the second stage of the Genesis. This decision makes historical sense because the RL-10 engine is very popular upper stage engine that currently power the Atlas V and Ariane 5 launch vehicles. It is important to note that the Merlin engine had overall totals that were very close to the RL-10. Because of this, if the RL-10 engine were to fail in actual application, the Merlin engine should be able to replace the RL-10. All in all, the weighted trade study that was applied to the second stage engine was successful in determining the most applicable engine for the role.

Secondary/Auxiliary Propulsion

Additional Solid / Liquid Propulsion

The design of this launch vehicle requires no additional solid or liquid rockets other than the main lower and upper stage propulsion systems.

Attitude Control Systems

The gimbaling of the main propulsion systems allow for attitude control of the vehicle, but their utilization is limited to when the engines are active. Additional auxiliary propulsion will be included in the case that the vehicle requires attitude adjustment outside of gimbaling, minor retrograde maneuvers, or finer trajectory adjustments for processes such as docking maneuvers. These auxiliary propulsion systems will allow for slight adjustments in the vehicles attitude and trajectory without full reignition of the main propulsion systems, and will be based on the design of the systems used on the Space Shuttle Orbiter and the 11D428A-16.

Thrusters

Two clusters of four sets of thrusters will be placed along the hull, hidden beneath to reduce drag. Their placement will allow for full 6 degree-of-freedom control of the vehicle's upper stage after engine cutoff. Below in Figure 19 is a diagram of the RCS thrusters used on the Gemini Spacecraft, which bear striking resemblance to those which will be utilized for this launch vehicle [12].

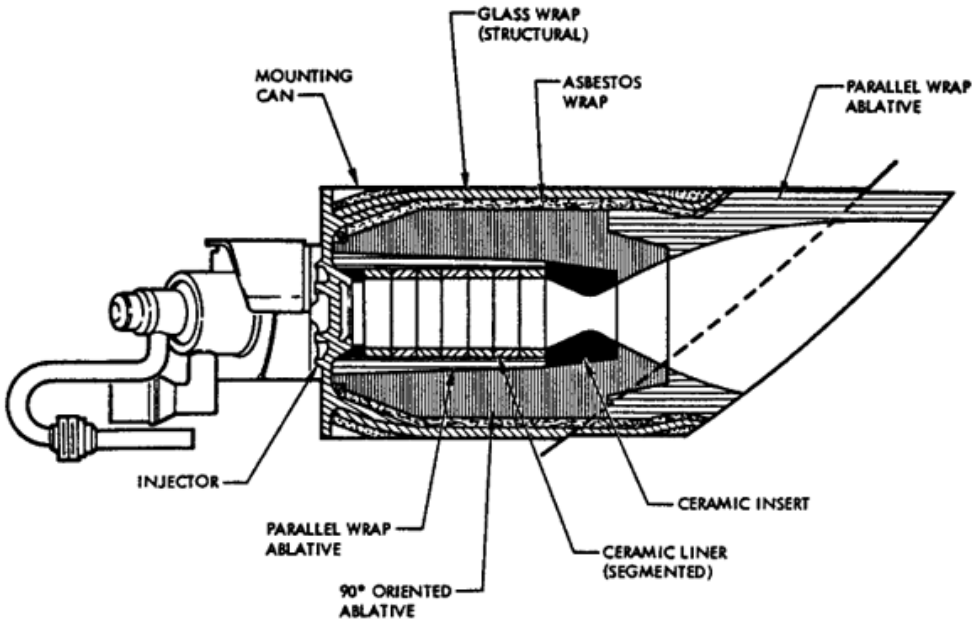

Figure 19: Gemini Spacecraft GRCS Thruster

Each of these thrusters will provide 130.5 Newtons of vacuum thrust, and have a nominal chamber pressure of 0.88 MPa. These thrusters have been heavily utilized and modified by the industry since their conception and are used onboard the Salyut 1, and have been rigorously tested. Additionally, these thrusters are currently used on the Zvezda module on the ISS as reaction control thrusters [13].

Propellant

The RCS thrusters on this launch vehicle will utilize a fuel composition and mixture identical to those used on the Space Shuttle Orbiter, comprised of a 1.85:1 nitrogen tetroxide oxidizer and monomethyl hydrazine fuel mixture [14]. As these engines utilize a different fuel than the main upper stage propulsion system, they will have their own propellant tanks to pull from. Additionally, reflecting the RCS systems aboard the Space Shuttle orbiter, each set of thrusters will have two spherical titanium tanks, one for fuel and one for oxidizer, of a 1m outer diameter, and 1cm thickness [14]. Two pairs of these tanks will be mounted on the inside of the skin with a support structure, one pair for each cluster of RCS thrusters. The auxiliary propulsion systems will be fed by these tanks and activated individually as needed either by ground station control inputs or automatic stability control inputs.

Configuration

Two sets of attitude control clusters will be placed above and below the oxidizer tank in the upper stage. The configuration of one attitude control cluster can be seen in Figure 20, where each blue triangle represents an individual thruster, and the outer circumference represents the launch vehicle's skin. Having two clusters of thrusters allows the vehicle to more effectively pitch and yaw, rather than limiting the influence to lateral motion.

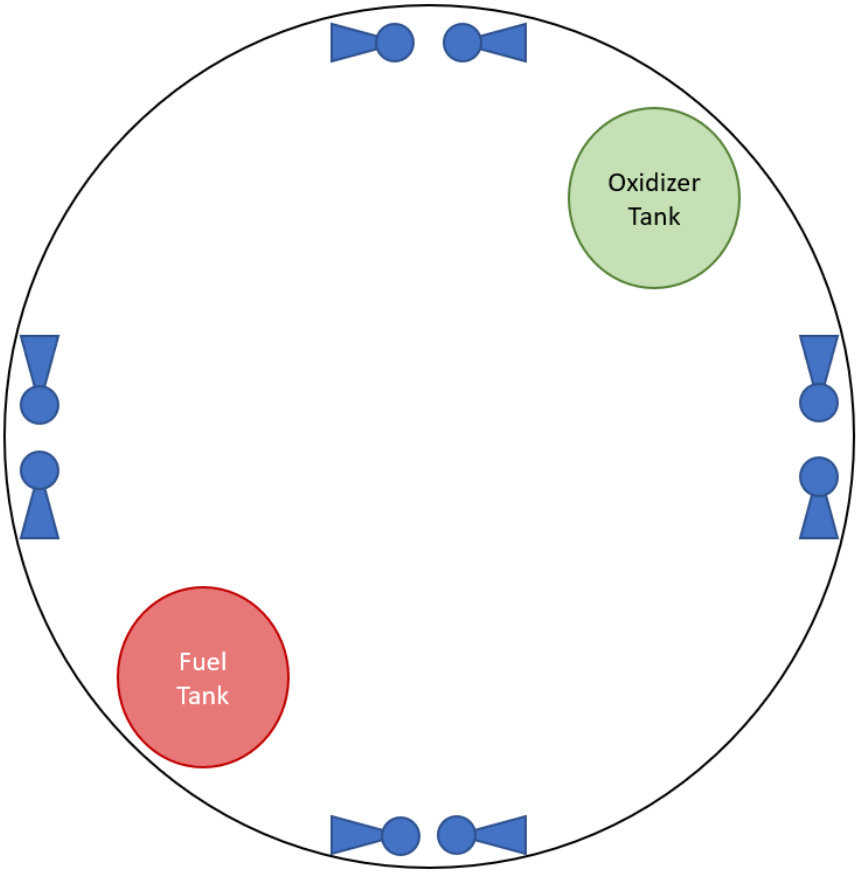

Figure 20: Configuration of Attitude Control Cluster

The total wet mass of the entire attitude control system including fuel and oxidizer is 2.24 Metric Tonnes.

Ullage Motors

Genesis-1 is designed to handle second stage shutoff and re-ignition. Therefore, four cold gas thrusters will be integrated for ullage handling. These cold gas thrusters will be fed gaseous LH2 from the second stage LH2 tank. In the event of the propellants creeping away from the feed area at the bottom of the tanks due to a zero-g environment before engine re-ignition, these motors will activate, propelling the structure of the upper stage forward. This brief acceleration will cause the propellant to resettle in a location conducive to engine re-ignition.

Liquid Main Engine Control

Main engine controls of the Genesis consist of nine Raptor engines and seven RL10C-1 engines all controlled by a Full Authority Digital Engine Control (FADEC) system. The FADEC will be enhanced with redundant fault tolerant avionic systems such as a FTINU system developed by Honeywell for the Atlas V [15]. With enhanced guidance analysis, the Genesis will be able to mitigate vehicle dispersions and guidance hardware and software errors while increasing total system injection accuracy. Thus, if any trajectory changes are necessary, the FADEC will be able to effectively communicate with Electronic Control Units (ECU) connected to the engines and provide throttle and gimbal adjustments. The figure below displays the Atlas V fault tolerant avionics system that the Genesis will be closely based with obvious changes concerning sensor, battery, and engine decisions.

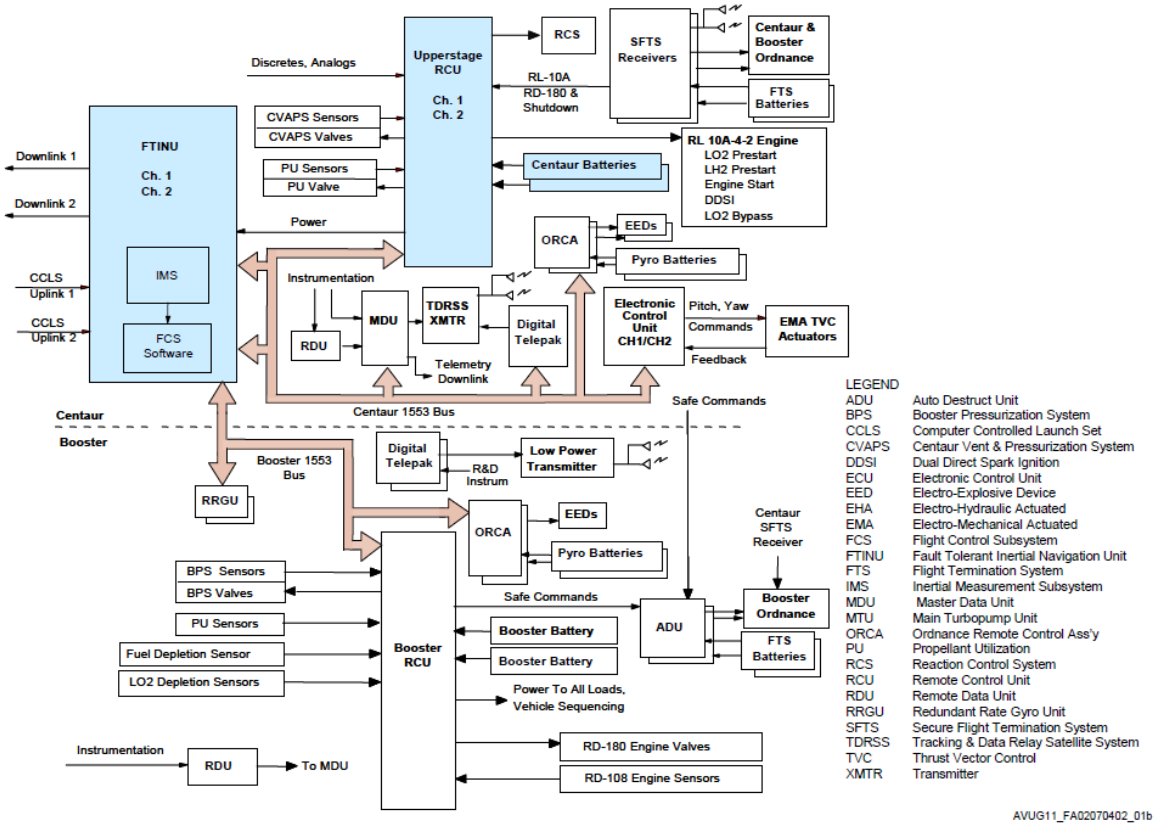

Figure 21: Fault Tolerant Avionics System

Stage 2

For the second stage of the Genesis, seven RL10C-1 engines were selected to power the launch vehicle. The RL10C-1 engine is a gimbaled, turbopump-fed, and regeneratively cooled engine. The optimal mixture ratio for the RL10 engines is 5.88:1 oxidizer to fuel [16]. Additionally, the engines are capable of deep-throttling from 104 percent to 5.9 percent of rated power. The Thrust Vector Control system of the second stage of the Genesis consists of an ECU and two electrohydraulic actuators of the engine. In order to provide pitch, yaw, and roll control of the second stage, a Reaction Control System (RCS) will be added to provide these flight parameters during the coast phase and during the coast up to main engine start [15]. Moreover, it will provide propellant settling. The RCS will be located on the aft bulkhead of the second stage of the Genesis. Additionally, thrust vector control

actuator supports will be located on the propellant tank aft bulkhead as well. If the launch vehicle were to become unstable, the FADEC will gimbal the rocket nozzles by offsetting the thrust direction from the centered position in order to adjust the vehicle back onto a level trajectory. The thrust vector corrections will be sent to the ECUs of each actuator from the FADEC to apply these adjustments. Moreover, the FADEC will communicate with the RCS and apply necessary throttling increases or reductions pending on the Guidance Navigation and Control (GNC) main flight system's trajectory course [15]. Thus, in conjunction with the gimbaling systems, FADEC, ECUs, and actuators, the second stage of the Genesis will be able to boost the payload into the correct flight trajectory while maintaining vehicular stability. The figure below displays the RL10 configuration propulsion system utilized on the second stage Centaur of the Atlas V that will be incorporated within the Genesis.

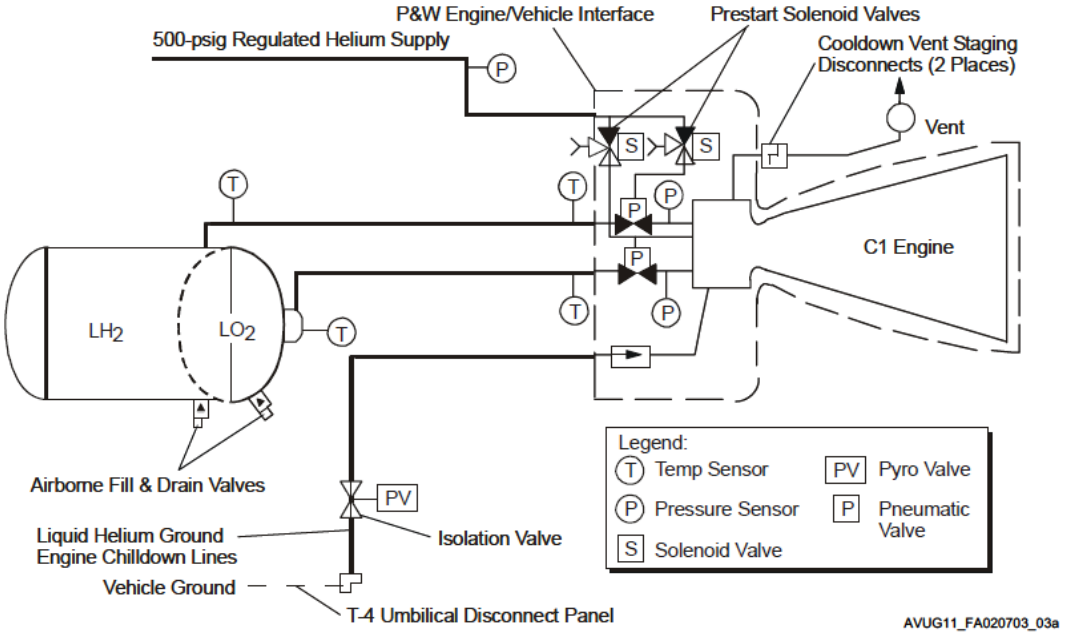

Figure 22: Atlas V Centaur Second Stage Propulsion System

Stage 1

For the first stage of the Genesis, nine Raptor engines were selected to power the launch vehicle. The Raptor engine has throttle capability ranging from 40 to 100 percent with an optimal mixture oxidizer to fuel ratio of 3.55:1 [17]. The continuous throttle capability of the Raptor engine provides an advantage for the first stage of the Genesis by allowing stable control of the launch vehicle over LV to SC environments. The Thrust Vector Control system of the Raptor engines will each consist of dual ECUs with dual electrohydraulic actuators. In order to account for trajectory adjustments, the FADEC will signal the ECUs of the Raptor engines to gimbal to account for ascent attitude pitch, yaw, and roll control. Moreover, the FADEC has the ability to authorize throttling of the Raptor engines to account for changes in thrust to maintain vehicle trajectory to enter orbit in accordance with the GNC system of the Genesis. Programmed thrust profiles within the flight system of the Genesis will allow the launch vehicle to throttle the Raptor engines back in order to minimize vehicle loads during high dynamic pressure periods of flight and peak transonic loads [15]. The importance of this throttling capability is that the programmed thrust profiles will maximize performance of the engine system by allowing high sustained acceleration during flight.

Solid Rocket Motor Thrust Termination Approach (if applicable)

For the Blue Oregano launch vehicle, Genesis 1, it was determined that no solid rocket motors would be required. As such, this page is purposely left blank.

Liquid Propellant Tank Pressurization

Type

The liquid propellant tanks will be fed helium gas through their top surface to allow for pressure control as the propellant tanks are expelled. This helium will be located at various points throughout the vehicle for ease of distribution. The tanks will be pressurized initially to 45 psi.

Pressurant

The pressurant used for each of these tanks is helium, and the maximum assumed pressure for all non helium tanks is 45 psi or 310 kPa. Helium was chosen because of its historical proof from the space shuttle and many other vehicles showing it to be relatively easy to handle and store.[18] The helium tanks are placed strategically to allow for a top fed pressure control as it has been successful in reducing slosh in the past.

Structural

Overall Design

The overall design of the launch vehicle follows a stacked orientation with the second stage located forward of the first stage. There will be four main propellant tanks that fill the bulk of the structure. The skins will be semi-structural and have an orthogrid design from the bottom to the top of the rocket. Inside the skins, there will be a thrust structure for both stages, inter-tank support structures, mid-tank lateral stiffeners, various inter-tank ribs and stringers, mount fixtures for embedded systems and helium supply, and a structural support for the payload. A labeled diagram of the launch vehicle configuration can be seen in Figure 23

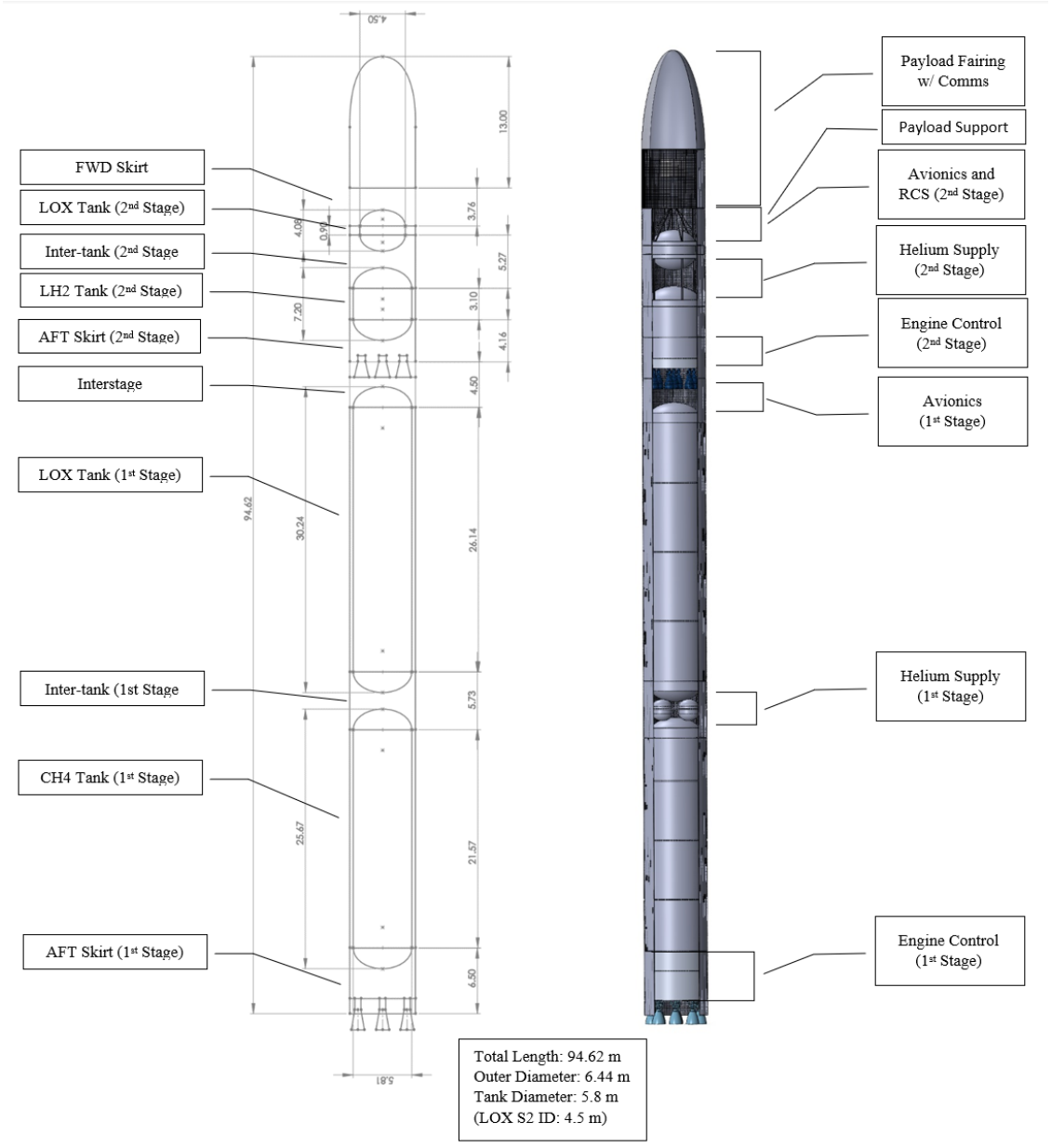

Figure 23: Rocket Configuration

Center of Gravity

Measured from the tip of the payload fairing, the center of gravity for the launch vehicle ranges from 51.12 meters to 42.12 meters wet to dry.

Stability Margin

The launch vehicle's stability was determined for when the vehicle is both wet and dry. The center of pressure of the launch vehicle was determined to be located at 47.43 meters from the tip of the nosecone. Thus, when wet, the difference between CP and CG of the launch vehicle measured from the tip is -3.69 meters, and 5.31 meters once dry. Thus, the rocket will gradually become more stable as it burns off propellant.

Structural Breakdown

The launch vehicle can be broken down into three sections per stage, one being the thrust structure, another being the inner structure support, and the last being the outer structure skins. The first stage thrust structure can be broken into two halves. The lower half is responsible for mounting the engines, main engine control equipment, and providing structural support. The top half is responsible for taking the thrust and transmitting it to both the skins and the lower ellipsoid end cap of the liquid methane tank. A rendering of this structure can be seen in Figure 24.

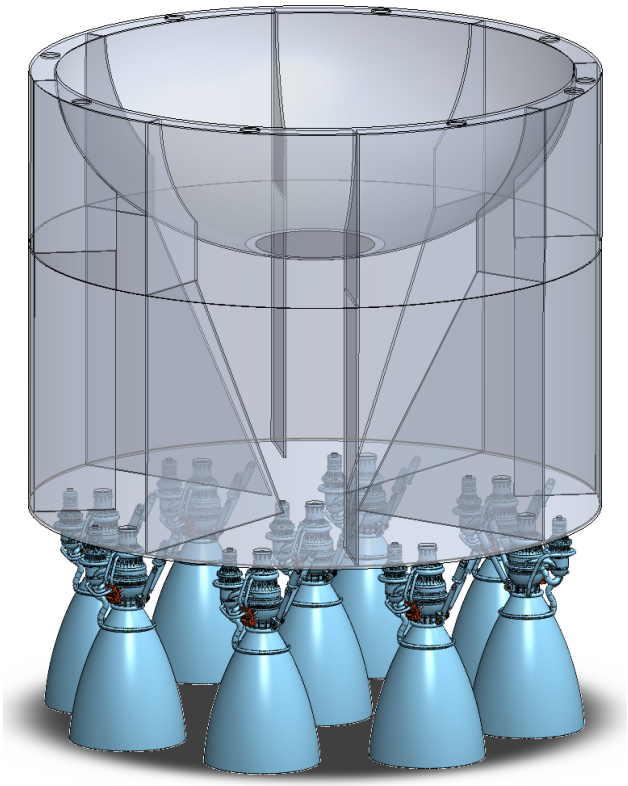

Figure 24: First Stage Thrust Structure

This structure was designed to allow CH4 fuel lines to be run directly from the tank above it and to allow the oxidizer lines for all nine engines to be run through ports in the top side of the structure as they feed down the side of the CH4 tank from the LOX tank.

The internal structure of the 1st stage forward of the thrust structure will be responsible for centering the fuel tanks and mounting them within the skin. It will also allow housing for various internal components, as well as facilitate the routing of electrical and fluid systems throughout the vehicle. This structure is composed of various ring-like ribs that are placed periodically along the tanks. At the top and bottom of each tank are inter-tank supports that provide a more rigid structure at key points within the vehicle. These points of interest will house larger components such as the helium distribution network for tank pressurization. These areas are also thickened to provide the opportunity for auxiliary inter-tank support structures to be implemented in the event the vehicle structures do not hold up as expected. This stage 1 internal structure can be seen in Figure 25

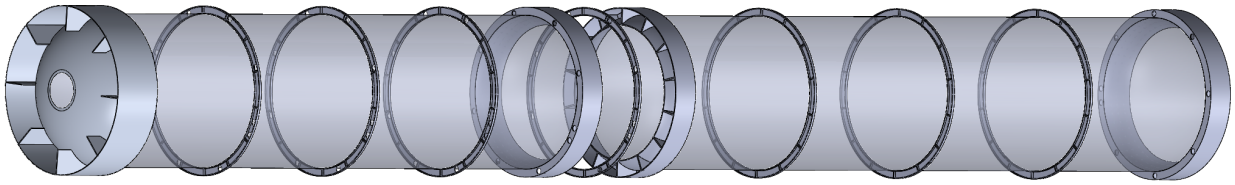

Figure 25: Internal Structure of the First Stage

It can be noted that there is an extra support ring placed between the CH4 and LOX tanks. This is an addition made to provide an easier attachment of helium tanks to the vehicle skins. Above the LOX tank, the support structure and skin around the inter stage is designed to accommodate ample sensors and avionics controllers, as well as the flight termination system.

The second stage thrust structure is smaller than the first, but can also be broken into two halves. The lower half is responsible for mounting the engines, main engine control equipment, and providing structural support. The top half is responsible for taking the thrust and transmitting it to both the skins and the lower ellipsoid end cap of the liquid hydrogen tank. A rendering of this structure can be seen in Figure 26.

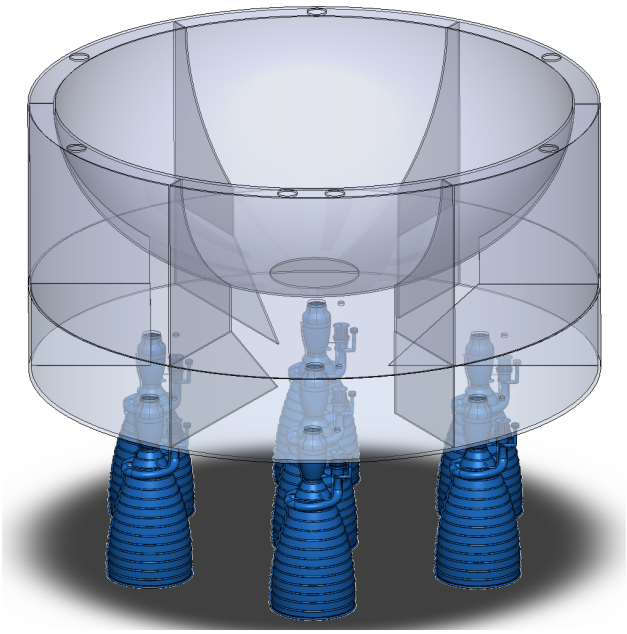

Figure 26: Second Stage Thrust Structure

This structure was designed to allow LH2 fuel lines to be run directly from the tank above it, and to allow the oxidizer lines for all seven engines to be run through ports in the top side of the structure as they feed down the side of the CH4 tank from the LOX tank.

The internal structure of the 2nd stage forward of the thrust structure will be responsible for centering the fuel tanks and mounting them within the skin. It will also allow housing for various internal components, as well as facilitate the routing of electrical and fluid systems throughout the vehicle. This structure is composed of various struts designed to transmit the thrust load through he system evenly. At the top and bottom of each tank are inter-tank supports that provide a more rigid structure at key points within the vehicle. Much like the first stage, these points of interest will house larger components such as the helium distribution network for tank pressurization. This stage 2 internal structure can be seen in Figure 27

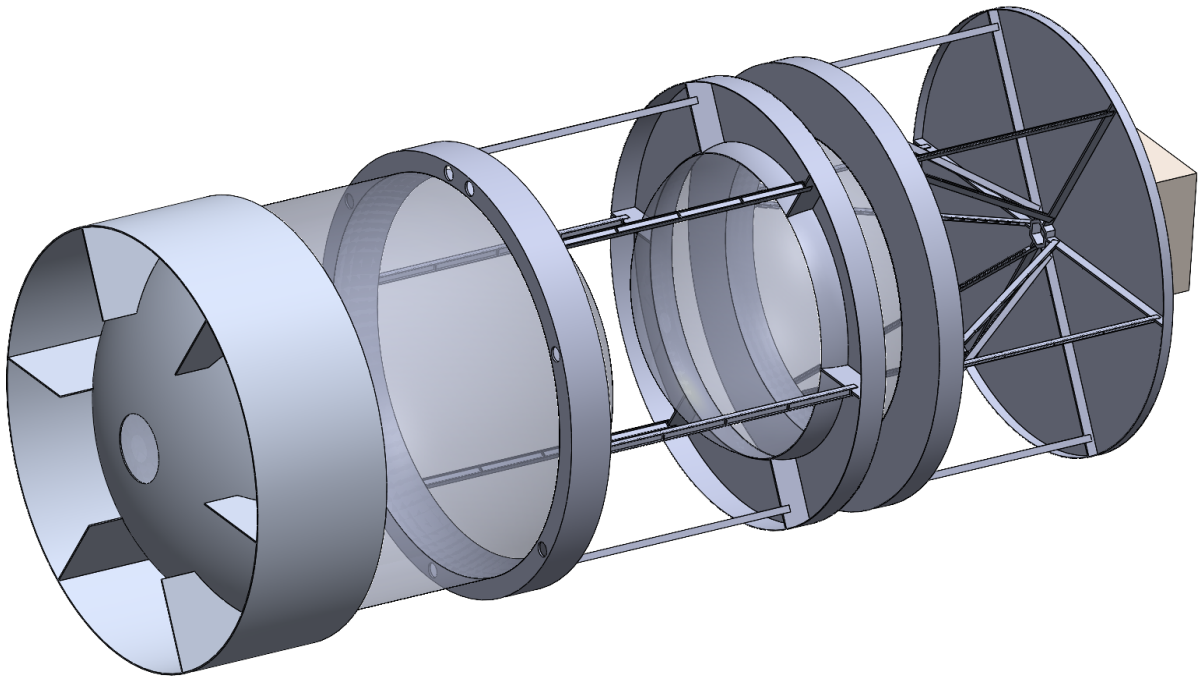

Figure 27: Internal Structure of the Second Stage

It can be noted that the support struts throughout this stage will provide ample mounting fixtures for the various on board systems such as avionics, sensors, RCS, and helium distribution.

Along the entire launch vehicle, there will be a skin that provides structural support and withstands aerodynamic loadings. This skin will be an orthogrid to increase structural rigidity while minimizing mass. The skin panels will be divided into sections defined in Figure 23. An example of an orthogrid skin panel can be seen in Figure 28.

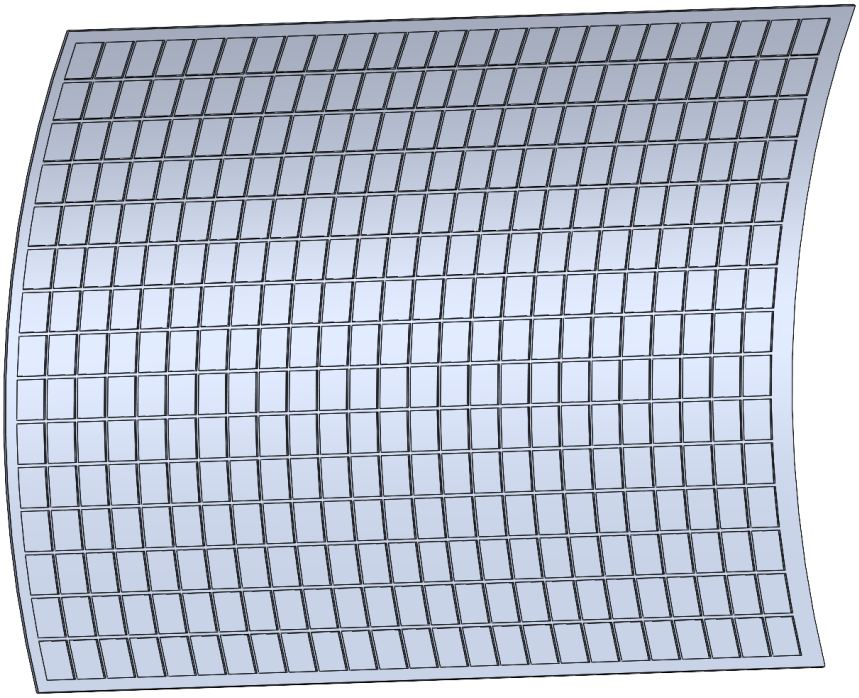

Figure 28: Orthogrid on the AFT Skirt of the Second Stage

The entire skin assembly can be seen in Figure 29.

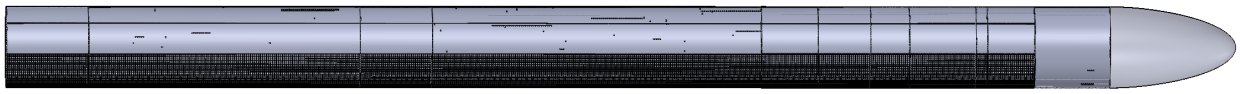

Figure 29: Skin Assembly (half set transparent for visual aid)

Tank Structure

For the propellant tanks, the team initially estimated the thickness of the walls to be around 5mm. Following this conclusion, the design was started. For this launch vehicle, elliptical ends for a cylindrical shell were selected. This saves space within the launch vehicle, thus reducing structural mass while not sacrificing internal volume. It is understood that a common bulkhead is another tactic to conserve length, however this was deemed unnecessary for the added structural complexity it would add to the system. The total volume for a cylindrical tank with two elliptical end caps is

$$V_{tank} = \pi r^2 l + \frac{4}{3\sqrt{2}} r^3 \tag{22}$$

when given an aspect ratio such that the height of the end cap is

$$h = \frac{r}{\sqrt{2}} \tag{23}$$

With this in mind, tank sizing could occur. Tank sizing was done through a MATLAB script utilizing total propellant values and the mixture ratios of the engines while also providing

enough of a safety margin to account for various unforeseen losses. The script followed the following path:

$$m_{p,tot,s1} = \frac{m_{ptot,s1}}{.95} \tag{24}$$

$$m_{LOX,s1} = \frac{m_{ptot,s1}\gamma}{\gamma + 1} \tag{25}$$

where

$$\gamma = mixture\ ratio\ of\ engine \tag{26}$$

$$m_{CH4,s1} = \frac{m_{ptot,s1}}{\gamma + 1} \tag{27}$$

$$V_{LOX,tot,s1} = \frac{m_{LOX,s1}}{\rho_{LOX}} \tag{28}$$

$$V_{LOX,tot,s1} = \frac{V_{LOX,tot,s1}}{.95} \tag{29}$$

$$V_{CH4,tot,s1} = \frac{m_{CH4,s1}}{\rho_{CH4}} \tag{30}$$

$$V_{CH4,tot,s1} = \frac{V_{CH4,tot,s1}}{.95} \tag{31}$$

With these volumes on hand, the equation for length of the cylindrical portion of the tank could be computed by

$$L_{cyl} = \frac{V_{LOX,tot,s1} - \frac{4}{3\sqrt{2}}r^3}{\pi r^2} \tag{32}$$

From this length, the total length of the tank could be computed as

$$L_{tank} = L_{cyl} + 2h \tag{33}$$

The volume and mass calculations were repeated for the upper and lower stage propellants as well as the helium tanks.

From there, an initial radial estimation was made. This estimation was used to calculate an initial length for one of the lower stage tanks. The radius was then iterated until a desirable tank length was achieved. This ideal radius was then used for every propellant tank excluding the upper stage LOX tank. The respective tank lengths were then calculated. The 2nd stage LOX tank was too small to use the lower stage radii, so its radius was calculated separately. The resulting dimensions of each tank are compiled into Table 15.

Tank	**Inner Radius (m)**	**Thickness (m)**	**Length (m)**
CH4	2.9	.005	25.67
LOX 1st Stage	2.9	.005	30.24
LOX 2nd Stage	2.25	.005	4.08
LH2	2.9	.005	70.2
He	.5	.003	2.164

Table 14: Propellant tank dimensions

Hand Calculations for Tank Loadings

Since the structural design relies on the size of the pressure vessels, the sizing of the tanks was verified first. The two stresses associated with pressure vessels are the hoop stress and longitudinal stress. The hoop stress is the stress around the circumference of the pressure vessel due to a pressure gradient. It can be calculated using the following equation:

$$\sigma_H = \frac{Pr}{t}$$

The longitudinal stress is the stress experienced by the pressure vessel due to the internal pressure acting along the direction of the pipe's length. This value is identical to the hoop stress for spheres, so the LOX tank value remains the same. For the LH2 tank, the longitudinal stress is given by the following equation:

$$\sigma_L = \frac{Pr}{2t} = \frac{1}{2}\sigma_H$$

where P is the internal pressure of the tank, r is the inner radius of the tank, and t is the thickness of the tank walls. Applying a factor of safety of 1.5 to the design internal pressure, the final total internal pressure is 465396.1 Pa (67.5 psi). The dimensions of the tanks can be found in Table 15. Substituting these values into the above stress equations grants the following results.

Tank	Hoop Stress (MPa)	Longitudinal Stress (MPa)
CH4	269.9	134.95
LOX 1st Stage	269.9	134.95
LOX 2nd Stage	209.4	104.7
LH2	269.9	134.95
He	77.6	38.8

Table 15: Propellant tank stress results

Comparing these values to the Aluminum 2219-T851 tensile yield strength of 290 MPa, it is clear that the aluminum will be strong enough to withstand the internal pressure of the tanks. This provides a factor of safety of 1.07 for the CH4, LOX first stage, and LH2 tanks, a factor of safety of 1.38 for the LOX second stage tank, and a factor of safety of 3.74 for the Helium tanks in addition to the 1.5 factor of safety already included for all tanks.

Finite Element Structural Analysis (FEM)

A preliminary SolidWorks FEM analysis has been conducted on the propellant tanks and tank support structures. Large scale analyses such as these take sometimes hours to perform, as many parts require very fine mesh sizes. Because of this, further analysis of the structures performance under flight loads has also been done in Patran, to be discussed further. The FEM for the tanks and tank structures was completed to prove the structure's resilience under a substantial pressure loading of 60 psi. This preliminary analysis can be seen in Figures 30 and 31.

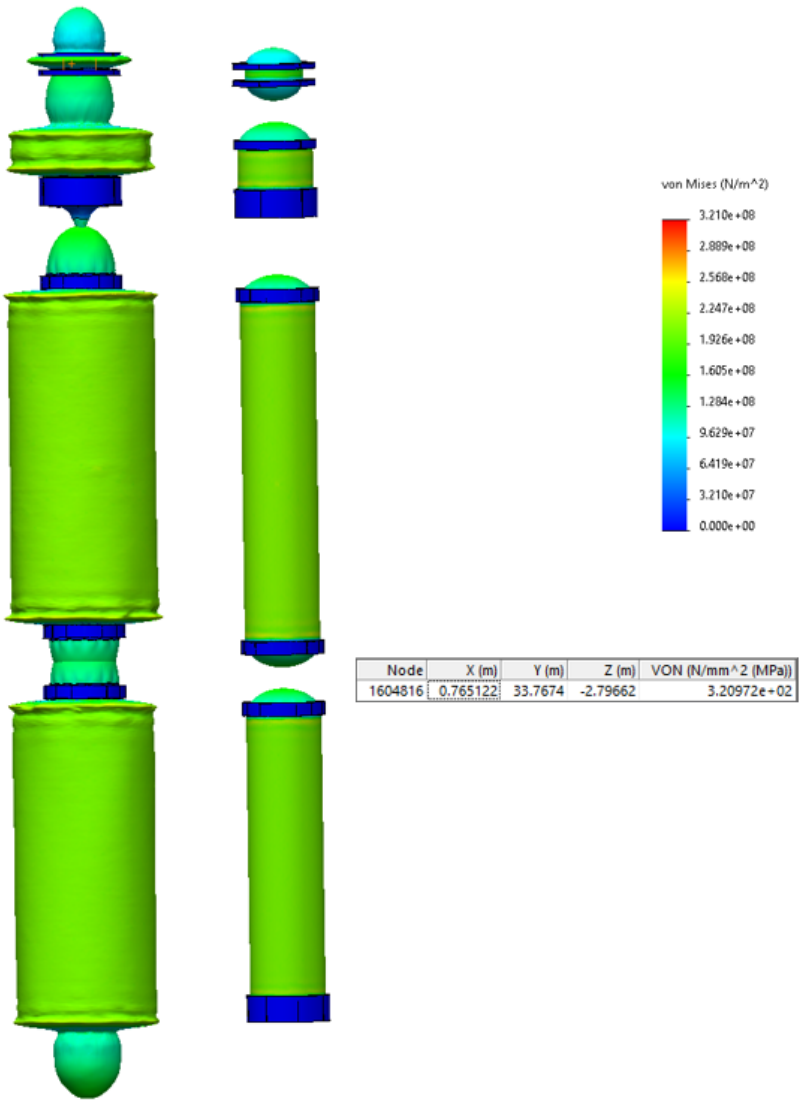

Figure 30: Preliminary Tank Stress FEM

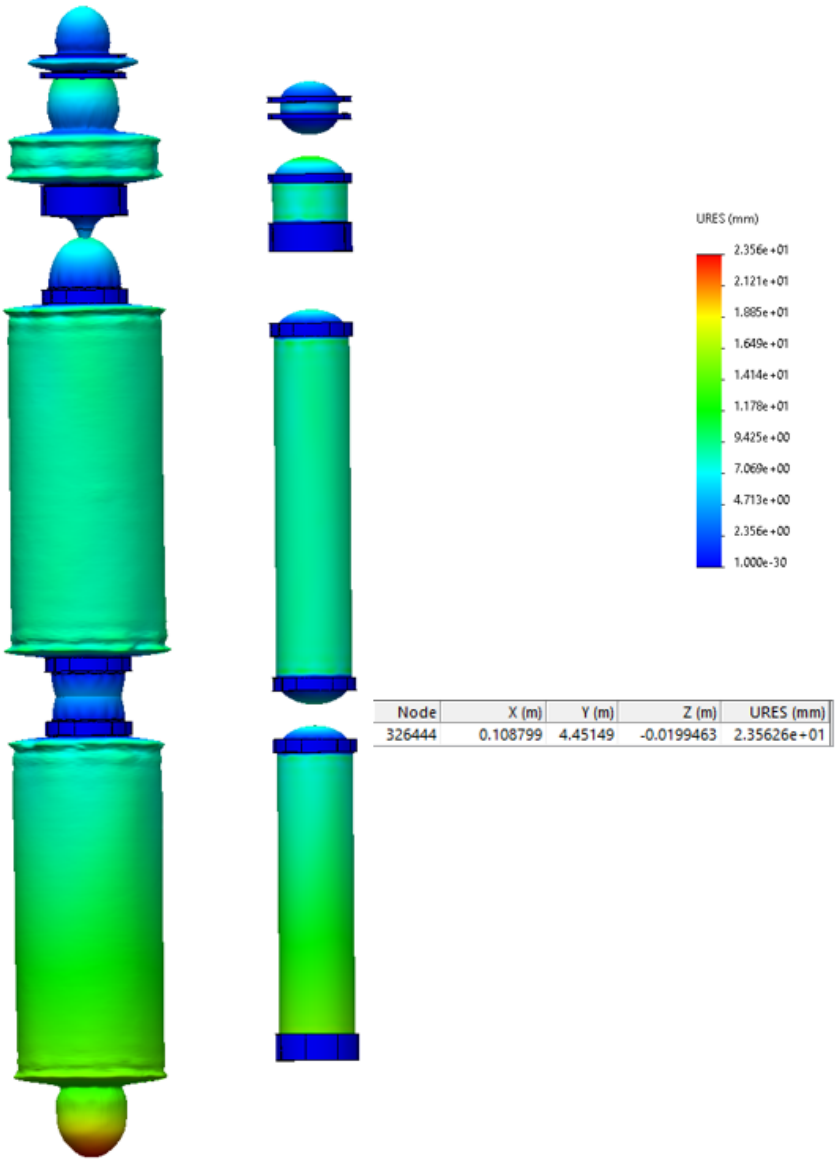

Figure 31: Preliminary Tank Displacement FEM

From these pseudocolor plots, it can be seen that the maximum stress is still well within the allowable stress of each material. This fact may allow weight reduction to take place in the future. It can also be seen that the maximum displacement for any of the tanks only reached about 2mm at peak, proving that there will be negligible displacements due to tank pressurization.

Following the preliminary tank analysis, a full Patran model of the system was developed and run to calculate structural resilience under full flight conditions with high fidelity. To begin this analysis, the tanks were first modeled in Patran. This can be seen in Figure 32.

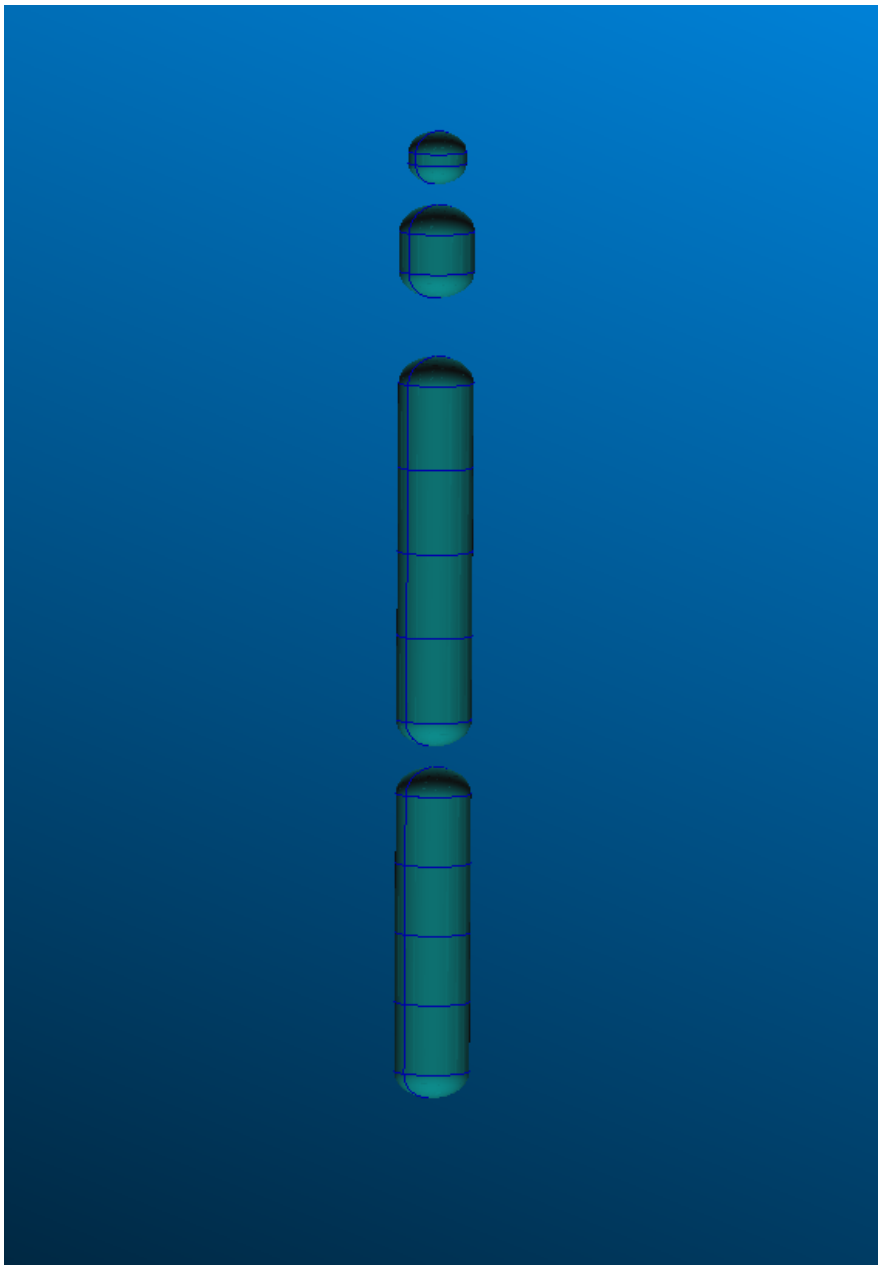

Figure 32: Patran Tank Model

These tanks were then subjected to an internal pressure loading of 45 psi. This was converted to Pa by the conversion

$$P_{SI} = P_{ENG} * 6894.76 \tag{34}$$

$$P_{SI} = 45 * 6894.76 \tag{35}$$

$$P_{SI} = 310264Pa \tag{36}$$

This loading application can be seen in Figure 33.

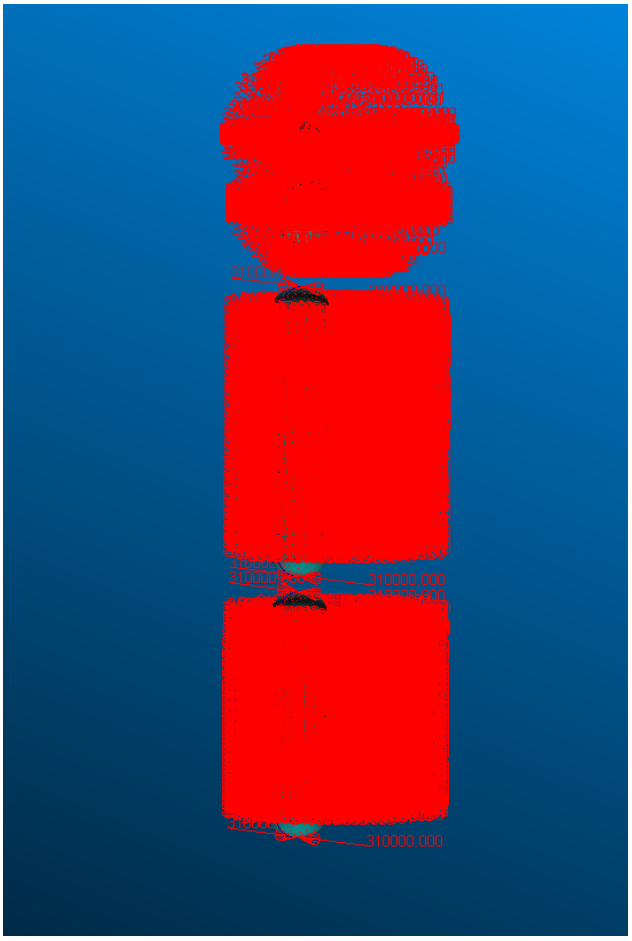

Figure 33: Patran Tank Pressurization

Next, all of the internal support structures for the tanks were modeled. This Patran model can be seen in Figure 34.

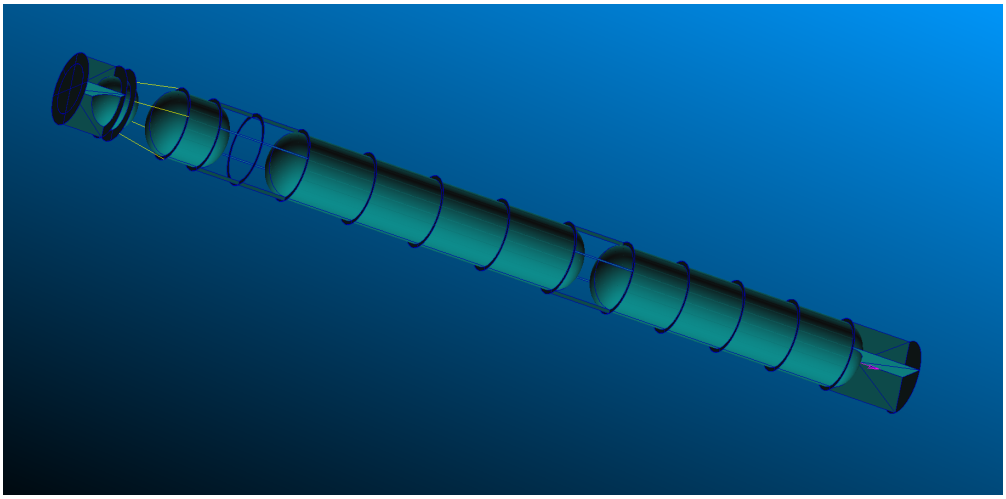

Figure 34: Patran Support Modeling

Next, the bottom surfaces of the thrust structure were constrained at the points of engine contact with zero displacement in all axes. A distributed load for the payload over the surface area of the payload support structure was calculated at 6 g's by the following

$$F_{payload} = \frac{50T * 1000\frac{kg}{T} * 9.81\frac{m}{s^2} * 6}{\pi * 3.25^2} \tag{37}$$

$$F_{payload} = 88690\frac{N}{m^2} \tag{38}$$

This loading was distributed over the top surface of the payload support structure. The forces and constraints applied to the support structure can be seen in Figure 35.

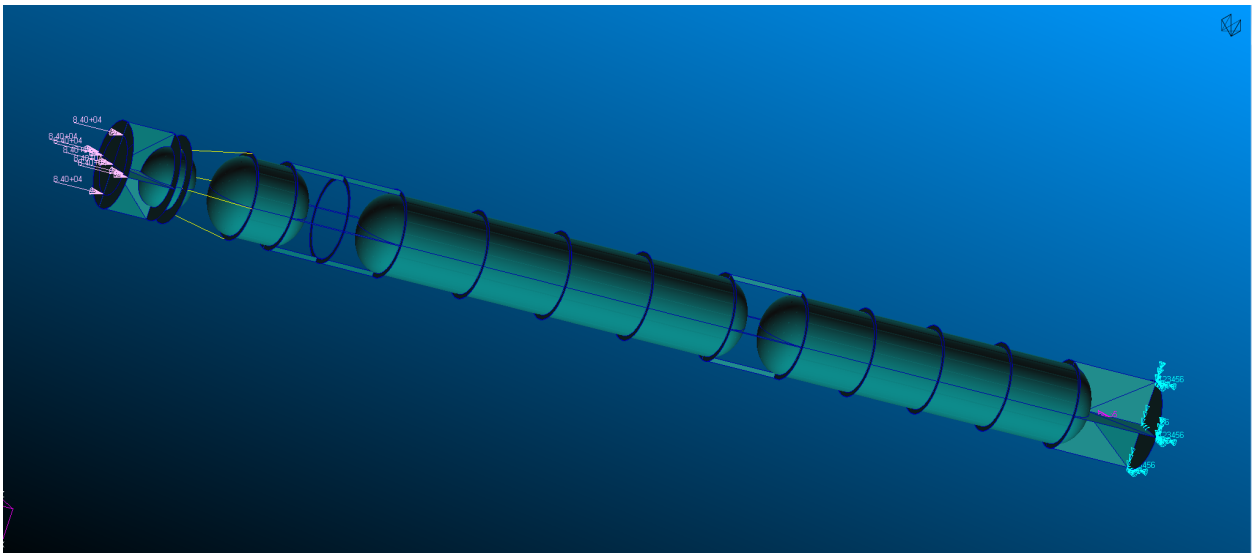

Figure 35: Patran Support Constraints

Next, the skins were modeled in Patran. These skins can be seen in Figure 36

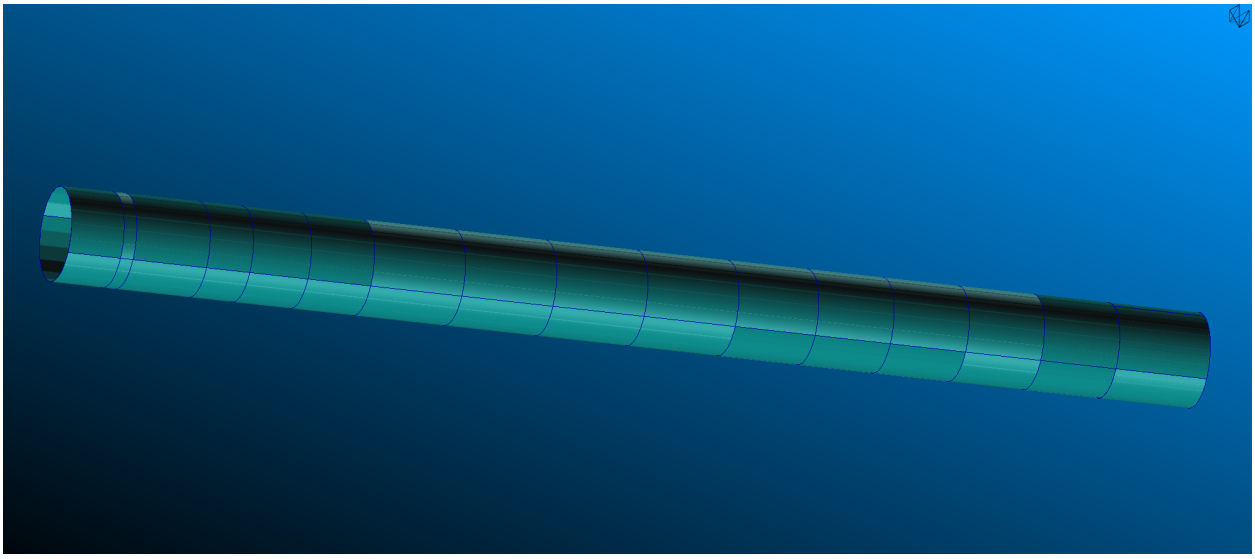

Figure 36: Patran Skin Models

For the skin, a distributed aerodynamic loading was calculated. The Genesis-1 max-q event will occur at approximately 11 kilometers with a maximum velocity of about 425 m/s. Knowing that the vehicle has a drag coefficient of about 0.45 at max-q, the drag force per unit area can be approximated as

$$F/A = C_d \frac{1}{2} \rho V^2 = 0.45(\frac{1}{2})(0.3)(425^2) = 12192.77 \ \frac{N}{m^2}$$

This force was applied as a CID distributed load across the surface of the skin panels parallel to the freestream. The applied skin friction can be seen in Figure 37.

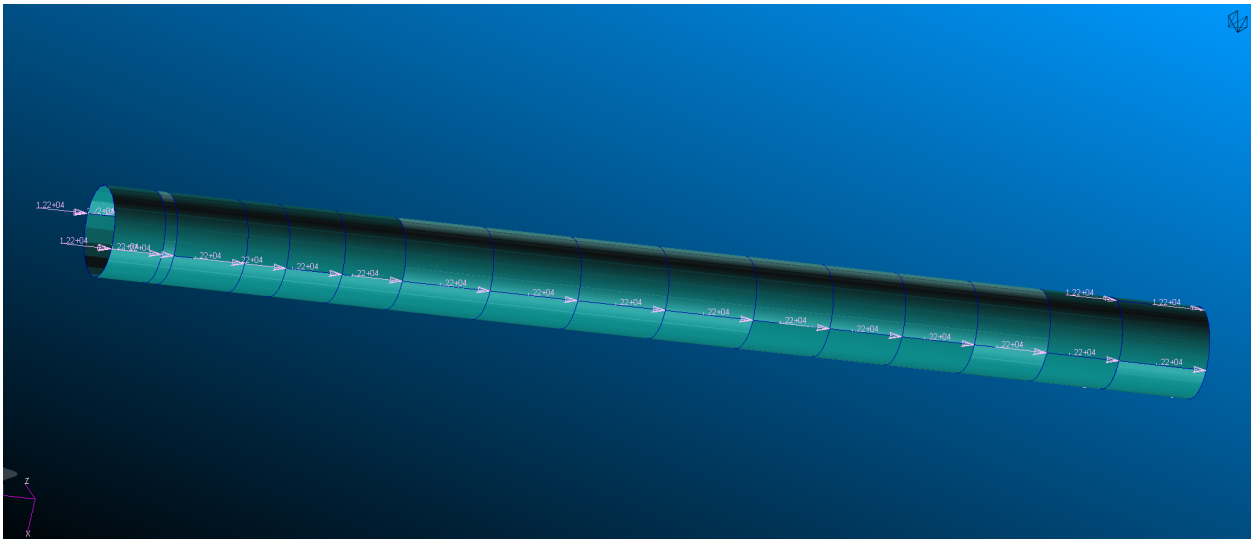

Figure 37: Patran Skin Distributed Load

Lastly, the entire structure was subjected to an inertial loading equivalent to 6 g's in the body fixed negative Y axis and 1 g 30 degrees to the negative Y axis to simulate maximum

g-force in a gravity turn maneuver that also happens at max-q. This analysis was chosen to maximize body stresses in a worst-case scenario. The resulted inertial loading applied to all structures can be seen in Figure 38.

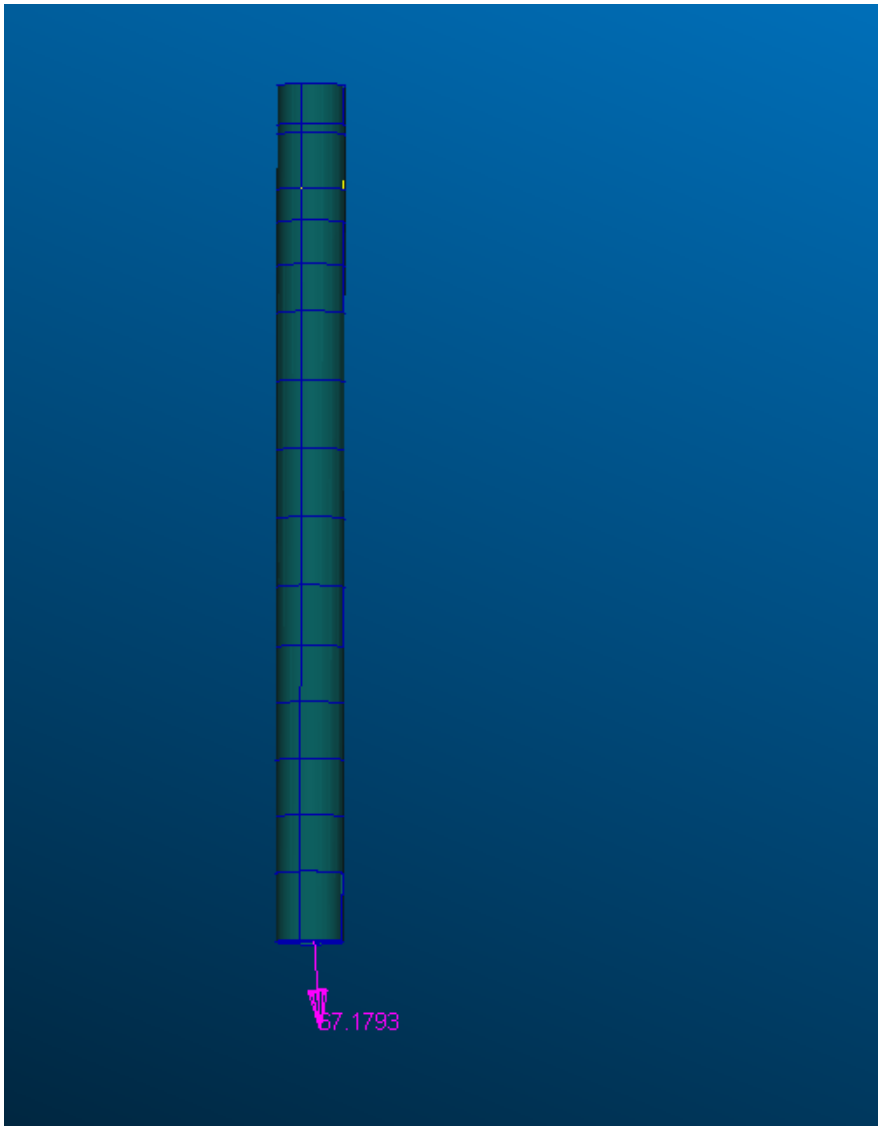

Figure 38: Patran Inertial Loading

From here, a mesh was applied to each section of the vehicle and an analysis was performed in Patran. The resultant data was extremely promising. The resulting displacement pseudocolor plot can be seen in Figure 39.

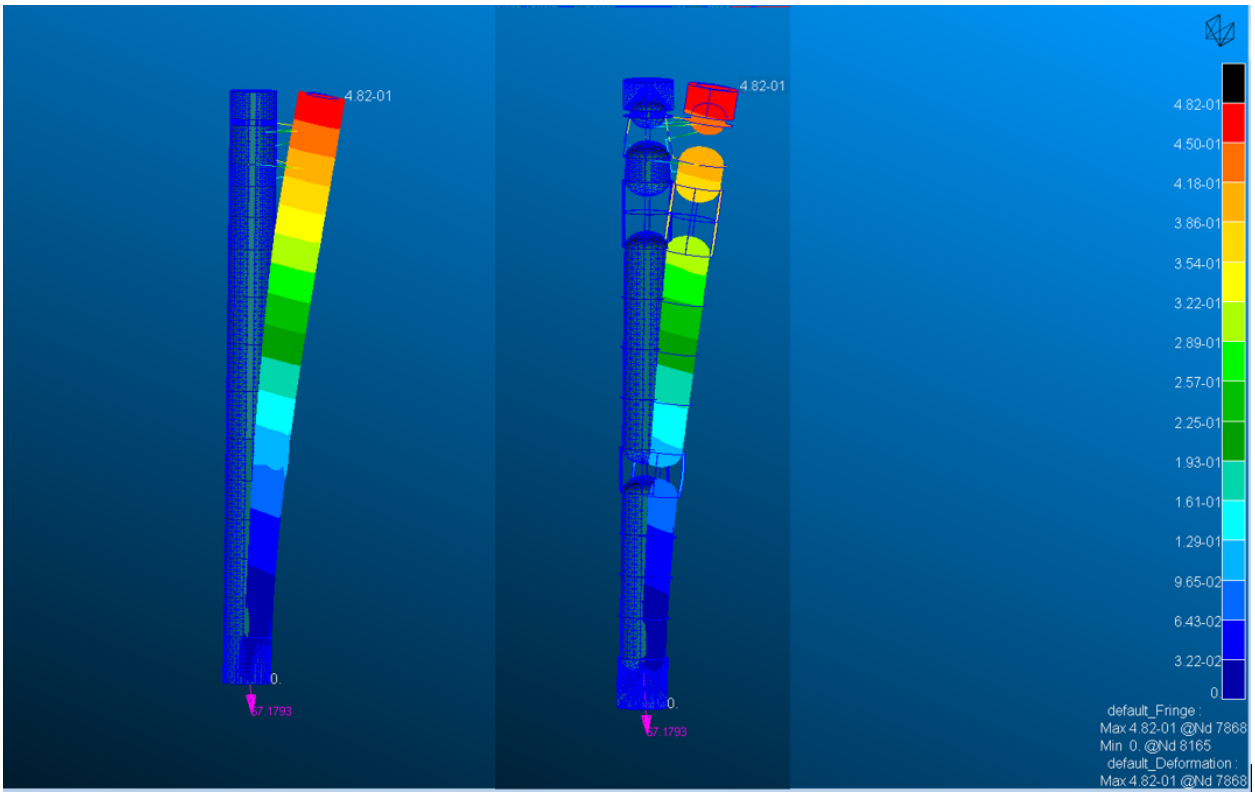

Figure 39: Patran Simulated Displacement

In can be seen in Figure 39 that the maximum displacement occurs at the top of the vehicle at about 50 cm. This is very promising, as if the displacement is propagated evenly, the angle formed by the displaced body would only be approximately .27 degrees. This is a displacement barely noticeable to the human eye. It can also be seen that no other section of the structure experiences any particularly large displacements, thus proving that the structure should withstand max forcing if manufactured to specifications.

The stress pseudocolor plot of the structure can be seen in Figure 40.

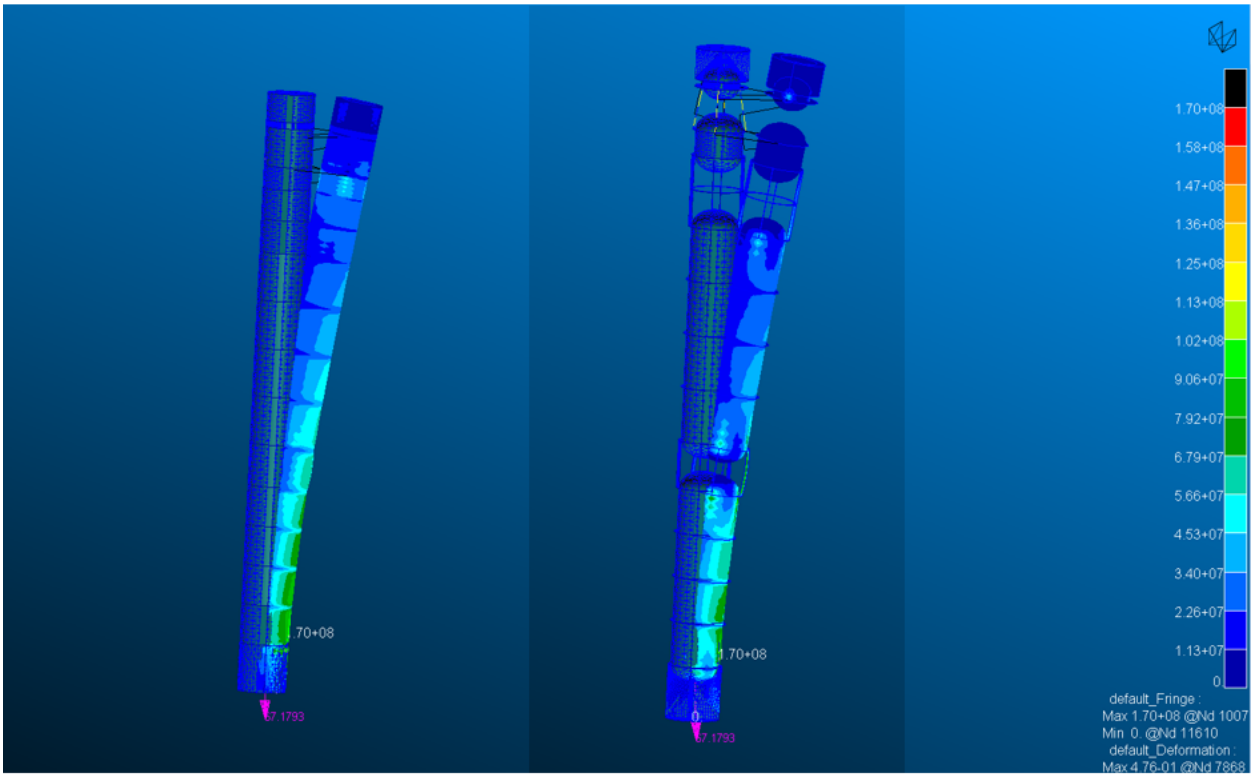

Figure 40: Patran Simulated Stress

From the plots in Figure 40 it can be seen that the maximum stress experienced in the Genesis-1 vehicle occurs towards the first stage thrust structure and reaches about 170 MPa. This is a great range for stress because it is utilizing the material efficiently. The maximum stress for 6061-T6 aluminum is about 207 MPa and the maximum stress for A2219 aluminum is about 324 MPa, therefore the structure still maintains a factor of safety of at least 1.2. If more data becomes available from real world testing and the flight forces become greater than anticipated, the structure has some room to grow, however, this analysis was ran at max-q conditions and max-g conditions which will not occur simultaneously in reality. This adds an extra factor of safety level that can be exploited if deemed necessary.

From this analysis, the team at Blue Oregano has deemed this launch vehicle's structural design feasible in simulation.

Materials / trade studies

Tanks, Engine Mounts, Skin, Thermal Control

When deciding on the structural components used in the tanks, engine mounts, and skin, two trade studies were completed to determine what the optimal material would be. The Materials considered were aluminum alloys 7075-T6, 6061-T6, and A2219 for their notable prior use in aeronautics as lightweight study alloys. Also included were 301 and 304 stainless steel and Ti-6Al-4V(Grade 5) titanium for their strength. These materials were weighed over 5 categories; Density, Strength, Cost, Thermal Conductivity, and Corrosion Resistance. These categories were chosen based on a number of factors. Lower density material allows for a lower vehicle weight and requires less total thrust to escape atmosphere. The higher a materials structural strength, the more force the material can take before buckling or breaking, allowing for a thinner structure while retaining its ability to withstand extreme

force. Lower cost allows for better use of budget. Thermal Conductivity and Corrosion resistance are both rather important but change depending on the area the material is used, as such these materials were additionally judged based on their position. These Values can be seen in Table 16 below.

Actual Value						
Item	7075-T6 Aluminum	6061-T6 Aluminum	304 Stainless Steel	301 Stainless Steel	Titanium (Ti-6Al-4V (Grade 5))	A2219 Aluminum
Density (g/cc)	2.81	2.7	8	7.93	4.43	2.84
Strength (MPa)	331	207	650	637.5	550	324
Cost ($ per Kg)	3	3	6	3	21	2
Thermal Conductivity (W/m.K)	196.00	176.50	16.20	16.00	7.20	143.00
Corrosion Resistance	Low	High	High	High	High	Med

Table 16: Material Trade Study with Actual Values

These actual values were then normalized into reference categories and given values between 1 and 6 as shown in Table 17 and given unweighted values corresponding to the normalized values in Table 18

Normalization Values						
Item	1	2	3	4	5	6
Density (g/cc)	> 6	5-5.99	4-4.99	3-3.99	2-2.99	1 - 1.99
Strength (MPa)	200-299	300-399	400-499	500-599	600-699	700+
Cost ($ per Kg)	50-59.99	40-49.99	30-39.99	20-29.99	10-19.99	1 - 9.99
Thermal Conductivity (W/m.K)	>250	200-249	150-199	100-149	50-99.99	1-49.99
Corrosion Resistance	Dissolves in water	Low	Med-Low	Med	Med-High	High

Table 17: Material Trade Study with Normalized Values

Un-Weighted Values						
Item	7075-T6 Aluminum	6061-T6 Aluminum	304 Stainless Steel	301 Stainless Steel	Titanium (Ti-6Al-4V (Grade 5))	A2219 Aluminum
Density (g/cc)	5	5	1	1	3	5
Strength (MPa)	2	1	5	5	4	2
Cost ($ per Kg)	6	6	6	6	4	6
Thermal Conductivity (W/m.K)	3	3	6	6	6	4
Corrosion Resistance	2	6	6	6	6	4

Table 18: Material Trade Study with Unweighted Values

After normalizing and creating unweighted values, weights are assigned to each category by importance scaling from 1 to 5 and representing the focus of the trade study done. This material trade study is split between Weighting Factors for the Skin and structure and Weighting factors for the Tankage as shown below in Tables 19 and 20.

Weighting Factors (Skin and Structure)		
Item	Factor	Reason
Density (g/cc)	5	The weight needs to be low to save cost and allow for easier exit from atmosphere.
Strength (MPa)	3	Strength is important but can be mitigated by design and supports.
Cost ($ per Kg)	3	Budgets are Very Important, but we need to get the job done.
Thermal Conductivity (W/m.K)	2	Thermal Conductivity isnt too important but should be considered due to extreme temperature changes.
Corrosion Resistance	4	Product must withstand atmospheric conditions of Mars.

Table 19: Weighting Factors for the Skin and Support Structures

Weighting Factors (Tankage)		
Item	Factor	Reason
Density (g/cc)	5	The weight needs to be low to save cost and allow for easier exit from atmosphere.
Strength (MPa)	3	Strength is important but can be mitigated by design and supports.
Cost ($ per Kg)	3	Budgets are Very Important, but we need to get the job done.
Thermal Conductivity (W/m.K)	2	Thermal Conductivity isnt too important but should be considered due to extreme temperature changes.
Corrosion Resistance	2	Product must only withstand fuel, oxidizer, and helium inside tanks.

Table 20: Weighting Factors for the Tankage and Internal Pipe Assembly

After weighting The factors it is notable that the values of Density, strength, and cost did not change between the two versions of the weighting factors this is because all three are equally

important in both situations. However the weight of Corrosion resistance is reduced when pertaining to the tanks because they are only interacting with the interior of the rocket and pressurized fuel/oxidizer. This results in a change in choice of material as shown if Tables 21 and 22

Weighted Values (Skin and Structure)						
Item	7075-T6 Aluminum	6061-T6 Aluminum	304 Stainless Steel	301 Stainless Steel	Titanium (Ti-6Al-4V (Grade 5))	A2219 Aluminum
Density (g/cc)	25	25	5	5	15	25
Strength (MPa)	6	3	15	15	12	6
Cost ($ per Kg)	18	18	18	18	12	18
Thermal Conductivity (W/m.K)	6	6	12	12	12	8
Corrosion Resistance	8	24	24	24	24	16
TOTALS	63	76	74	74	75	73

Table 21: Weighting Factors for the Tankage and Internal Pipe Assembly

Weighted Values (Tankage)						
Item	7075-T6 Aluminum	6061-T6 Aluminum	304 Stainless Steel	301 Stainless Steel	Titanium (Ti-6Al-4V (Grade 5))	A2219 Aluminum
Density (g/cc)	25	25	5	5	15	25
Strength (MPa)	6	3	15	15	12	6
Cost ($ per Kg)	18	18	18	18	12	18
Thermal Conductivity (W/m.K)	6	6	12	12	12	8
Corrosion Resistance	4	12	12	12	12	8
TOTALS	59	64	62	62	63	65

Table 22: Weighting Factors for the Tankage and Internal Pipe Assembly

Through this trade study it was decided that Aluminum 6061-T6 was the best option for the skin, support structures, and mounts because of its low density and cost while maintaining an exceptional corrosion resistance and an effective Thermal Conductivity. Additionally this choice of material produces the lowest structural mass out of the options reducing the volume of propellant required. For the Tank assemblies and piping structures Aluminum A2219 was chosen over Aluminum 6061-T6 due to the lower thermal conductivity allowing it to act as a better insulator against cryogenic temperatures and increased strength being useful in the highly pressured environment without sacrificing much in the way of density.

Fairings / Aerodynamic Coverings/ Thermal Control Surfaces

In order to provide protection of the payload and electrical systems, thermal shields will be added to the conical section of the fairings to provide thermal control [15]. Additionally, the payload fairing will possess an air-conditioning duct that will direct conditioned air throughout the conical section to provide thermal and humidity control while grounded. This duct will be controlled by the Genesis's main DAS where the ground control team can activate the duct during pre-launch activities when high humidity levels are reached, which is a likely occurrence at the Kennedy Space Center. Lastly, the protection of the electrical system and sensor will be completed with metallic fairing constructions where the fairings will provide electromagnetic shielding for the systems [15]. For the fairings, aerodynamic coverings, and thermal control surfaces within the vehicle, it was decided that Aluminum 6061-T6 and A2219 would suffice from the results of Tables 21 and 22 above. By doing this, the vehicle does not suffer increased weight. The figure below presents Atlas V based thermal shields that will be attached to the Genesis.

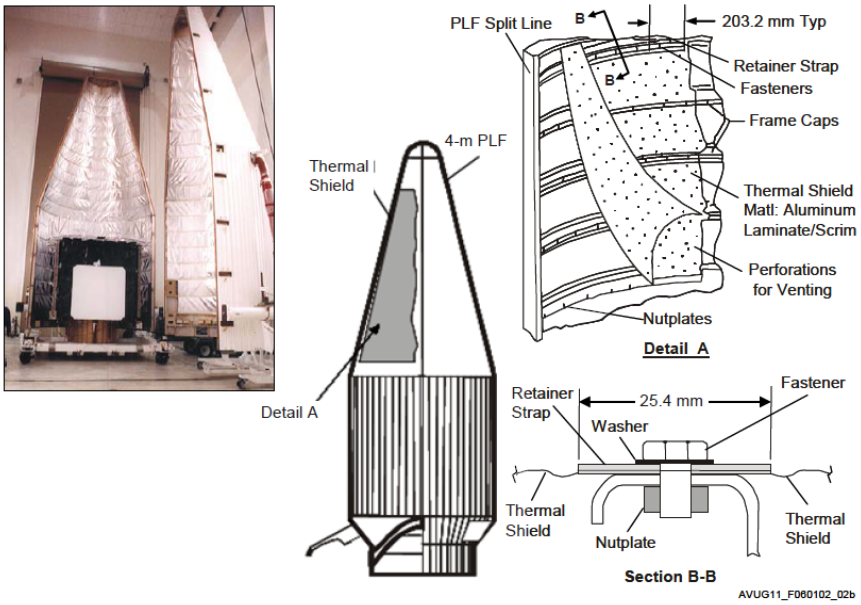

Figure 41: Atlas V PLF Thermal Shields

Aerodynamics

This section, aerodynamic analysis is based on chapter 6.4 of [7] and is covering the flight path on ascent and orbital trajectory simulation analysis.

Drag calculations

The initial drag estimations assumed a constant drag coefficient C_D of 0.3 for the entire ascent portion. The drag force can then be expressed using the definition of drag:

$$D = C_D S_{ref} \frac{1}{2} \rho v^2 \tag{39}$$

The atmospheric density and pressure can be approximated using the exponential model:

$$\rho(h) = \rho_0 e^{-h/h_0} \tag{40}$$

$$P_\infty(h) = P_0 e^{-h/h_0} \tag{41}$$

where $\rho_0 = 1.225\ (kg/m^3)$ and $P_0 = 101325\ (N/m^2)$ are atmospheric density and pressure at sea level and $h_0 = 7.64\ (km)$ is the scale height. The magnitude of the gravitational force can also be approximated as:

$$g(h) = \frac{g_0}{\left(1 + \frac{h}{R_E}\right)^2} \tag{42}$$

which is also a function of altitude h. The Δv loss due to drag and gravity can be expressed as:

$$\Delta v_{loss,aero} = \int_0^{t_f} \frac{D}{m} dt \tag{43}$$

$$\Delta v_{loss,grav} = \int_0^{t_f} g sin\gamma \tag{44}$$

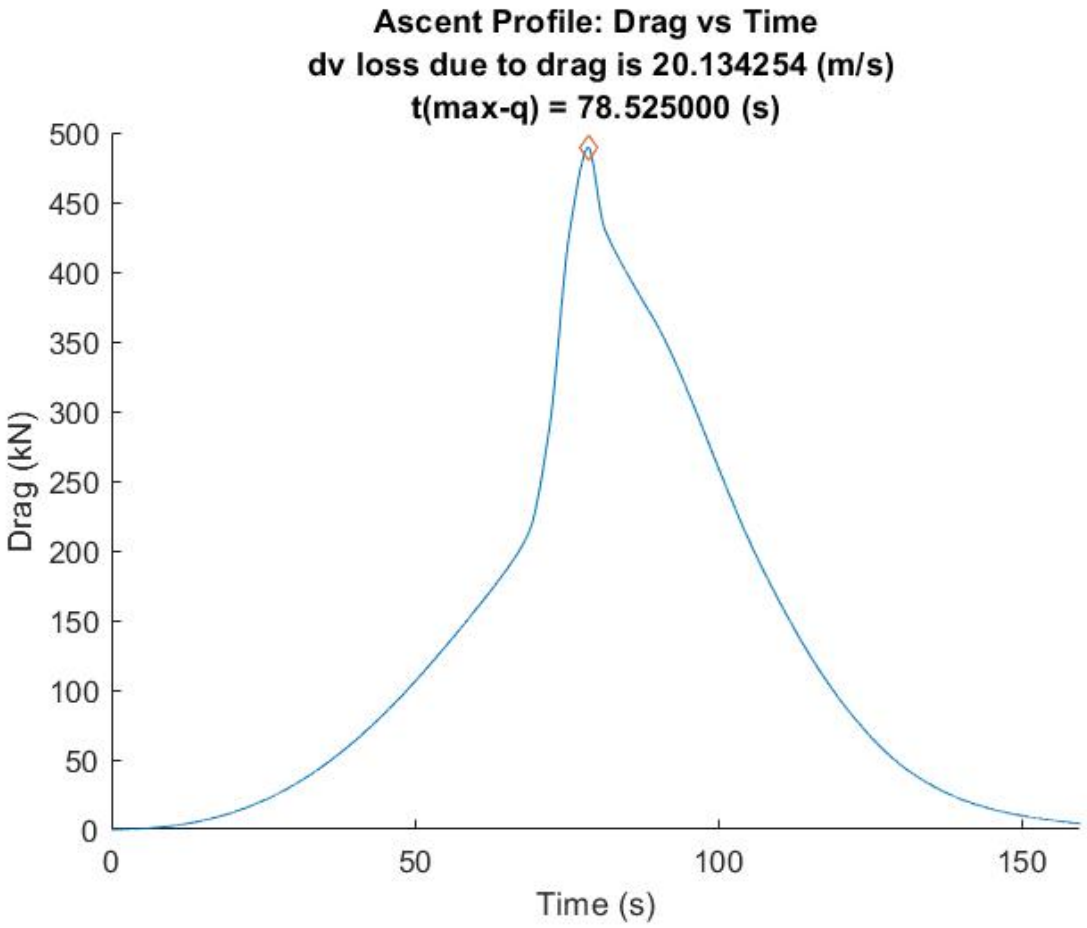

Figure 42: Ascent trajectory: drag profile

Using the ascent simulation explained later in in this section, collected data was used to recover the drag profile for the interval of flight which happens in the thick portion of the atmosphere (below 100 km altitude) and is presented in Figure 42. the total loss due to drag is only $\Delta v \approx 20\ m/s$. These aspects are both due to the vehicle taking a vary steep path out of the atmosphere and due to a constant drag coefficient not being scaled with the mach number.

CFD Analysis: Drag results

The CFD analysis of the vehicle was conduced at a wide range of Mach numbers, focusing at the subsonic and transonic regions. The exact CAD geometry (see Figure 43) was used with the following simplifications:

- The entire vehicle was considered a single solid body with no internal components.
- Since the vehicle was expected to escape thick layers of atmosphere while using the first step only, no separate CFD analysis was considered for the second step.
- Engines were simplified by closing off the internal volume and creating a surface at the nozzle exit.
- Engine structure was also closed of to reduce the complexity of the geometry.

- Engine exhaust was approximated as an additional inlet for air, with the specified velocity of the engine exhaust speed.
- The ratio of specific heat of air γ was assumed to remain constant during ascent at a value of 1.4.

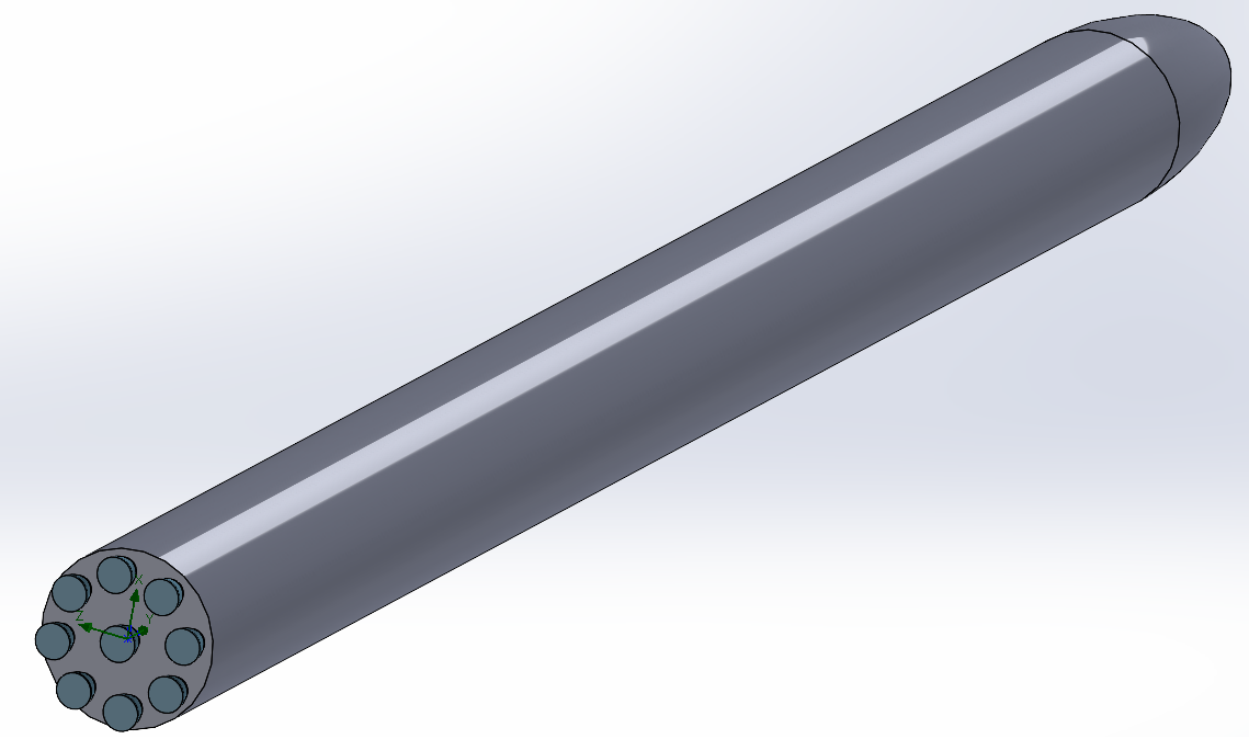

Figure 43: Simplified CAD geometry for CFD analysis.

For simulating the ascent of the vehicle it was convenient to re-derive drag coefficient using the ideal gas law and the definition of the speed of sound to get:

$$C_D = \frac{F_D}{\frac{1}{2} P M^2 \gamma_{air} A} \tag{45}$$

which makes drag coefficient only a function of ambient pressure P, mach number, and drag force F_D. The results of the CFD analysis at the altitude of 1 km above the surface is shown on Figure 44.

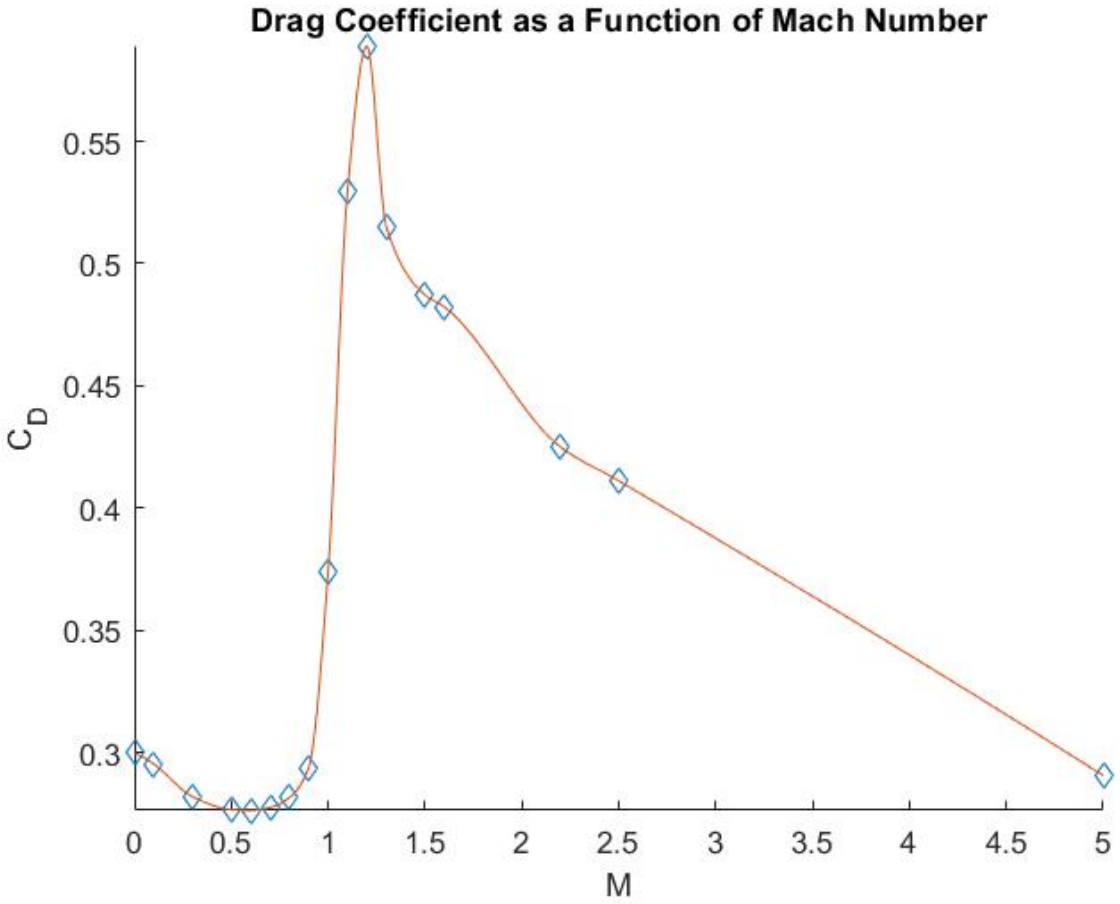

Figure 44: Drag coefficient as a function of mach number at 1 km altitude above sea level.

Ascent Heating Hand Calculations

While heating on ascent doesn't affect the vehicle a significant amount, it is still important to cover all bases. To determine the heating effects on the vehicle, the change in temperature across the shock wave that forms on the nose of the vehicle at max Q is analyzed. From Figure 51, the Mach number and altitude of the max Q event can be determined. At an altitude of 10.34 km the ambient temperature is approximately 221.03 K. At max Q, the vehicle is traveling at Mach 1.20. The equation for the temperature change across a normal shock is shown below.

$$\frac{T_2}{T_1} = \left(1 + \frac{2\gamma}{\gamma+1}(M_1^2 - 1)\right) \frac{2 + (\gamma-1)M_1^2}{(\gamma+1)M_1^2} = 1.219 \tag{46}$$

Using the above ratio and the given flight condition, the maximum estimated temperature experienced by the nosecone is approximately 269.46 K which is well within the operating range of Aluminum 6061 T6. The total temperature raise is 48 K.

CFD Analysis: Aerodynamic heating results

Using the ascent dynamics simulation, altitude and mach number of the vehicle at the maximum dynamic pressure were determined to be approximately 10 km and 1.2, respectively.

A separate CFD simulation has been conducted at this location to determine the maximum heating due to aerodynamic effects. Figures 45 and 46 displays the resulting heating profile shown on the surface of the vehicle, two cross sectional areas showing the total pressure change across the vehicle and resulting airflow direction.

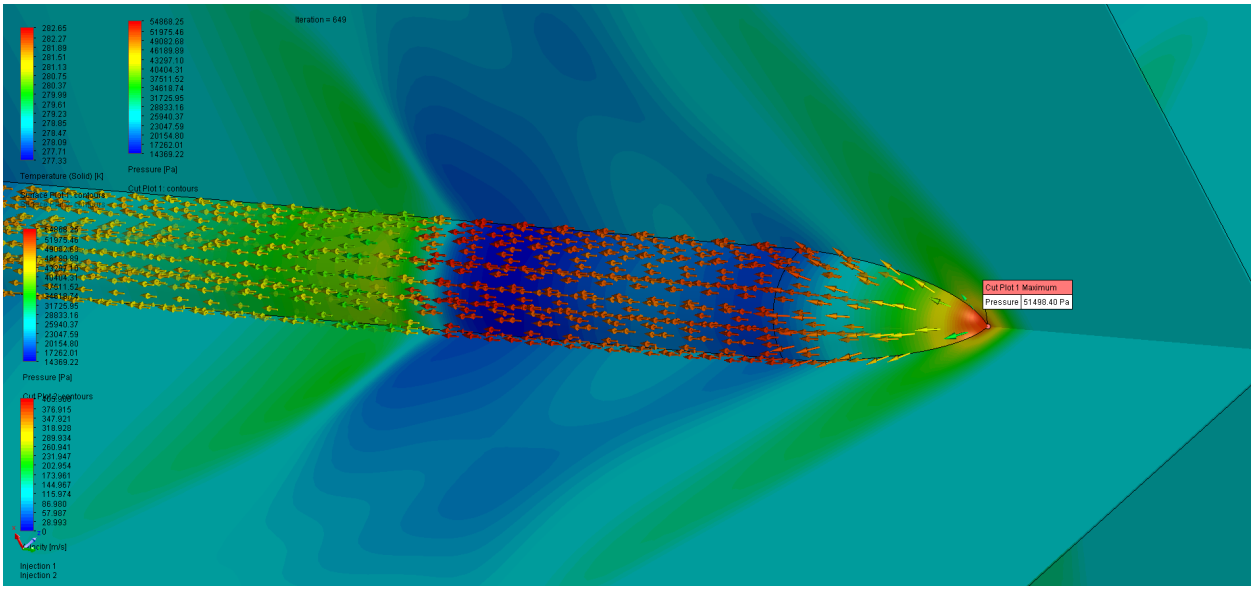

Figure 45: CFD results for aerodynamic heating, total pressure and airflow profile over the nosecone of the vehicle at maximum dynamic pressure during ascent.

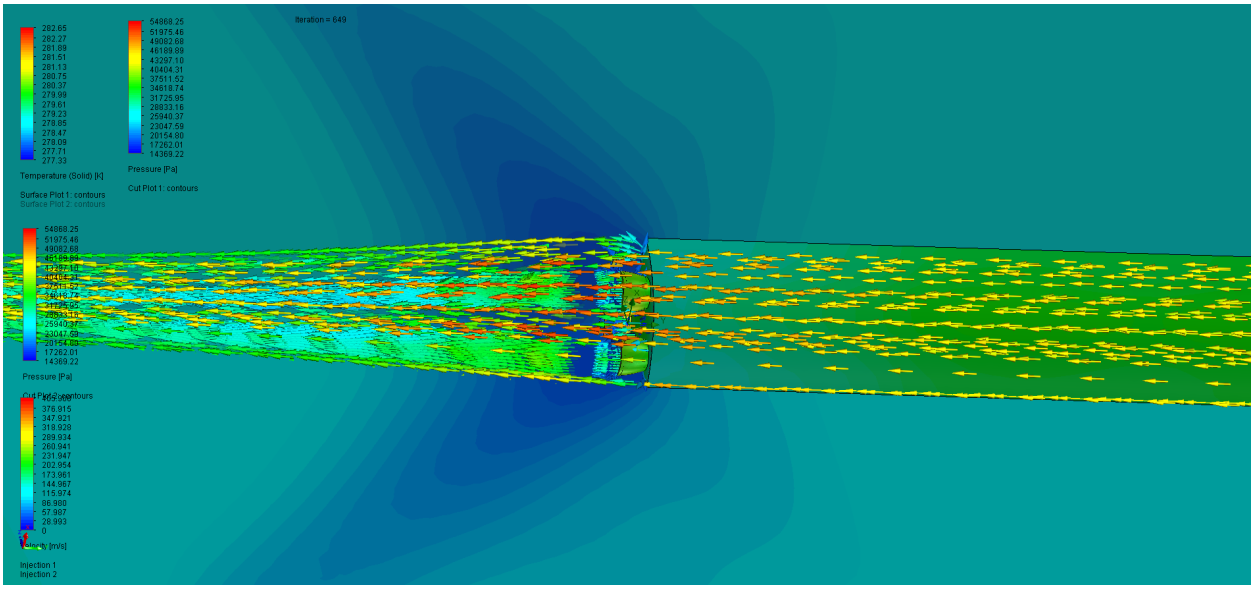

Figure 46: CFD results for aerodynamic heating, total pressure and airflow profile over the aft of the vehicle at maximum dynamic pressure during ascent.

Note, the ambient pressure at 10km altitude was assumed to be 221.034 K (boundary condition of the CFD analysis) and the maximum temperature on the surface of the vehicle (at the tip) was 282.65 K. Therefore, maximum aerodynamic heating during ascent is 61.62 K

as calculated by CFD analysis. This temperature is well below the melting temperature of aluminum and since no heat radiation was considered for this analysis, it is safe to say the vehicle will not need additional shielding to avoid overheating.

Center of Pressure

In order to determine the launch vehicle's stability, its center of pressure first had to be located. This was done in part using Barrowman's equations [7]. Given that the launch vehicle has no fins, the only contributing components are the ogive nosecone and body. The distance of the nosecone's CP from the tip of the rocket was determined using the following equation.

$$x_{CP,N} = 0.466(L_N) = 0.466(7m) = 3.262m \tag{47}$$

The body, whose cross section can be approximated as a rectangle, has a center of pressure which occurs at it's longitudinal midpoint, 43.81 meters. Adding these values gives the resultant location of the vehicle's center of pressure, at 47.43 meters from the tip.

Flight Dynamics Analysis and Trajectory Simulation

The flight path of the vehicle was simulated by integrating the the following set of ordinary differential equations to find velocity v, flight path angle γ, altitude h and downrange distance x:

$$\begin{aligned}
\frac{dv}{dt} &= \frac{T}{m} - \frac{D}{m} - gsin\gamma \\
\frac{d\gamma}{dt} &= \left(\frac{g}{v} - \frac{v}{R_E + h}\right) cos\gamma \\
\frac{dh}{dt} &= vsin\gamma \\
\frac{dx}{dt} &= \frac{R_E}{R_E + h} vcos\gamma
\end{aligned} \tag{48}$$

where mass m of the vehicle and mass flow rate are defined as:

$$\begin{aligned}
m(t) &= m_0 - \dot{m}t \\
\dot{m} &= \frac{T}{g_0 I_{sp}}
\end{aligned} \tag{49}$$

The thrust is defined by the engine's exhaust velocity v_e, exit pressure P_e and exit area A_e:

$$T = \dot{m}v_e + (P_e - P_\infty)A_e \tag{50}$$

The results of the numerical integration are presented in the following figures.

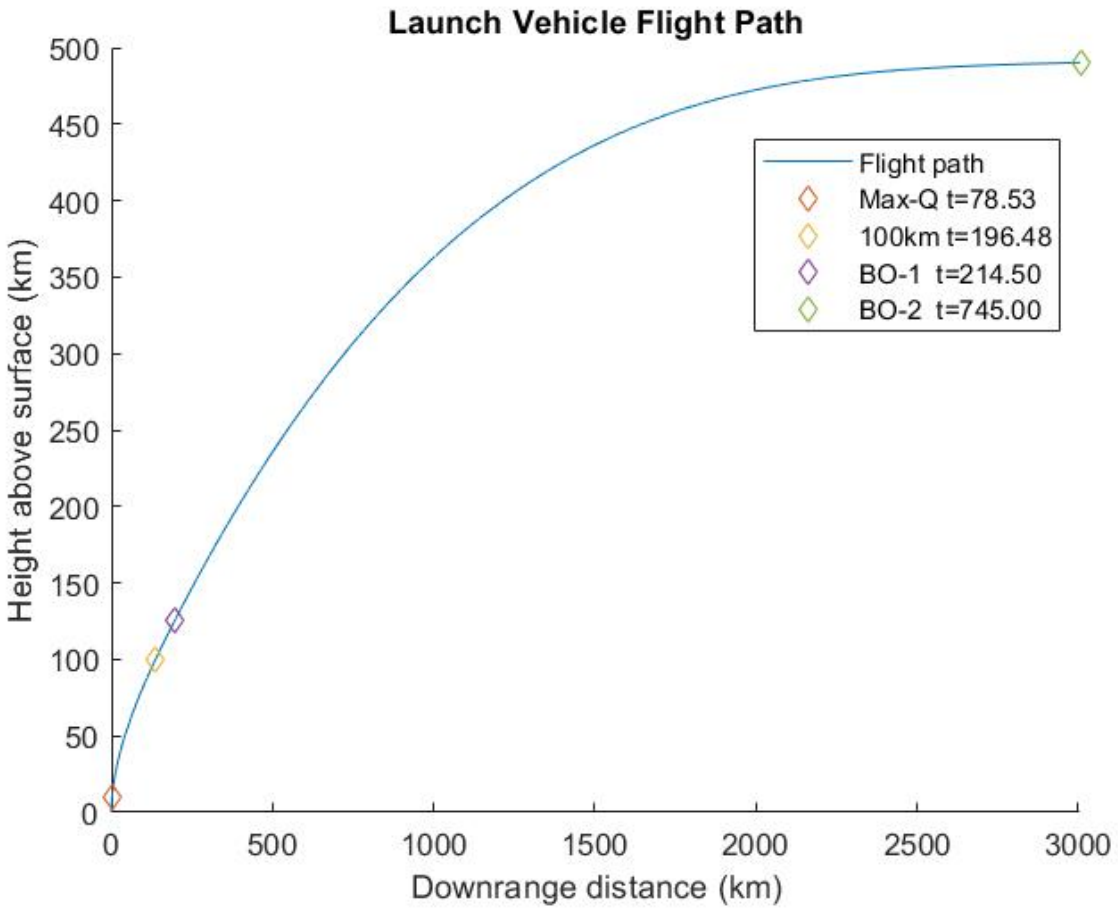

Figure 47: Flight path from liftoff until burnout of the second stage to parking Earth orbit.

The general requirements, milestones and assumptions for this preliminary simulation were to:

- Escape Earth atmosphere using just the first stage.
- Use gravity turn maneuver to slowly reduce the flight path angle to zero and gain enough tangential (circular) velocity to remain in orbit.
- Iteratively obtain the required gravity turn initiation angle and altitude for a successful trajectory.
- No trust vectoring is considered at this stage, vehicle can only burn in the direction of flight.
- Active engine throttling is considered only to limit the maximum acceleration of 6Gs.
- Second stage is not using all the propellant for ensuring the initial parking orbit is close to circular and stable.

Figure 47 presents the preliminary trajectory which was achieved by initiating the gravity turn maneuver at $1.0km$ altitude with a kick of 6.5° from vertical. The first stage's burn time is 214.5 (s) and, after separation, all the engines of the second stage are at maximum throttle for $530.5(s)$ to achieve close-to-circular orbit at approximately a $450-500$ km altitude. The primary flight parameters for the ascend portion are presented in Figure 48.

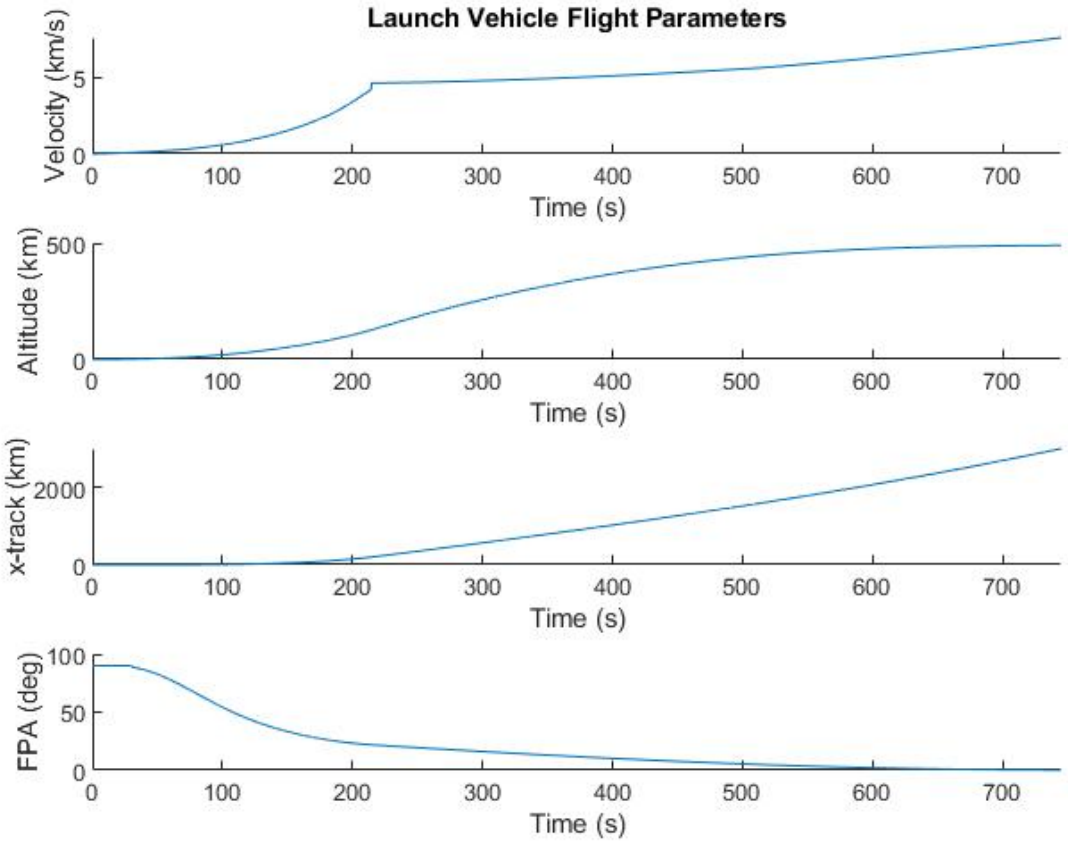

Figure 48: Flight path parameters from liftoff until burnout of the second stage to parking Earth orbit.

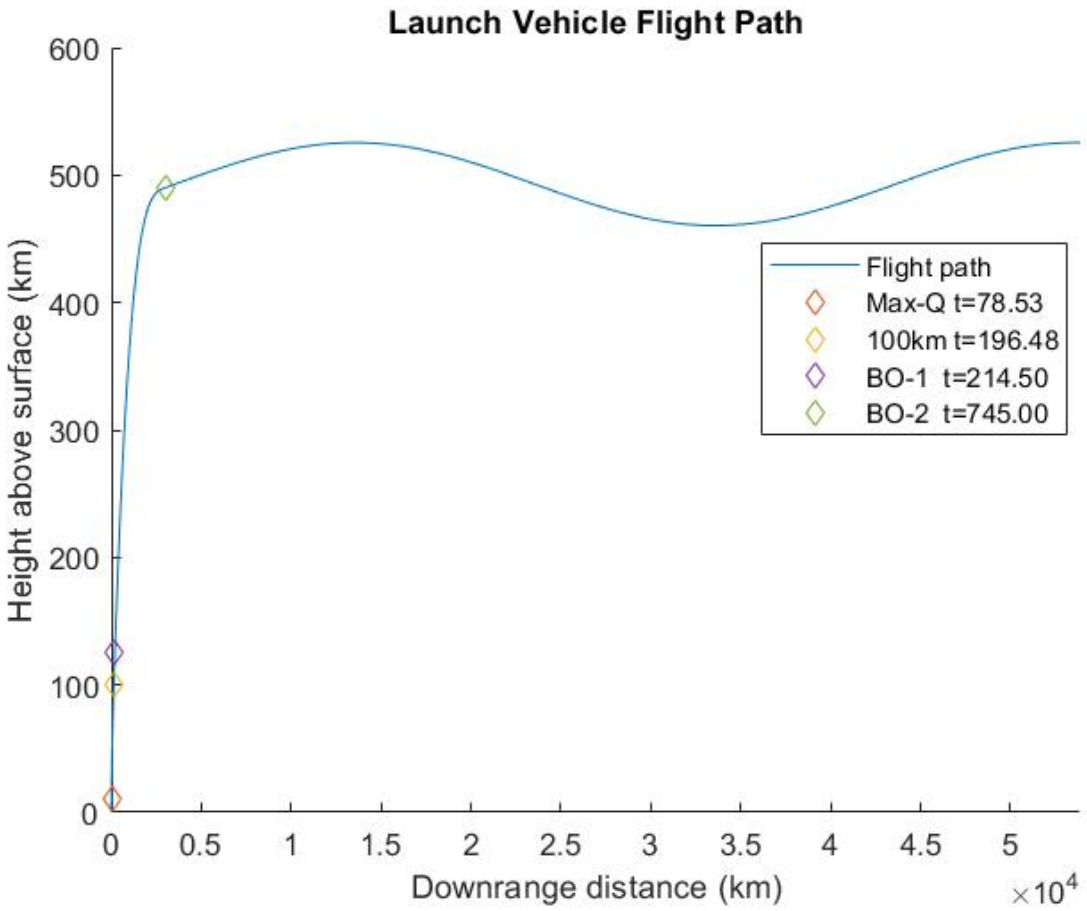

Figure 49: Flight path from liftoff until after burnout of the second stage at parking Earth orbit, propagated for 2 hours.

The maximum dynamic pressure event happens at $t = 78.53$ (s) into the flight, while the first stage is still burning. The max-Q event happens at the altitude of $h = 10.34$ (km) and mach number of 1.20 (see Figure 51). The vehicle reaches edge of space (100 km) at $t = 196.48$ (s) while the first stage is still burning. Also, the structural load at the maximum dynamic pressure event was only 0.877 g.

After reaching the parking orbit, the second step has 15.47 metric tons of propellant left which is over 20.6% of the original propellant. To verify the vehicle can remain in orbit, the simulation was propagated for two more hours with all the propulsion system inactive. The resulting trajectory is shown in Figure 49, with the clear apogee and perigee present after one full revolution around Earth. The achieved orbital speed at this orbit is 7.62 km/s and there is enough propellant left to reach the required orbit.

Acceleration of the vehicle was recovered by taking the resultant velocity profile and approximating acceleration by dividing the change in velocity by the time interval. Since engine throttling was considered to limit the maximum acceleration to 6Gs, the acceleration profile (see Figure 52) during the first was stage burn was capped. Moreover, more engine throttling can be considered for the the first stage of the vehicle needs to be human rated.

Extremely low aerodynamic loss can be explained by a steep flight path the vehicle takes out of the atmosphere. To increase the fidelity of the atmospheric and gravity model a more robust algorithm was using for the simulation during ascent up to the the 200km altitude,

but equations (40),(41) and (42) were than used for the remainder of flight altitudes to decrease computational cost. Note, CFD analysis was only conduced at 1 km altitude and a more robust analysis would be needed to account for addition losses. Moreover, no actual optimization has been attempted for the ascend trajectory and implementation of thrust vectoring as well as multiple events for changing flight path angle can lead to higher fuel efficiency. It is expected for the total velocity achieved in orbit to decrease, or the amount of excess propellant to decrease.

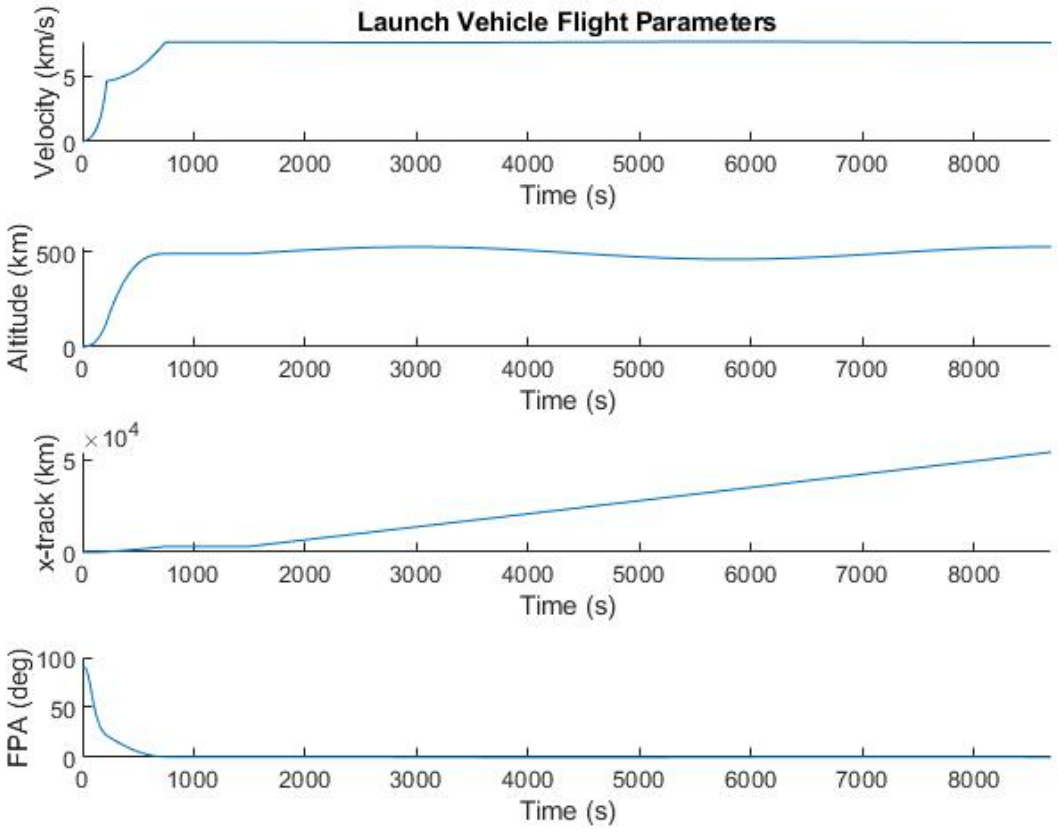

Figure 50: Flight path parameters from liftoff until after burnout of the second stage to parking Earth orbit, propagated for 2 hours.

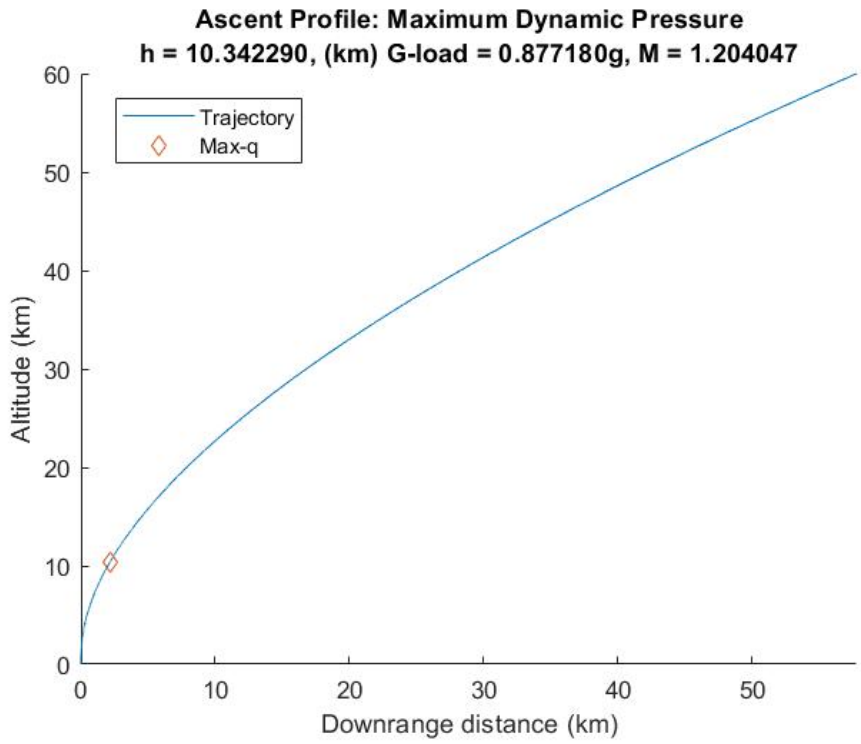

Figure 51: Ascent trajectory: maximum Q event

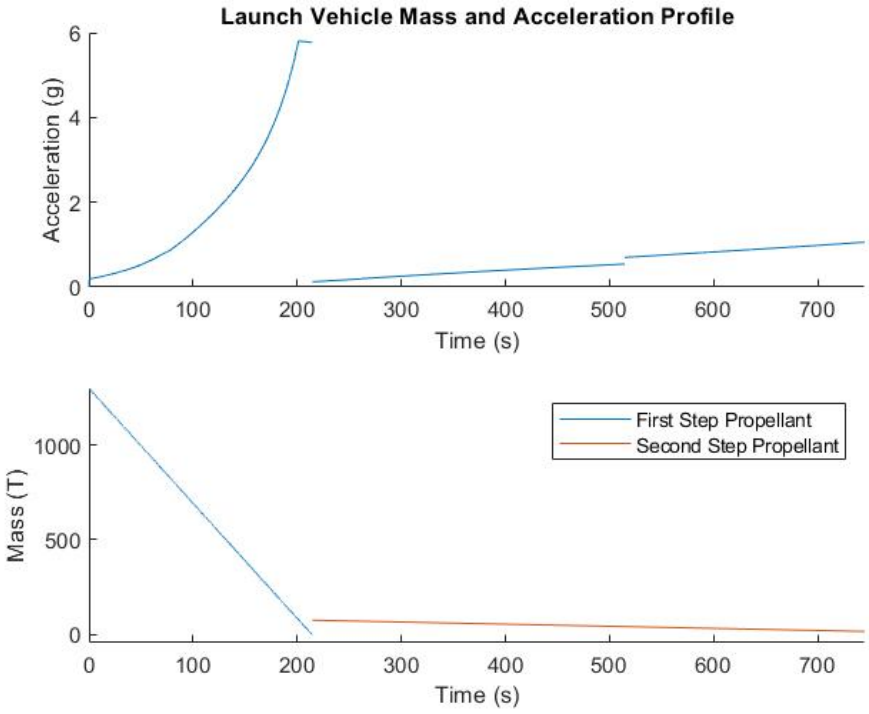

Figure 52: Ascent trajectory: acceleration and vehicle mass profile

Electrical

The electrical system of the Genesis will provide a wide range of digital techniques and transmissions that will power the multitude of functions aboard the launch vehicle. The electrical system will uphold the integrity of the GNC, health monitoring, communications, data acquisitions, fuel pumps, sensor systems, and other electrical powered systems aboard the spacecraft. The main power supply of the electrical system will be powered by Space Information Laboratories' 52 Ah Li-Ion Polymer Intelli-Pack batteries [19]. These batteries were selected for their high energy density, small size and weight, and the advanced Battery Management System (BMS) within each battery [20]. Moreover, these batteries are certified with a mission proven TRL of 9. The figure below displays the battery itself with its system interface diagram.

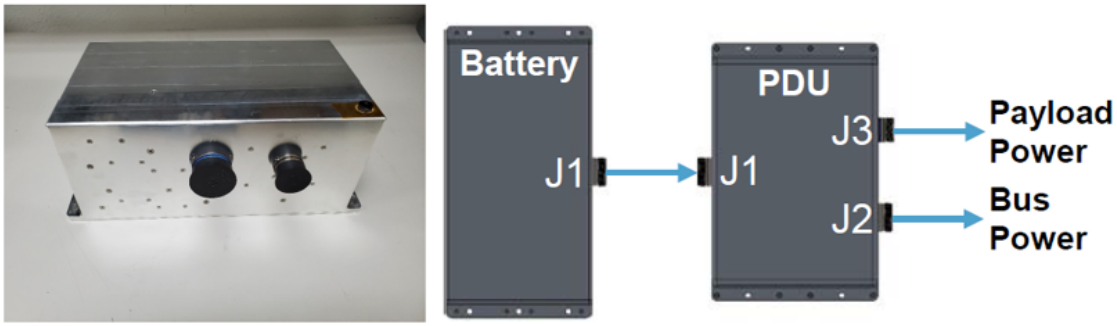

Figure 53: Lithium Ion-Polymer Battery and its Schematic Diagram

One of the main functions of the battery will be to power the control command interface of the launch vehicle. Specifically, two Li-Ion Polymer batteries, two pyrotechnic batteries, the Upper-stage Remote Control Unit (URCU) grounding system, and relative electrical harnesses will provide the power for the multitude of commands the Genesis and control unit will utilize [15]. The following figure outlines the characteristics of the batteries and their power distribution units.

Battery Specifications	
Voltage Range	33.6-22.4 Vdc, Unregulated
Capacity	52 Ah Beginning-of-Life
Cycle Life	1,400 Cycles to End-of-Life
Weight Max	26.2 lbs
Dimensions	7.0" W x 14.0" L x 5.5" H (with flanges)
Steady State Load	104A (2C)
Pulse Load	208A (4C), < 10 second
Telemetry	RS-422 (1 Hz, broadcast only)
Protection and Safety Features	Cell & Pack Level Electrical Protection Vibration and Flame Suppressing Foam Durable Aluminum Enclosure
Temp Range	-40 °C to +60 °C (Environment)

Table 23: Lithium Ion-Polymer Battery Specifications

The URCU of the Genesis will closely resemble the control unit currently used on the Atlas V and will similarly configure the commands at 28 Vdc (nominal). The URCU will consist of at most 16 commands with two master switches that will provide protection against switch

failures due to early commands [15]. Additionally, the URCU will provide 96 outputs of solid-state switching via a MIL-STD-1553B data bus. Standard electrical switching capabilities that will dictate the system are a max allowable current of 8 amps for each master switch and that each command switch will be rated with a max current of 4 amps. Lastly, as stated above, the nominal voltage will be at 28 Vdc with an allowable deterrence from 22-33 Vdc [21]. With these capabilities and guidelines in mind, the Li-Ion Polymer batteries chosen are more than capable to power these electrical systems as seen in Figure 23. The figure below will demonstrate the switch interface schematic that the Genesis will utilize based off the Atlas V model.

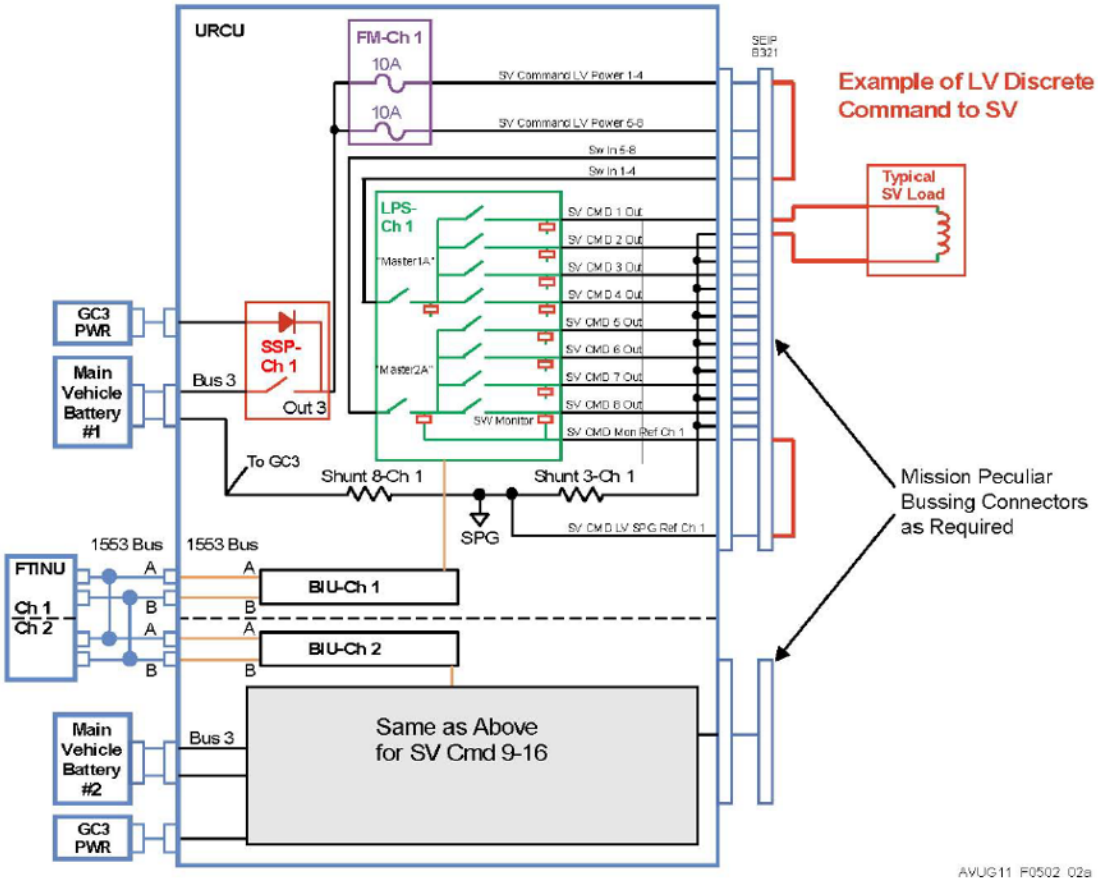

Figure 54: Atlas V Spacecraft Switch Interface Schematic

Further roles of the electrical system within the Genesis include umbilical, separation indicator, and payload separation electrical interfaces. The electrical system that interacts with the umbilical equipment interface plays an important role in the provision of signal paths carried between the spacecraft and the Ground Support Equipment (GSE) system. These power lines will charge the batteries of the Genesis and provide external power to the payload with up to a maximum voltage and current of 128 Vdc and 11 amps, respectively [22]. Without the umbilical attached to the second stage of the Genesis, spacecraft system monitoring will be jeopardized during pre-launch and launch countdown operations. The figure below displays the simplified overall electric command diagram of the Genesis that will be similar to the Ariane 5.

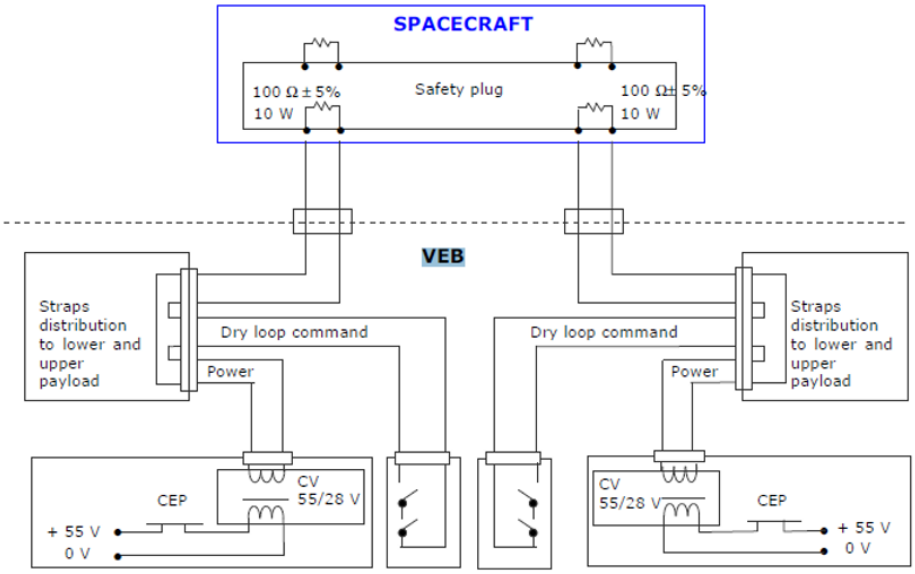

Figure 55: Ariane 5 Electric Command Diagram

Concerning spacecraft to launch vehicle separation and payload adapter systems, redundant pyrotechnic devices based off Atlas V and Delta IV Marmon-type clampbands will be utilized. As expected, these devices will be powered through the electrical system of the Genesis. The separation systems will be initiated by special separation detection circuits that will be required to be single fault tolerant. Thus, with these systems in place, the mission critical functions will be carried out as planned with an ensured notion of protection against inadvertent initiation due to unforeseen automated flight initiations.

Avionics / Guidance, Navigation, and Control

Guidance and navigation is an extremely important component in the launch vehicle design process, and is responsible for ensuring the vehicle stays close to the most efficient predetermined flight path, and is able to finely adjust its attitude for orbital maneuvering without losing control.

Block diagram of Control Structure

The figure below, Figure 56, contains a simplified guidance system structure. The difference desired attitude and the actual trajectory and attitude readouts are inputted into a cluster of PID controllers, which translate the error into control signal inputs. These inputs actuate both the thruster gimbaling and RCS systems, and the resulting sensor readouts are fed back into the system.

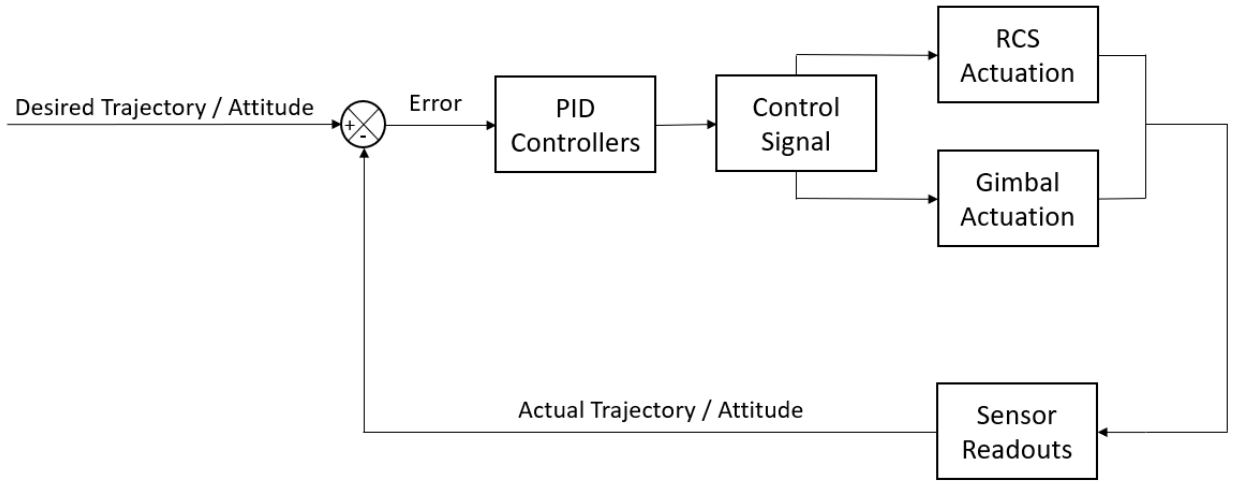

Figure 56: Basic GNC PID Control Diagram

Component Functions

Sensors

In order to determine the launch vehicle's orientation once in space, the spacecraft will be equipped with certain sensors in order to ascertain its orientation and rotational velocities. This launch vehicle will use a system known as a star tracker, pictured in Figure 57 [23]. This device consists of a camera which feeds optical data into a small computer to identify a pattern match to determine the spacecraft's relative orientation [24]. The spacecraft will be outfitted with two of these sensors on opposite sides of the skin in the upper stage of the launch vehicle, to ensure that at least one camera will always have visibility.

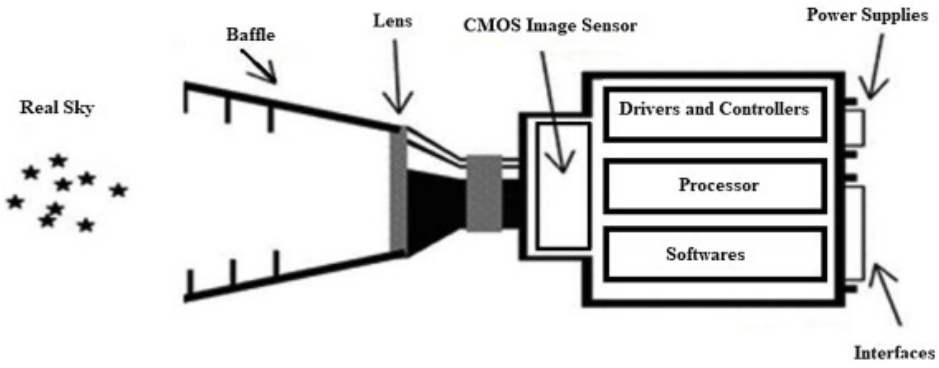

Figure 57: Diagram of a Star Tracker

The second important sensor system onboard the vehicle is the inertial measurement unit (IMU). This sensor rig typically consists of several accelerometesr and a ring laser gyroscope; one such example can be seen in Figure 58 [25]. The accelerometer is responsible for detecting longitudinal accelerations on board the spacecraft. This can be used to determine how long an engine has been burning for an automatic cutoff or compared with the vehicles attitude to determine if an engine needs to gimbal. The gyroscope is responsible for keeping track of the spacecraft's rotational velocity [26]. Two IMU systems will be positioned on board the upper stage of the spacecraft for redundancy.

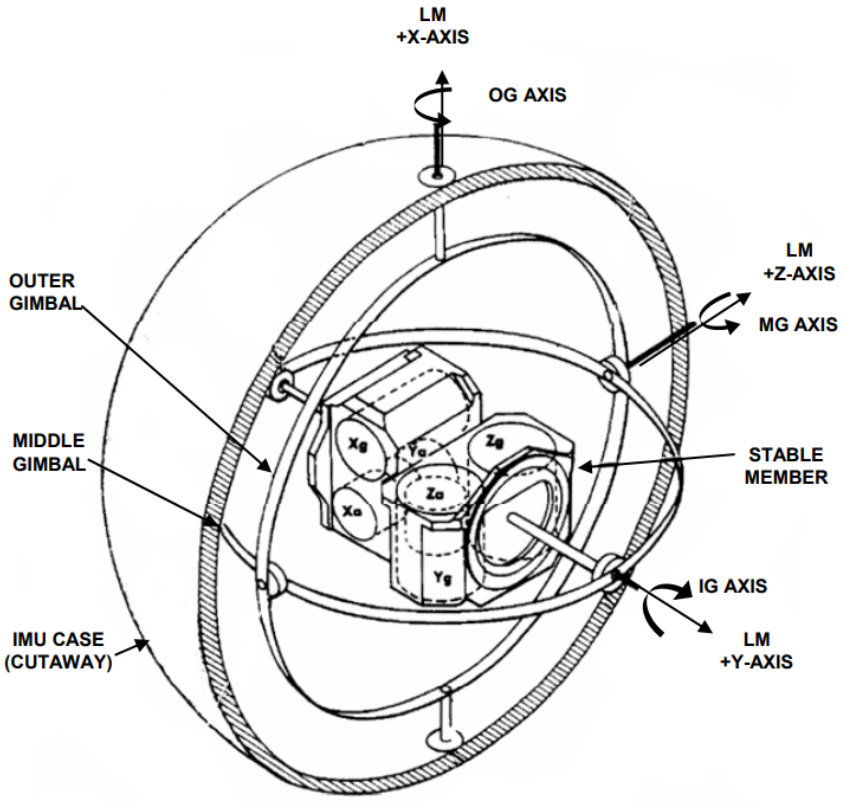

Figure 58: Diagram of the IMU onboard Apollo

Readouts from both of these sensors are combined and compared to gather data regarding the vehicle's position and attitude, which are then fed into the guidance system.

Guidance

The guidance system consists of a system with PID controllers which is able to compare the desired attitude and trajectory of the spacecraft with the sensor indicated attitude and trajectory to determine the error, and the difference between the two. This is fed into PID controllers which are able to tune and apply gains in order to translate the error into logical control signals to actuate gimbaling systems or RCS thrusters.

Control

The resulting control signals are next sent throughout the vehicle to actuators on gimbaling systems or actuators on RCS thrusters which provide the necessary outputs to attempt to correct the vehicle. The sensors are able to detect this adjustment and the process repeats.

Communications

Objective of Communications System

The primary purposes of the communications systems onboard the launch vehicle are to transmit telemetry down to the command center as well as communicate with external satellites to track the location of the launch vehicle. The communication system will be based on the communication system used on the Space Shuttle Orbiter. The launch vehicle will utilize the Tracking and Data Relay Satellites System (TRDSS). TDRSS is a communication signal relay system which provides tracking and data aquisition services between low-earth orbiting spacecraft and control and/or data processing facilities. [27] This will be coupled with NASA's Near Earth Network (NEN), a series of ground stations owned by NASA, commercial entities, and other partners that provide communications and tracking services to missions operating in the near earth region, including Earth-orbiting spacecraft. [28] A diagram of the TRDSS can be found in Figure 59

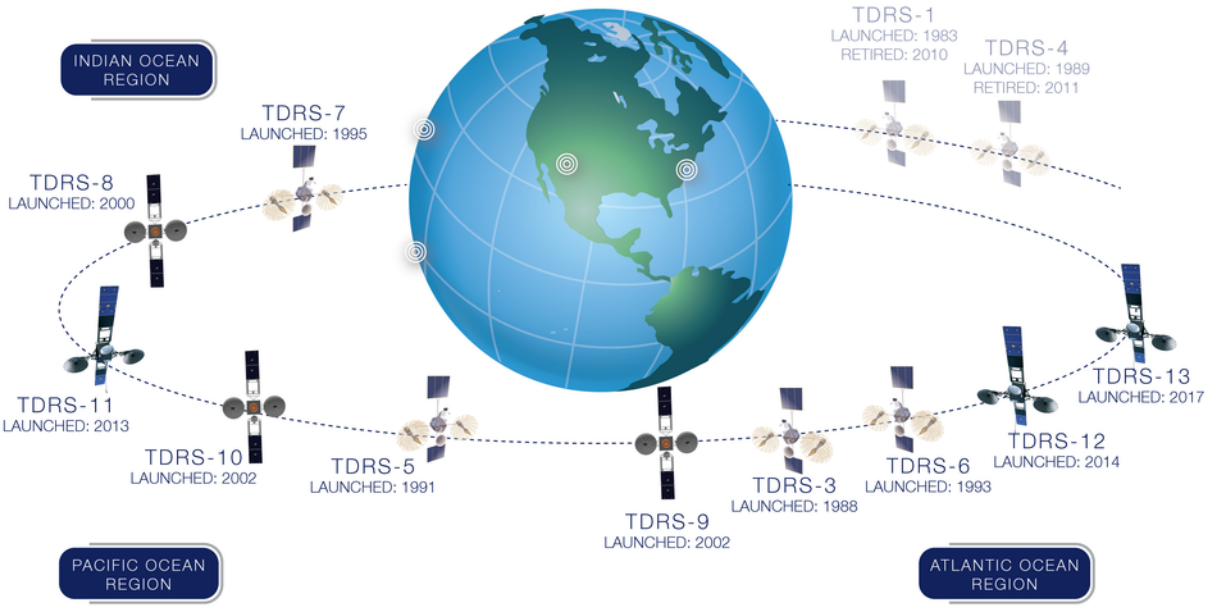

Figure 59: TRDS fleet diagram

Requirements

All communication systems onboard the launch vehicle will follow the Consultative Committee for Space Data Systems (CCSDS) standards. CCSDS standards are for space-related data and communications systems. These are internationally agreed standards that lower operating costs and risks, and provide interoperability and innovative capabilities for current and future NASA missions. Besides being essential for international interfaces, they provide those same benefits between NASA organizations and contractors. [29]

The three primary systems that need to communicate on the launch vehicle are the Flight Termination System (FTS), Guidance, Navigation, and Control systems, and video feed systems. The data from each of these systems also needs to be transmitted down to mission control.

Uplink / downlink

Transmission band(s)/ Frequency ranges

The current Tracking and Data Relay Satellite configuration consists of 10 in-orbit satellites (four first generation, three second generation and three third generation satellites) distributed to provide near continuous information relay service to missions. [30] Together these satellites provide data transmission on S, Ku, and Ka bands. The frequencies that the satellites operate on are shown in Table 24.

Transmission Band	Frequency Type	Frequency (GHz)
S-band Single Access	Forward	2.025 - 2.120
S-band Single Access	Return	2.200 - 2.300
S-band Multiple Access	Forward	2.1064
S-band Multiple Access	Return	2.2875
Ku-band Single Access	Forward	13.775
Ku-band Single Access	Return	15.0034
Ku-band Ground Link	Uplink	14.6 - 15.25
Ku-band Ground Link	Downlink	13.4 - 14.05
Ka-band Single Access	Forward	22.55 - 23.66
Ka-band Single Access	Return	25.25 - 27.50

Table 24: TRDSS transmission bands and frequency ranges [31]

The other component of this communication network is the Near Earth Network. The locations of these ground stations as well as their operating transmission bands are shown in Table 25.

Facility	Location	Transmission Band	Assets
KSAT Singapore	Singapore, Malaysia	S/X Band	9.1m
KSAT Svalbard	Svalbard, Norway	S/X Band	11.3m/11.3m/13m
KSAT TrollSat	Antarctica	S/X Band	7.3m/7.3m
NASA Alaska Satellite Facility	Fairbanks, Alaska	S/X Band	11.3m/11m/9.1m
NASA Kennedy Uplink Station	Cape Canaveral, Florida	S-band	6.1m
NASA McMurdo Ground Station	Antarctica	S/X Band	10m
NASA Ponce de Leon Station	New Smyrna Beach, Florida	S-band	6.1m
NASA Wallops Ground Station	Wallops Island, Virginia	VHF, S/X Band	11m/5m
NASA White Sands Ground Station	Las Cruces, New Mexico	VHF, S/Ka Band	18.3m
SANSA Hartebeesthoek	Hartebeesthoek, South Africa	S/X Band	12m/10m
SSC Kiruna	Kiruna, Sweden	S/X Band	13m/13m
SSC Santiago	Santiago, Chile	S Band	9m/12m/13m
SSC Space US North Pole	North Pole, Alaska	S/X Band	5m/7.3m/11m/13m
SSC Space US Dongara	Dongara, Australia	S/X Band	13m
SSC Space US South Point	South Point, Hawaii	S/X Band	13m/13m

Table 25: Near Earth Network ground station locations and assets

Finally, the FTS will use two L-band antennas operating in the 1-2 GHz frequency range.

Equipment

The communication equipment located on the launch vehicle will be located in the second stage, with video cameras located in positions throughout the launch vehicle. There are four S-band PM antennas located on the fairing of the launch vehicle separated by 90 degrees. These S-band PM antennas communicate with one of two S-band PM transponders which in turn communicate with one of two network signal processors. These two systems are redundant and only one is used at a time. The S-band PM system is responsible for transmitting information to or receiving information from the ground or the TRDSS. Two S-band FM antennas are located on the payload bay separated by 180 degrees. These communicate with two S-band FM transmitters which in turn communicate with 2 FM signal processors. The S-band FM system is responsible for transmitting data to ground stations. [32]

There is one Ku-band antenna located in the payload bay that is responsible for communications with the TRDSS when the payload is being deployed and for radar operations. The radar portion is used to skin-track satellites or payloads in orbit to facilitate a rendezvous. There is a single Ku-band transponder that communicates with the network signal processors. [32]

The payload communication system is used to transfer information between the launch vehicle and its payload or payloads. It supports hardline and radio frequency communications with a variety of payloads. The system is used to activate, check out and deactivate attached and detached payloads.The basic elements in the payload communication system are the payload interrogator, payload signal processor, communication interface unit, payload data interleaver, pulse code modulation master unit, payload patch panel, payload recorder and payload MDMs 1 and 2. These elements are in the avionics bay. [32]

Four L-band antennas are located on the fairing of the launch vehicle separated by 90 degrees. These communicate with the two FTS units and are responsible for communicating with GPS satellites to determine the location of the launch vehicle.

Data and Communications Block Diagram

Below are the block diagrams for the S-band and Ku-band systems onboard the launch vehicle as well as the block diagram for the ground portion of the TRDSS.

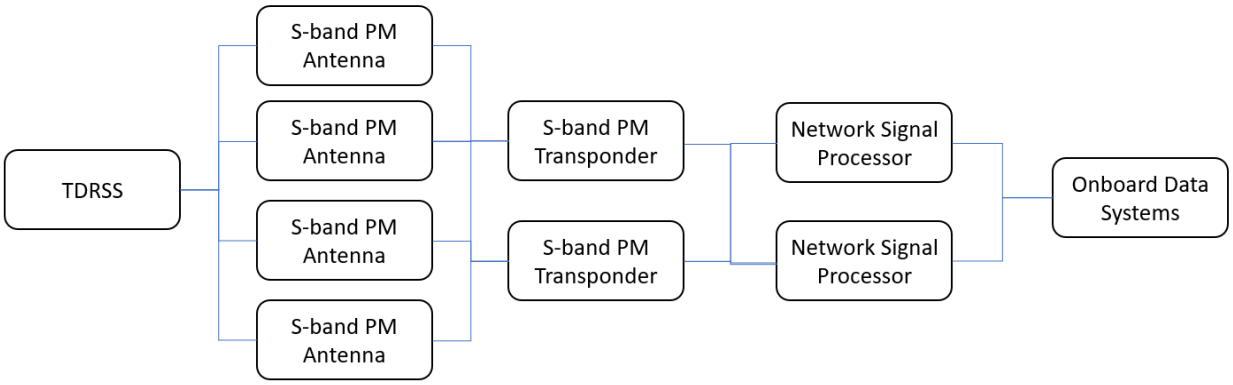

Figure 60: S-band PM diagram

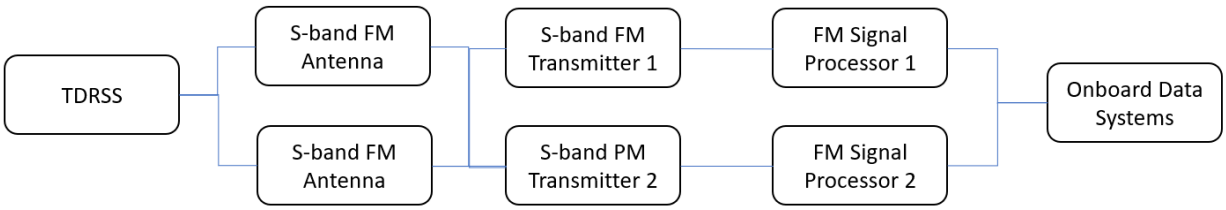

Figure 61: S-band FM diagram

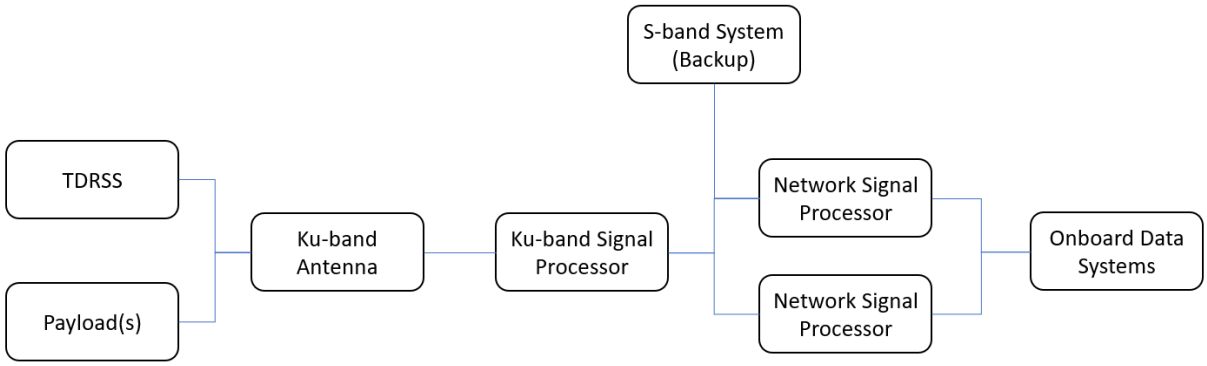

Figure 62: Ku-band diagram

Health monitoring of vehicle

Objective of health monitoring system

Health monitoring technologies are critical components in the success and integrity of launch vehicles. A network of sensors and data interpretation equipment will comprise the health monitoring system of the Genesis and will have the ability to read and interpret information regarding the state of the vehicle. The Genesis will be equipped with a dedicated network of Fiber Optic Sensing System (FOSS) sensors that will be connected to a Data Acquisition System (DAS). The Fiber Optic sensors will be able to detect pressure, temperature, stress, strain, and vibrational changes which will allow the ground control team to actively monitor these parameters. In conjunction with Fiber Bragg Gratings (FBG), the FOSS sensors have the ability to contain 100s of sensor onto singular optical fibers that allow for an increase in structural and system efficiency of the Genesis [33]. The following figure provides the schematic of FBG fabrication.

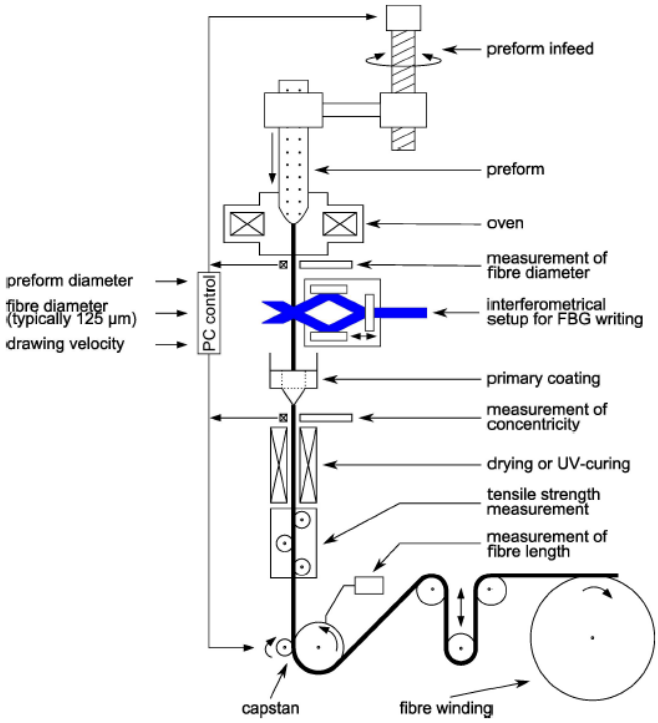

Figure 63: FBG Fabrication

Moreover, due to the lightweight capability and highly defined sensing of FOSS sensors, the structural design and data system integration costs can be reduced for future adaptations of the Genesis. The importance of this is that this will allow for a further understanding of the structural performance throughout the Genesis's mission life cycle and allow for increased capability of structural parameters.

Requirements and Sensors

The highly accurate method of cryogenic liquid level-sensing (CryoFOSS) will be utilized within the fuel tanks and pumps of the Genesis. The CryoFOSS sensors will measure liquid

levels within the tanks by discerning between liquid and gas states along the fibers of the sensors [33]. The figure below shows the FOSS approach to liquid level-sensing.

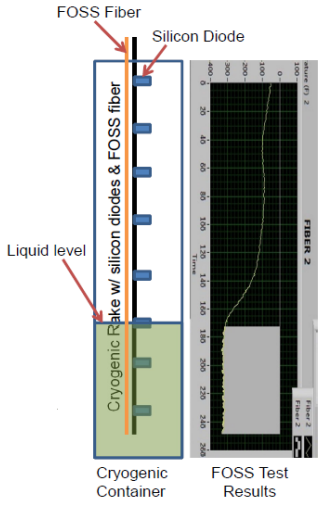

Figure 64: CryoFOSS Measuring Technique

With this method, the Genesis's cryogenic fuel levels will be measured with high precision and accuracy and will allow the ground control team to actively observe the rocket's fuel status. Furthermore, the DAS will be able to actively interpret the fuel levels of the tanks and translate them for the Genesis's main flight computers. In doing so, the FTINU and on board flight computers can apply necessary trajectory maneuvers in order to mitigate extensive fuel level discrepancies.

An additional method that will be implemented within the vehicle will be the Hybrid fiber optic sensing system (hyFOSS). This method will utilize traditional wavelength division multiplexing (WDM) in conjunction with the highly accurate technique of Optical Frequency Domain Reflectometry (OFDR) [34]. By utilizing these techniques in conjunction with hyFOSS sensors, high frequency vibrational changes throughout the vehicle can be effectively measured and translated by the DAS. Thus, thereby making these fiber optic accelerometers great tools in measuring and detecting vibrations and acoustic signals of structures within the Genesis [35]. The figure below presents an example application of the hyFOSS technique.

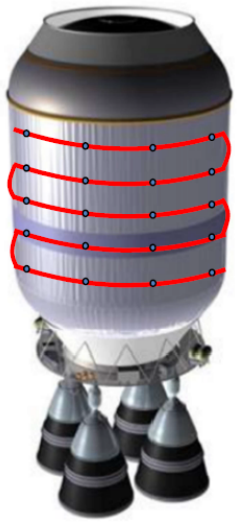

Figure 65: hyFOSS Fiber Layout Example

Furthermore, the FOSS sensors will be able to actively monitor strain and pressure. An advantage of FOSS sensing over traditional strain gauges is the reduction in size and complexity. This reduction can be attributed to the large decrease in the number of wires needed to contain fibers for FOSS sensors. Therefore, the FOSS sensors that will act as strain gauges within the vehicle will placed within the stages, stage joints, and fuel tanks of the Genesis [33]. The FOSS sensors will also allow for the detection of shape sensing for increased vehicle control and COPV strain with temperature monitoring. Lastly, FOSS pressure sensors and pressure transducers will monitor pressure changes within the fuel tanks and turbopumps. By monitoring the propellant head pressures on the tanks, the FOSS sensors will be able to measure the mass of the remaining propellants and translate this data to the FTINU in order to calculate mass imbalances between the fuel tanks. In all, by utilizing FOSS sensors, the detection of abnormal pressure, stiffness, and strain changes can be quantified and interpreted by the DAS in order to apply necessary re-balancing of the launch vehicle's structural integrity. Concerning the monitoring of the payload compartment, acoustic panels will be attached to the payload fairing and the first few bays of the top cone. These acoustic panels will be riddled with FOSS sensors and will allow the system to reach acceptable sound pressure levels through communication of the DAS and main launch vehicle computers.

Data Handling System

The telemetry Data Acquisition System (DAS) that will be used on the Genesis will monitor several hundred vehicle parameters before launch and throughout all phases of flight. The figure below presents the telemetry data acquisition system configuration block diagram. This diagram was closely based off the DAS of the Atlas V.

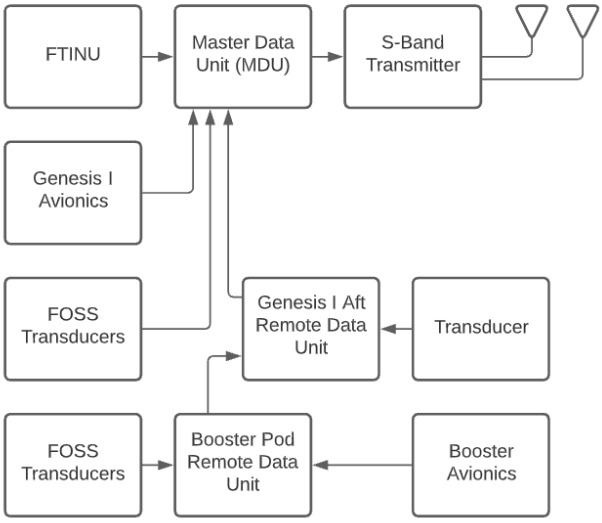

Figure 66: Telemetry Data Acquisition System Configuration

The DAS telemetry system will utilize a Master Data Unit (MDU) and two Remote Data Units (RDUs) with one on the booster and the other on the end of the second stage [15]. The MDU will be in charge of receiving and formatting data from the RDUs and the Fault Tolerant Inertial Navigation Unit (FTINU). With this system in place, the ground control team will be able to actively process data from the ground telemetry network and observe necessary data exchanges between automated and human controlled decisional processes.

Flight termination system (FTS)

Requirement of system and overview

The launch vehicle will utilize the Autonomous Flight Termination System (AFTS) developed by NASA. This independent, self-contained system uses redundant flight processors that track the launch vehicles location using GPS/IMU navigation systems. The software has configurable rule sets that are uploaded pre-flight and should the launch vehicle trigger one of these conditions, the AFTS will terminate the flight automatically.[PDF Source] This system has been proven on other launch vehicles as reliable and the flight software, called Core Autonomous Safety Software (CASS), and has been approved for operational use at both the eastern and western launch ranges. [36] An overview of the AFTS is shown in Figure 67.[37]

Figure 67: Overview of AFTS

Adherence to requirements

This system has been designed to adhere to the requirements in the Flight Termination Systems Commonality Standard, and the software has been approved for operational use at

our launch site. Further flight tests will be conducted to qualify the system in its entirety.

Components,locations, and proposed outcomes

The system will use four independent flight processors housed in the second stage in two separated hardware units that receive input from GPS and IMU navigation sensors. The system also utilizes L-band and S-band antennas, a vehicle battery, and a power distribution box. All of these systems (GPS, INS, Processors) will be COTS. The AFTS hardware and the functional baseline are shown in Figures 68 and 69.[38]

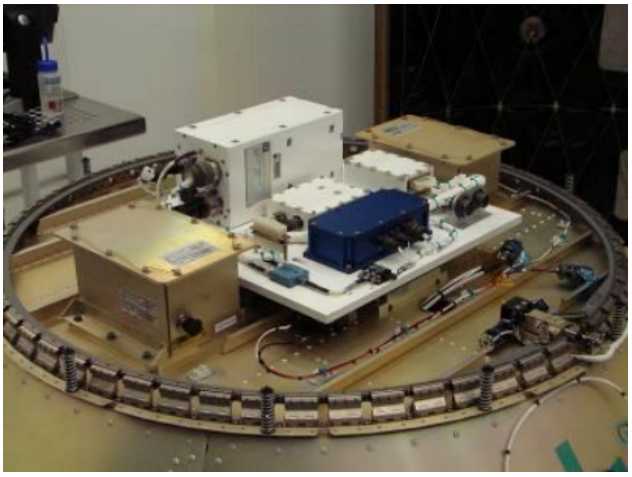

Figure 68: AFTS hardware

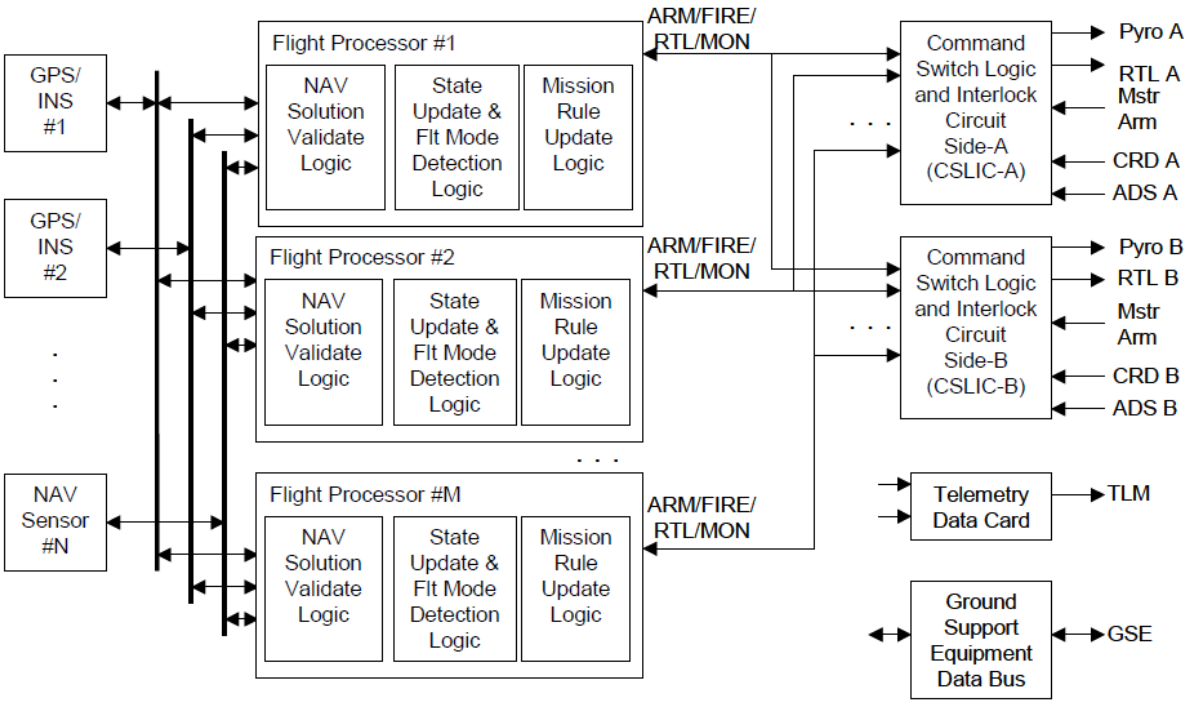

Figure 69: AFTS Block Diagram

In the event of the AFTS terminating the flight, explosive charges lined on the propellant tanks will detonate, igniting the propellants and destroying the launch vehicle.

Flight Test Plan

Figure 70 shows the timeline for the project.

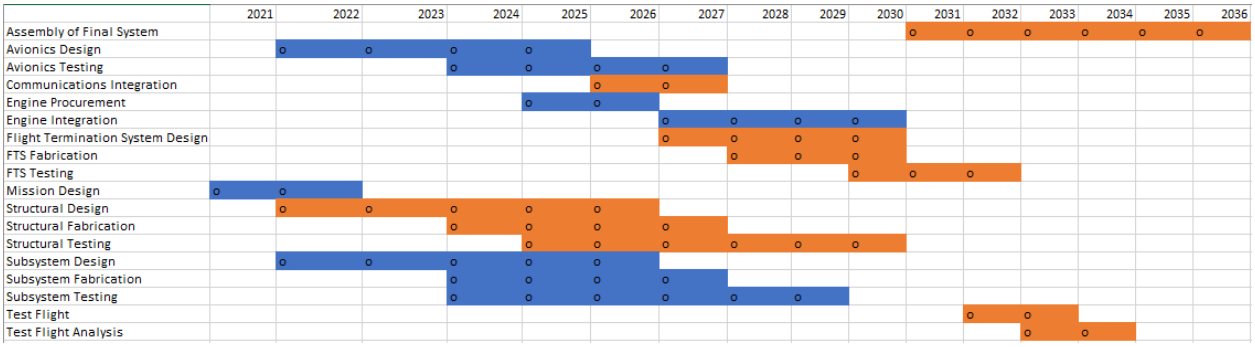

Figure 70: Project Timeline

Testing for the components of each system is scheduled to take place as soon as fabrication of the components allows. This testing will continue throughout the fabrication process, testing each major addition as they are fabricated until the final, complete system undergoes testing. Once all system components have been tested, the final structural and interoperability tests will take place with the flight termination system tests being the final tests before flight testing.

The test flight period is scheduled to start in 2032, just after the flight termination system testing is completed. This time frame was selected because all fabrication of the launch vehicles as well as all component testing will have been completed. Two test flights are scheduled to take place, one in 2032 and the other in 2033 as well as a two year period of post test analysis following the first test. This allows for any issues discovered during the first test flight to be corrected and tested on the second flight. Should no problems occur on the first flight, a second flight will still be conducted to ensure reliability.

Final Overview of Flight Systems

Blue Oregano's launch vehicle as part of the Advanced Rocket Trade Enterprise: Mars International Space Station (ARTEMISS), Genesis 1, will launch 50 metric tons of uncrewed payload into a low earth orbit of approximately 900km above the Earth's surface in fiscal year 2036. It will be comprised of two stages, a lower stage responsible for the majority of the flight through atmosphere, and an upper stage responsible for the latter portions of escaping Earth's atmosphere and orbital maneuvering into a circular 900km altitude orbit. The rocket will maintain an optimum predetermined flight trajectory using various gimbaling and RCS systems to assist with stability, while communicating with ground stations throughout its flight. All systems on the vehicle feature some kind of redundancy including flight termination systems and health monitoring systems to ensure the safety of the vehicle as well as that of any associated parties. This vehicle was designed in order to maximize efficiency for the prescribed mission and to establish Blue Oregano as the primary launch provider for the developing space trade industry.

References

[1] "Human-Rating Requirements for Space Systems," .

[2] Lambright, W. H., "Maintaining Momentum: Robert Lightfoot as NASA's Acting Administrator, 2017–2018," *Space Policy*, Vol. 48, 2019, pp. 87–90.

[3] Jones, H., "Much Lower Launch Costs Make Resupply Cheaper Than Recycling for Space Life Support," 47th International Conference on Environmental Systems, 2017.

[4] Owens, A. and de Weck, O., "Sensitivity Analysis of the Advanced Missions Cost Model," 46th International Conference on Environmental Systems, 2016.

[5] Jones, H. W., "Estimating the life cycle cost of space systems," 45th International Conference on Environmental Systems, 2015.

[6] Wall, M., "NASA's Kennedy Space Center (KSC) Information," Space.com, 2018.

[7] Edberg, D. and Costa, W., *Design of Rockets and Space Launch Vehicles*, American Institute of Aeronautics and Astronautics, Inc., 2020.

[8] Thomson, W., "Introduction to space dynamics," *NASA STI/Recon Technical Report A*, Vol. 87, 1986, pp. 18420.

[9] Hinkel, H., Strube, M., Zipay, J. J., and Cryan, S., "Technology Development of Automated Rendezvous and Docking/Capture Sensors and Docking Mechanism for the Asteroid Redirect Crewed Mission," NASA, 2015.

[10] Farnham, S., Garza., J., Castillo, T., and Lutomski, M., "METHODOLOGY FOR DEVELOPING A PROBABILISTIC RISK ASSESSMENT MODEL OF SPACECRAFT RENDEZVOUS AND DOCKINGS," NASA Johnson Space Center, 2011.

[11] Brandon N. Dick, N. M. and Rupp, T., "Capture Latch Assembly for the NASA Docking System," 2018.

[12] "RCS Thruster - FLOWN Artifact," Historic Space Systems, 2018.

[13] Sorokin, V., "Yantarnaya istoriya," Novosti Kosmonavtiki, 1997.

[14] Dumoulin, J., "REACTION CONTROL SYSTEM," NASA, 2000.

[15] Wilkins, M. and Sowers, G., "Atlas V Launch Services User's Guide," 2010.

[16] Rocketdyne, A., "RL10 Propulsion System," 2019.

[17] Corp, S. E. T., "STARSHIP USERS GUIDE," SpaceX, 2020.

[18] Dumoulin, J., "Main Propulsion Systems," .

[19] Burke, E., "Li-Ion Intelligent and Safe Battery Technology Innovation Leader," NASA MSFC Li-Ion Battery Workshop, 2020.

[20] Burke, E., "Lithium Intelli-Pack® Battery: Smart, High Energy and Safe for multiple Aerospace platforms," AIAA Prupulsion and Energy Forum, 2019.

[21] Alliance, U. L., "Delta IV Launch Services User's Guide," 2013.

[22] Lagier, R., "Ariane 5 User's Manual," 2020.

[23] Sarvi, M., Moghadam, B., Abolghasemi, M., and Hoseini, H., "Design and implementation of a star-tracker for LEO satellite," ScienceDirect, 2020.

[24] Stoyanova, S., "Star Trackers Light the Way," NASA, 2008.

[25] "Guidance, Navigation, and Control (GNC), Efficient, Responsive, and Effective," George C. Marshall Space Flight Center.

[26] "Sensors, Mars Reconnaissance Orbiter," NASA.

[27] "NASA's Tracking and Data Relay Satellites," .

[28] "What kind of communications do NASA mission's require?" .

[29] "CCSDS Standards," .

[30] "Tracking and Data Relay Satellites," .

[31] "NASA Space Network Project Operations Management: Past, Present and Future for the Tracking and Data Relay Satellite Constellation," .

[32] "Orbiter Communications," .

[33] Chan, P., "Fiber Optics Sensing System (FOSS) at NASA Armostrong Flight Research Center (AFRC): Summary and Recent Deployments," 3rd Edward Technical Symposium, 2018.

[34] Pen a, F. and Richards, L., "Fiber Optics Sensing System (FOSS) Technology," NASA Armstrong Flight Research Center, 2015.

[35] Obringer, L., "Orthogrid trial panel for Vulcan Rocket propellant tank," 2017.

[36] Philman, A., "Engineer Manages Certification of Autonomous Flight Termination System Software for Launches at Florida Spaceport," 2017.

[37] Valencia, L., "Autonomous Flight Termination System (AFTS)," 2017.

[38] Bull, J. B. and Lanzi, R. J., "An Autonomous Flight Safety System," 2007.

A 2-D Drawings / 3-D CAD of vehicle segments and overall vehicle with dimensions.

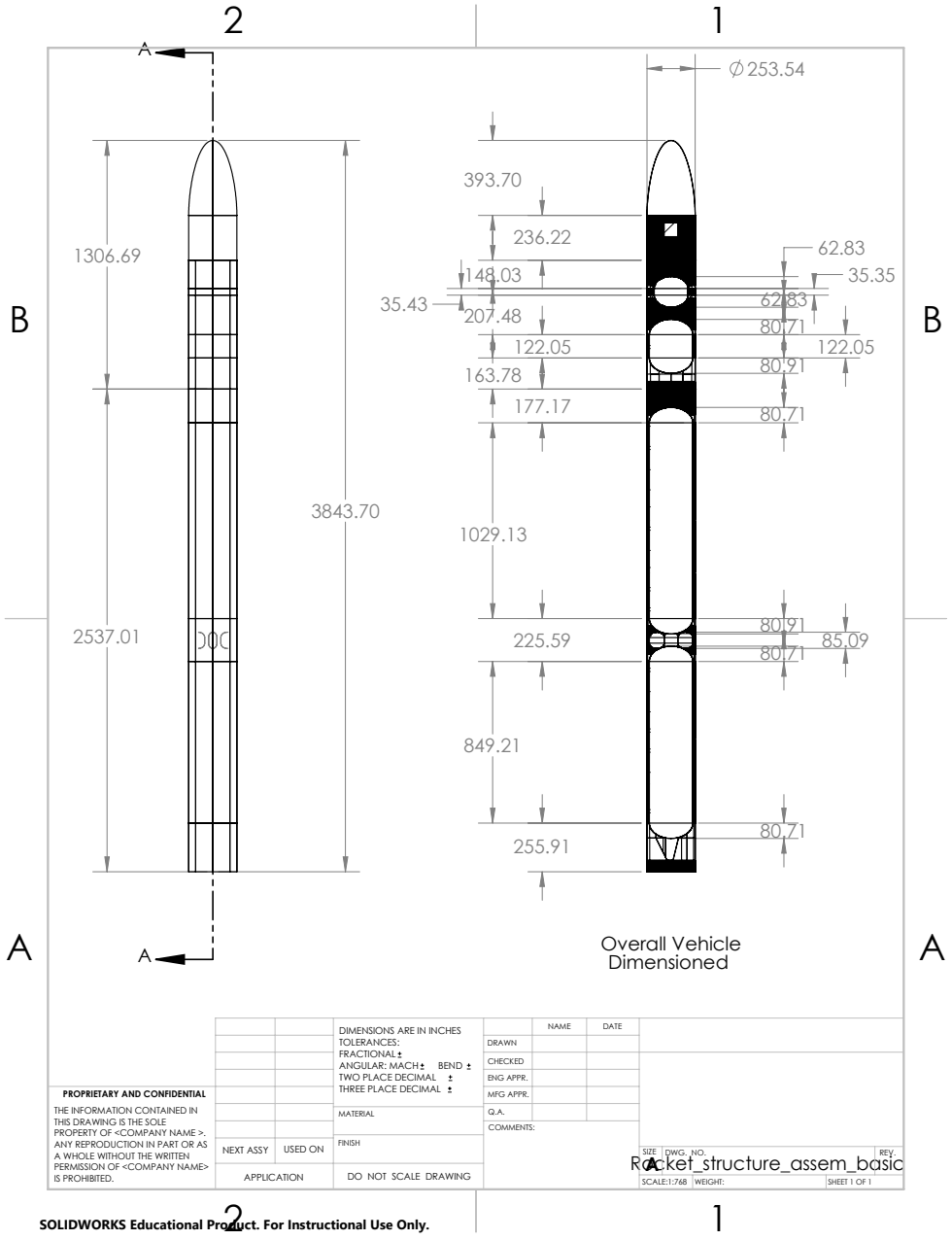

Figure 71: Overall Vehicle with Dimensions

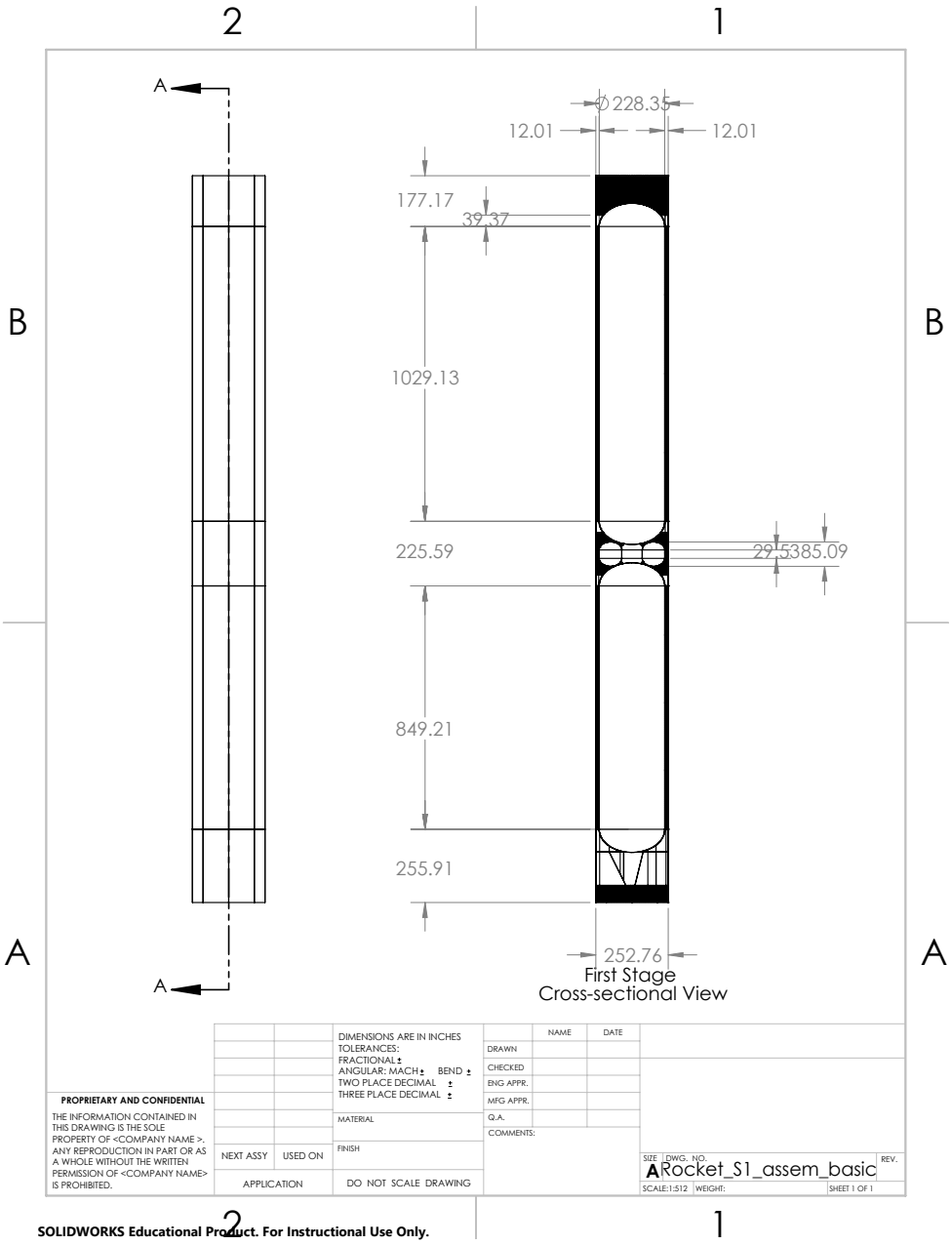

Figure 72: First Stage 2D Drawing

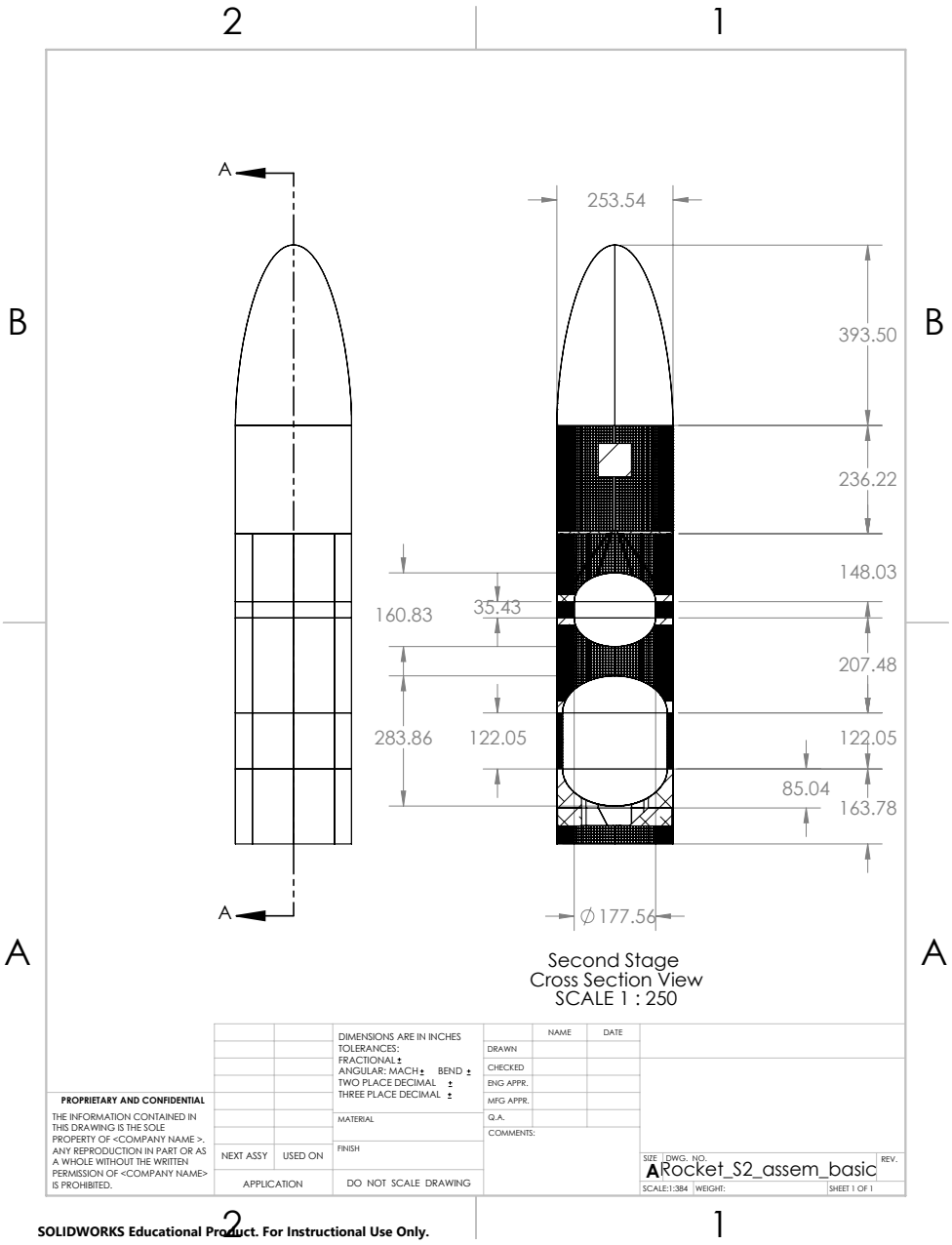

Figure 73: Second Stage 2D Drawing

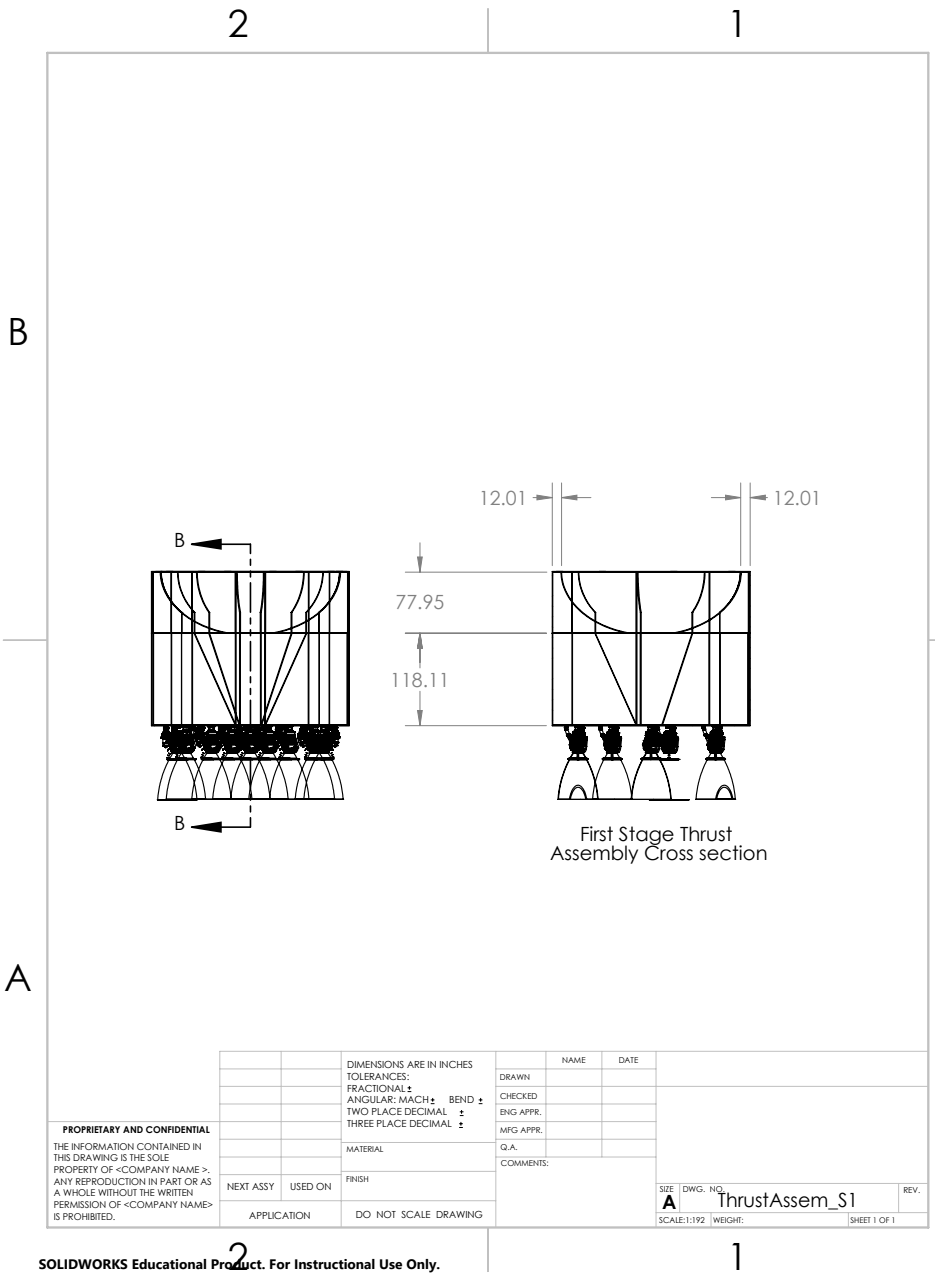

Figure 74: First Stage Thrust Assembly 2D Drawing

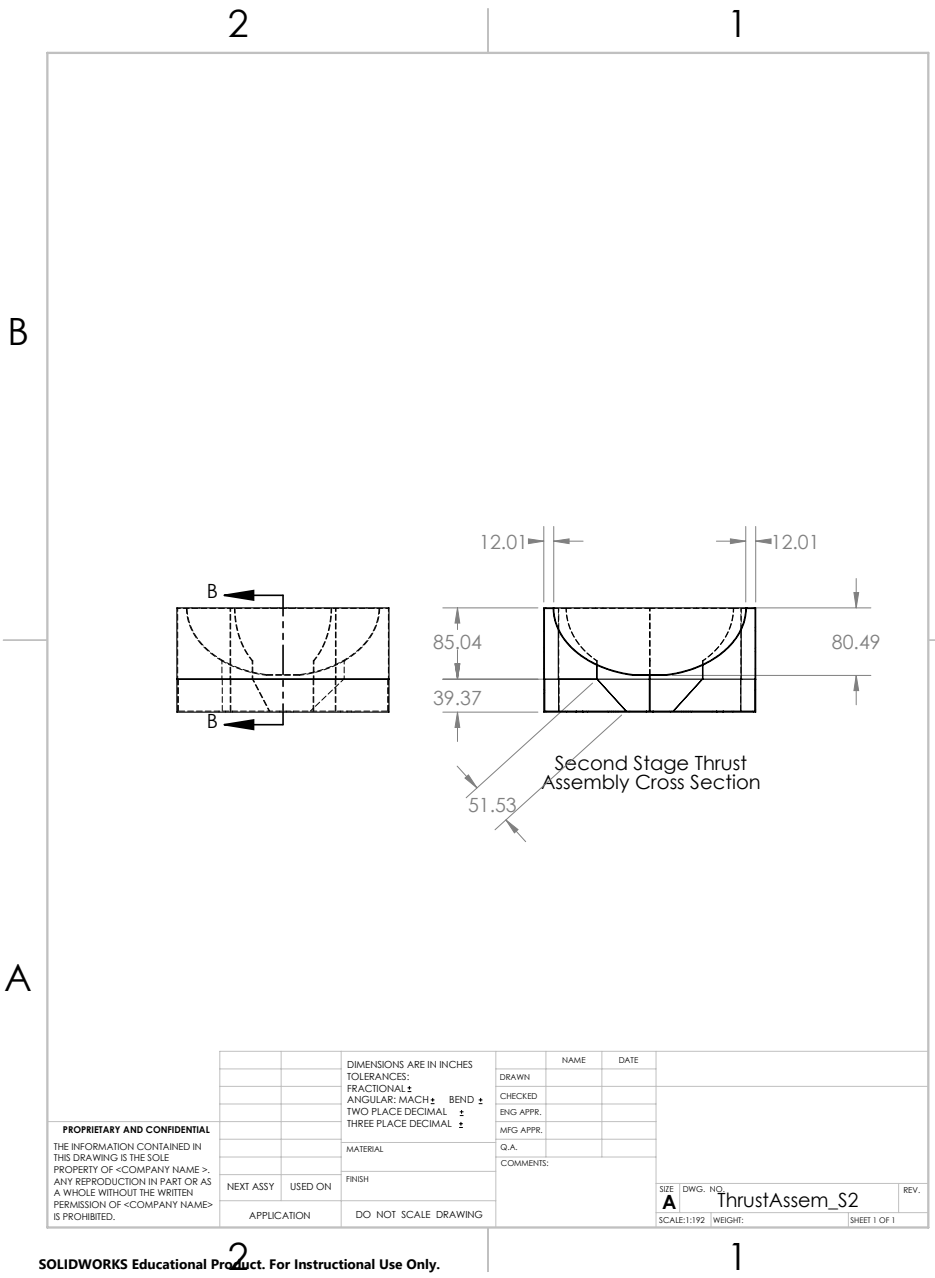

Figure 75: Second Stage Thrust Assembly 2D Drawing

A MATLAB Scripts and Routines

```
%{
This function takes the total required delta v
for the vehicle, dv losses and breaks it down for each stage
accodring to burnout altitudes.

alpha and beta are parameters used for allocating dv loses
must always be:
sum(alpha_i)    == 1
sum(beta_i)     == 1
Think of alpha_i as how much total dv will be provided by step i
and beta_i would be the amount of dv_tot.loss absorbed by step i

Author: Yevhenii (Jack) Kovryzhenko
yzk0058@auburn.edu
4/13/2021
%}
function dv_design = alloc_dv(dv_tot,Nsteps,alpha,beta)
CheckSize(Nsteps,[1,1]);
CheckSize(alpha,[Nsteps,1]);
CheckSize(beta,[Nsteps,1]);
%Required inputs:
%{
dv_tot.bo           total delta v after firing all n steps
dv_tot.loss         total dv losses
alpha and beta are are prameters used for allocating dv loses for each step
%}
dv_design   = zeros(length(Nsteps),1);


if sum(alpha) ~= 1.0
        fprintf("\nERROR, sum(alpha_i) must always be 1.0, sum(alpha_i) = %f",sum(alpha))
        return;
end
if sum(beta) ~= 1.0
        fprintf("\nERROR, sum(beta_i) must always be 1.0, sum(beta_i) = %f",sum(beta))
        return;
end
for i = 1:Nsteps
    dv_design(i,1)   = alpha(i,1)*dv_tot.bo + beta(i,1)*dv_tot.loss;
end
end
```

```
%{
This is short-hand approximation for calculating
effective Isp value (s)

Book: pg. 259
Author: Yevhenii (Jack) Kovryzhenko
yzk0058@auburn.edu
4/13/2021
%}
function Isp_eff = calc_Isp_eff(Isp_vac,Isp_sl)
Isp_eff = (Isp_sl+2*Isp_vac)/3;
end
```

```
%{
This function is  used to run the
GLOM optimization

Author: Yevhenii (Jack) Kovryzhenko
yzk0058@auburn.edu
4/13/2021
%}
function [m_p,m_pl,m_0,m_f,staging_dv_km,liftoff_mass_T] =
calc_mass(Nsteps,m_pl_f,dv_tot,sigma,Isp,alpha,beta)
%get design dv for each step:
dv_design = alloc_dv(dv_tot,Nsteps,alpha,beta);
%optimize the mass ratio:
mu = Optimize_mu(Nsteps,dv_design,sigma,Isp);

if mu < 0
    fprintf("\nWARNING: -mu detected, exting...")
    return
end

m_p                 = zeros(Nsteps,1);
m_pl                = zeros(Nsteps,1);
m_0                 = zeros(Nsteps,1);
m_f                 = zeros(Nsteps,1);
m_pl(end,1)         = m_pl_f;

for i = Nsteps:-1:1
    m_p(i,1)    = m_pl(i,1)*(mu(i,1)-1)*(1-sigma(i,1)) / (1-mu(i,1)*sigma(i,1));
    m_0(i,1)    = m_pl(i,1)*(mu(i,1)*(1-sigma(i,1))) / (1-mu(i,1)*sigma(i,1));
    m_f(i,1)    = m_pl(i,1)*(1-sigma(i,1)) / (1-mu(i,1)*sigma(i,1));

    if i> 1
        m_pl(i-1,1)   = sum(m_0);
    end
end


staging_dv_km    = dv_design(1)*1.0E-3; %km/s
liftoff_mass_T   = m_0(1)*1.0E-3; % T
end
```

```
%{
This simple function checks size of an input

Author: Yevhenii (Jack) Kovryzhenko
yzk0058@auburn.edu
2/7/2021
%}

function CheckSize(Matrix,SizeCheck)
[m,n] = size(Matrix);
if SizeCheck(1) ~= m
    fprintf("\n Error: Input dimentions do not match, check number of columns\n");
    fprintf(" Must be %i, but is %i \n",SizeCheck(1),m);
    %res_bul = false;
    return;
elseif SizeCheck(2) ~= n
    fprintf("\n Error: Input dimentions do not match, check number of rows\n");
    fprintf(" Must be %i, but is %i \n",SizeCheck(2),n);
    %res_bul = false;
    return;
else
    %res_bul = true;
    return;
end
end
```

```
%{
This function has various propulstion
options for GLOM optimization.

Author: Yevhenii (Jack) Kovryzhenko
yzk0058@auburn.edu
4/13/2021
%}
function Isp = get_Isp(Nsteps,Nopts,Input_str,Type_str)
CheckSize(Nsteps,[1,1]);
CheckSize(Input_str,[Nsteps,Nopts]);
CheckSize(Type_str,[Nsteps,Nopts]);

Isp = zeros(Nsteps,Nopts);
for i = 1:Nopts
    for ii = 1:Nsteps
        % Liquid Methane:
        if Input_str(ii,i) == "Raptor"
            Isp_vac = 380.0; %(s)
            Isp_sl  = 330.0; %(s)
            Isp_eff = calc_Isp_eff(Isp_vac,Isp_sl);

            if Type_str(ii,i) == "vac"
                Isp(ii,i) = Isp_vac;
            elseif Type_str(ii,i) == "sl"
                Isp(ii,i) = Isp_sl;
            elseif Type_str(ii,i) == "eff"
                Isp(ii,i) = Isp_eff;
            else
                fprintf("\nERROR: Unkown type, splease select between vac, sl and eff")
            end

        %Liquid Oxygen-Hydrogen:
        elseif Input_str(ii,i) == "RL10"
            Isp_vac = 465.5; %(s)
            Isp_sl  = 450.1; %(s)
            Isp_eff = calc_Isp_eff(Isp_vac,Isp_sl);

            if Type_str(ii,i) == "vac"
                Isp(ii,i) = Isp_vac;
            elseif Type_str(ii,i) == "sl"
                Isp(ii,i) = Isp_sl;
            elseif Type_str(ii,i) == "eff"
                Isp(ii,i) = Isp_eff;
            else
                fprintf("\nERROR: Unkown type, splease select between vac, sl and eff")
            end
        elseif Input_str(ii,i) == "Vinci"
            Isp_vac = 465.0; %(s)
            %Isp_sl  = 465.1; %(s)
            %Isp_eff = calc_Isp_eff(Isp_vac,Isp_sl);

            if Type_str(ii,i) == "vac"
                Isp(ii,i) = Isp_vac;
```

```
        %elseif Type_str(ii,i) == "sl"
        %    Isp(ii,i) = Isp_sl;
        %elseif Type_str(ii,i) == "eff"
        %    Isp(ii,i) = Isp_eff;
        else
            fprintf("\nERROR: Unkown type, splease select vac")
        end
    elseif Input_str(ii,i) == "RS25"
        Isp_vac = 452.3; %(s)
        Isp_sl  = 366.0; %(s)
        Isp_eff = calc_Isp_eff(Isp_vac,Isp_sl);

        if Type_str(ii,i) == "vac"
            Isp(ii,i) = Isp_vac;
        elseif Type_str(ii,i) == "sl"
            Isp(ii,i) = Isp_sl;
        elseif Type_str(ii,i) == "eff"
            Isp(ii,i) = Isp_eff;
        else
            fprintf("\nERROR: Unkown type, splease select between vac, sl and eff")
        end

    %LOX/RP-1:
    elseif Input_str(ii,i) == "Merlin"
        Isp_vac = 311.0; %(s)
        Isp_sl  = 282.0; %(s)
        Isp_eff = calc_Isp_eff(Isp_vac,Isp_sl);

        if Type_str(ii,i) == "vac"
            Isp(ii,i) = Isp_vac;
        elseif Type_str(ii,i) == "sl"
            Isp(ii,i) = Isp_sl;
        elseif Type_str(ii,i) == "eff"
            Isp(ii,i) = Isp_eff;
        else
            fprintf("\nERROR: Unkown type, splease select between vac, sl and eff")
        end
    elseif Input_str(ii,i) == "F1"
        Isp_vac = 304.0; %(s)
        Isp_sl  = 263.0; %(s)
        Isp_eff = calc_Isp_eff(Isp_vac,Isp_sl);

        if Type_str(ii,i) == "vac"
            Isp(ii,i) = Isp_vac;
        elseif Type_str(ii,i) == "sl"
            Isp(ii,i) = Isp_sl;
        elseif Type_str(ii,i) == "eff"
            Isp(ii,i) = Isp_eff;
        else
            fprintf("\nERROR: Unkown type, splease select between vac, sl and eff")
        end
    elseif Input_str(ii,i) == "RD180"
        Isp_vac = 338.0; %(s)
        Isp_sl  = 311.0; %(s)
```

```
            Isp_eff = calc_Isp_eff(Isp_vac,Isp_sl);

            if Type_str(ii,i) == "vac"
                Isp(ii,i) = Isp_vac;
            elseif Type_str(ii,i) == "sl"
                Isp(ii,i) = Isp_sl;
            elseif Type_str(ii,i) == "eff"
                Isp(ii,i) = Isp_eff;
            else
                fprintf("\nERROR: Unkown type, splease select between vac, sl and eff")
            end
        else
            fprintf("\nERROR: Unknown Engine Type")
        end

    end
end
end
```

```
%{
This is the main sctipt for calculating the
mass of the vehicle and finding staging speed.

The primary goal is to obtain optimum solution
given structural ratios and Isp values for each
step and match the required total delta v with
losses.

Alpha and Beta are used to allocate how much
total dv will be provided by step i and the
amount of dv loss absorbed by each step.

Author: Yevhenii (Jack) Kovryzhenko
yzk0058@auburn.edu
4/13/2021
%}

clear all; close all; clc;
%Load the file path:
addpath('../OrbitalStuff')
%---------------- Inputs ------------------%
[dv_orbit,dv_loss_aero,dv_loss_gravity,dv_gain_ls] = getdv;
% dv_gain_ls      = 0.35; %(km/s) dv gain due to launch site
% dv_loss_aero    = 1.15; %(km/s) dv gain due to aerodynamic drag
% dv_loss_gravity = 0.0;  %(km/s) dv gain due to gravity
% dv_orbit        = 7.8;  %(km/s) dv requred for orbit (circular orbital speed)

Nsteps  = 2; %number of steps (for now only up to 2)
Nopts   = 4; %number of test configurations (trying different ISPs)
m_pl_f  = 50.0E3; %(kg) payload mass

find_staging_dv_km = [...
    7
    7
    7
    7];

%Provide range of effective ISP for each step:
Input_str = [...
    "Raptor" "RL10"
    "RD180" "RL10"
    "Raptor" "Raptor"
    "Raptor" "Merlin"]';
Type_str = [...
    "eff" "vac"
    "eff" "vac"
    "eff" "vac"
    "eff" "vac"]';

Isp = get_Isp(Nsteps,Nopts,Input_str,Type_str);
range_alpha = 0.0:0.01:1.0;

%Assign strutural ratio for each step:
```

```
sigma = [...
    0.06
    0.08];

staging_dv_km       = zeros(Nopts,length(range_alpha));
liftoff_mass_T      = zeros(Nopts,length(range_alpha));
min_liftoff_mass_T  = zeros(Nopts,1);
min_alpha           = zeros(Nopts,1);
min_staging_dv_km   = zeros(Nopts,1);
min_p               = zeros(Nsteps,Nopts);
min_pl              = zeros(Nsteps,Nopts);
min_0               = zeros(Nsteps,Nopts);
min_f               = zeros(Nsteps,Nopts);
m_p                 = zeros(Nsteps,1,Nopts,length(range_alpha));
m_pl                = zeros(Nsteps,1,Nopts,length(range_alpha));
m_0                 = zeros(Nsteps,1,Nopts,length(range_alpha));
m_f                 = zeros(Nsteps,1,Nopts,length(range_alpha));

spec_liftoff_mass_T  = zeros(Nopts,1);
spec_alpha           = zeros(Nopts,1);
spec_staging_dv_km   = zeros(Nopts,1);
spec_p               = zeros(Nsteps,Nopts);
spec_pl              = zeros(Nsteps,Nopts);
spec_0               = zeros(Nsteps,Nopts);
spec_f               = zeros(Nsteps,Nopts);
for ii = 1:Nopts
    for i = 1:length(range_alpha)
            %Allocate % of dv provided by each step:
        alpha = [...
            range_alpha(i)
            1-range_alpha(i)];
        %Allocate % of dv loss absorbed by each step:
        beta  = [...
            1.00
            0.00];

        dv_tot.bo       = (dv_orbit - dv_gain_ls); %(m/s) total dv after firing all n steps
        dv_tot.loss     = (dv_loss_aero + dv_loss_gravity); %(m/s) add losses (aero+gravity)
        %Run optimization:
        try %this might fail

            [m_p(:,1,ii,i),m_pl(:,1,ii,i),m_0(:,1,ii,i),m_f(:,1,ii,i),staging_dv_km(ii,i),liftoff
            _mass_T(ii,i)] = calc_mass(Nsteps,m_pl_f,dv_tot,sigma,Isp(:,ii),alpha,beta);
        catch % don't record values if optimization failed
            staging_dv_km(ii,i)     = NaN;
            liftoff_mass_T(ii,i)    = NaN;
            m_p(:,1,ii,i)           = NaN;
            m_pl(:,1,ii,i)          = NaN;
            m_0(:,1,ii,i)           = NaN;
            m_f(:,1,ii,i)           = NaN;
        end
    end
    %Extract minimums:
    min_liftoff_mass_T(ii)  = min(liftoff_mass_T(ii,:));
```

```
    min_alpha(ii)           = range_alpha(liftoff_mass_T(ii,:) == min_liftoff_mass_T(ii));
    min_staging_dv_km(ii)   = staging_dv_km(ii,(liftoff_mass_T(ii,:) == min_liftoff_mass_T(ii)));
    min_p(:,ii)             = m_p(:,1,ii,range_alpha == min_alpha(ii));
    min_pl(:,ii)            = m_pl(:,1,ii,range_alpha == min_alpha(ii));
    min_0(:,ii)             = m_0(:,1,ii,range_alpha == min_alpha(ii));
    min_f(:,ii)             = m_f(:,1,ii,range_alpha == min_alpha(ii));


    %Extract location for specified staging speed:
    [val,spec_ind] = min(abs(staging_dv_km(ii,:) - find_staging_dv_km(ii,1)));

    spec_staging_dv_km(ii)  = staging_dv_km(ii,spec_ind);

    spec_liftoff_mass_T(ii) = min(liftoff_mass_T(ii,spec_ind));
    spec_alpha(ii)          = range_alpha(spec_ind);
    spec_p(:,ii)            = m_p(:,1,ii,spec_ind);
    spec_pl(:,ii)           = m_pl(:,1,ii,spec_ind);
    spec_0(:,ii)            = m_0(:,1,ii,spec_ind);
    spec_f(:,ii)            = m_f(:,1,ii,spec_ind);

    %Construct legend for plotting:
    temp_str = sprintf("%s",Input_str(1,ii)');
    for iii = 1:Nsteps-1
        temp_str = sprintf("%s-%s",temp_str,Input_str(iii+1,ii)');
    end
    tit(ii) = temp_str;
end

figure
hold on
plot(staging_dv_km',liftoff_mass_T')
plot(min_staging_dv_km,min_liftoff_mass_T,'d')
plot(spec_staging_dv_km,spec_liftoff_mass_T,'d')
ylim([0,1.0E4])
xlim([min(min(staging_dv_km)),max(max(staging_dv_km))])
xlabel("Staging speed (km/s)")
ylabel("Liftoff mass (T)")
legend(tit)
title('GLOM Optimization')

dv_total_km = (dv_orbit+dv_loss_aero+dv_loss_gravity-dv_gain_ls)/1000;

for i = 1:Nopts
    fprintf("\n\n\n#==================Minimum==================#\n");
    fprintf("#===========================================#\n");
    fprintf("Minimum masses for %s option:",tit(i))
    for ii = 1:Nsteps
        fprintf("\n Step-%d:",ii)
        fprintf("\n Intial (total) mass: %f (T)",min_0(ii,i)*1.0E-3)
        fprintf("\n Propellant mass:     %f (T)",min_p(ii,i)*1.0E-3)
        fprintf("\n Final (b/o) mass:    %f (T)",min_f(ii,i)*1.0E-3)
        fprintf("\n Payload mass:        %f (T)",min_pl(ii,i)*1.0E-3)
    end
    fprintf("\n\n Staging Speed is %f (km/s)",min_staging_dv_km(i))
```

```
    fprintf("\n Total dv is %f (km/s",dv_total_km)
    fprintf("\n Total minimum liftoff mass is %f (T)",min_liftoff_mass_T(i))

    fprintf("\n\n#==================Custom====================#\n");
    fprintf("Masses for %s option with specified staging speed:",tit(i))
    for ii = 1:Nsteps
        fprintf("\n Step-%d:",ii)
        fprintf("\n Intial (total) mass: %f (T)",spec_0(ii,i)*1.0E-3)
        fprintf("\n Propellant mass:     %f (T)",spec_p(ii,i)*1.0E-3)
        fprintf("\n Final (b/o) mass:    %f (T)",spec_f(ii,i)*1.0E-3)
        fprintf("\n Payload mass:        %f (T)",spec_pl(ii,i)*1.0E-3)
    end
    fprintf("\n\n Staging Speed is %f (km/s)",spec_staging_dv_km(i))
    fprintf("\n Total dv is %f (km/s",dv_total_km)
    fprintf("\n Total minimum liftoff mass is %f (T)",spec_liftoff_mass_T(i))
end
```

```
%{
This function tries to obtain mass ratio mu
by optimizing the GLOM. It is assumed structural
ratio, Isp and delta V values for each stage are given.

Author: Yevhenii (Jack) Kovryzhenko
yzk0058@auburn.edu
4/13/2021
%}
function mu = Optimize_mu(Nsteps,dv,sigma,Isp)
%Check for proper inputs
CheckSize(Nsteps,[1,1]);
CheckSize(Isp,[Nsteps,1]);
CheckSize(sigma,[Nsteps,1]);
CheckSize(dv,[Nsteps,1]);

%Set Constants:
g0 = 9.81; %(m/s^2)
syms L;

% OptimizeMe  = zeros(Nsteps,1);
mu          = zeros(Nsteps,1); %we are solving for this one
L_ans       = zeros(Nsteps,1);
for step_i = 1:1:Nsteps
    %Find structural ratio for the step:
    sigma_i = sigma(step_i);

    %Find the Isp for the step:
    Isp_i   = Isp(step_i);
    %Find the specific exhaust speed for the step:
    c_i     = g0.*Isp_i;

    OptimizeMe(step_i,1) = dv(step_i) - c_i.*log((1+L.*c_i)./(L.*c_i.*sigma_i));

    %Construct function to optimize:
    fun = matlabFunction(OptimizeMe(step_i,1));

    L0 = -0.01; %initial guess

    options = optimoptions('fsolve','Display','off');
    %options = optimoptions('fsolve');
    options.OptimalityTolerance = 1e-9; %increase tolerance
    options.MaxIterations = 5000000; %increase number of iterations
    [L_ans(step_i,1),~,exitflag] = fsolve(fun,L0,options);
    if (exitflag)<=0
        fprintf("\nERROR: No solution found, returning -1");
        mu = -1;
        return
    end
end

for step_i = 1:1:Nsteps
    %Find structural ratio for the step:
    sigma_i = sigma(step_i);
```

```
    %Find the Isp for the step:
    Isp_i   = Isp(step_i);
    %Find the specific exhaust speed for the step:
    c_i     = g0*Isp_i;

    mu(step_i,1) = (1+L_ans(step_i,1)*c_i)/(L_ans(step_i,1)*c_i*sigma_i);
    fprintf("\n Overall mass ratio mu for Step %i is %2.6f with Lagrange multiplier of
    %3.12f",step_i,mu(step_i,1),L_ans(step_i,1));
end
end
```

```
%{
This a quick approximation of the atmosphere
based of altitude

Author: Yevhenii (Jack) Kovryzhenko
yzk0058@auburn.edu
4/13/2021
%}

function [rho,g,P] = atmos_model(h,const)
% const = get_const; %get all the constants

if h < 100000
    P   = const.P0.*exp(-h./const.h0); %Atmospheric pressure
    rho = const.rho0.*exp(-h./const.h0); %Density of atmosphere
else
    P = 0;
    rho = 0;
end
g   = const.g0./((1+h./const.R_E).^2); %Gravity

% [T, a, P, rho] = atmosisa(100000);
end
```

```
%{
This function interpolates
atmospheric and gravity tables
and estimates drag based
on the mach number

Author: Yevhenii (Jack) Kovryzhenko
yzk0058@auburn.edu
4/13/2021
%}

function [D,P,g,M,CD] = get_aero(h,v,Sref,Const)
temp_l  = length(h);
D       = zeros(length(h),1);
CD      = zeros(length(h),1);
c       = zeros(length(h),1);
P       = zeros(length(h),1);
g       = zeros(length(h),1);
M       = zeros(length(h),1);

for i = 1:temp_l
    %Atmospheric Model:
    if h(i) > 200000
        [~,g,P] = atmos_model(h(i),Const);
        c(i)    = interp1(Const.atm.alt,Const.atm.c,h(i),'pchip','extrap');
        M(i)    = v(i)./c(i);
        CD(i)   = 0;
        D(i)    = 0;
    else


    c(i) = interp1(Const.atm.alt,Const.atm.c,h(i),'pchip','extrap');
    P(i) = interp1(Const.atm.alt,Const.atm.P,h(i),'pchip','extrap');
    g(i) = interp1(Const.atm.alt,Const.atm.g,h(i),'pchip','extrap');

    M(i)   = v(i)./c(i);
    Q       = 0.5*P(i).*(M(i,1).^2) .* Const.gamma.*Sref;
    CD(i)   = get_CD_M(Const.M_data,Const.CD_data,M(i,1));
    D(i)  = CD(i) * Q;
    end
end
```

```
%{
This function contains all the
CFD results and is only called
once in the beginning of the
script.

Author: Yevhenii (Jack) Kovryzhenko
yzk0058@auburn.edu
4/13/2021
%}

function [CD,M] = get_aero_model(Const)
addpath('Atmospheric_model/')
%CD = Fd/Q %drag coefficient
%Q  = 0.5*rho*vel^2 = 0.5*P*M.^2 * gamma*A; %dynamic pressure
%M  = vel/a = vel/sqrt(gamma*R*T)
D       = 6.44; % (m) Outer Diameter
alt_m   = 1000; % (km) Altitude
gamma   = 1.4;
P = interp1(Const.atm.alt,Const.atm.P,alt_m,'pchip','extrap');

M   = [...
    0.0
    0.1
    0.3
    0.5
    0.6
    0.7
    0.8
    0.9
    1.0
    1.1
    1.2
    1.3
    1.5
    1.6
    2.2
    2.5
    5.0];

Fd  = [...
    0.0
    6054.38122802688
    52018.5477971049
    141805.31961573
    203953.903159701
    278799.519190528
    369393.593030311
    487976.996070349
    766610.266010653
    1313328.67079808
    1738482.16445425
    1784037.25476708
    2246308.90937081
```

```
    2530130.12236387
    4215116.80390513
    5266505.47232546
    14892291.2437855]; %drag (N)

A  = pi.*(D./2).^2;
Q = 0.5*P.*(M.^2) .* gamma.*A;

CD = Fd./(Q);
CD(1) = 0.3;
end
```

```
%{
This is a quick extrapolation
function for the drag coefficient

Author: Yevhenii (Jack) Kovryzhenko
yzk0058@auburn.edu
4/13/2021
%}

function CD = get_CD_M(M_data,CD_data,M)
CD = interp1(M_data,CD_data,M,'pchip','extrap');
for i = 1:length(CD)
    if CD(i) <= 0 || isnan(CD(i))
        CD(i) = 0;
    end
end
end
```

```
%{
This function contains all the
global constant in one place

Author: Yevhenii (Jack) Kovryzhenko
yzk0058@auburn.edu
4/13/2021
%}

function const = get_const
const.P0      = 101325; %Pa
const.rho0    = 1.225; %kg/m3
const.h0      = 7.64E3; %m
const.g0      = 9.80665; %m/s2
const.R_E     = 6.3781E6; %m

const.v_eq = 465.1; %m/s
const.mu_E = 3.986e14; %m^3/s^2


[Z, ~, ~, T, P, rho, c, g, ~, ~, ~, ~, ~] = atmo(200,1.0,1);
fin_ind = length(c);
const.atm.alt = Z(1:fin_ind)*1000;
% const.atm.Z_L = Z_L(1:fin_ind);
% const.atm.Z_U = Z_U(1:fin_ind);
const.atm.T = T(1:fin_ind);
const.atm.P = P(1:fin_ind);
const.atm.rho = rho(1:fin_ind);
const.atm.c = c(1:fin_ind);
const.atm.g = g(1:fin_ind);
% const.atm.mu = mu(1:fin_ind);
% const.atm.nu = nu(1:fin_ind);
% const.atm.k = k(1:fin_ind);
% const.atm.n = n(1:fin_ind);
% const.atm.n_sum = n_sum(1:fin_ind);

[CD_data,M_data]    = get_aero_model(const);
const.CD_data       = CD_data;
const.M_data        = M_data;
const.gamma = 1.4;
end
```

```
%{
This function estimates the dv gain
based on the launch site location

Author: Yevhenii (Jack) Kovryzhenko
yzk0058@auburn.edu
4/13/2021
%}

function [vf,FPAf,hf,v_launch_pad] = getLEO_orbit(lat,alt)
const = get_const;

%lat = 28.3922; %Cape Canaveral
% lat = 28.5729; %Kennedy Space Center
v_launch_pad = const.v_eq*cosd(lat);

%assume circular orbit
v_orbit = sqrt(const.mu_E/(const.R_E+alt));
FPAf = 0;
hf = alt;

vf = v_orbit - v_launch_pad;
end
```

```
%{
This is the main sctipt for running a simple
ascent simulation for the two-stage launch
vehicle.

The primary goal is to obtain the required
gravity-turn kick angle and location for a
close-to-circular trajectory.

Author: Yevhenii (Jack) Kovryzhenko
yzk0058@auburn.edu
4/13/2021
%}

clear all; close all; clc;

%Launch site parameters:
lat                         = 28.5729; %latitude of the launch site
alt                         = 318E3; %altitude of the launch site
[vf,FPAf,hf,v_launch_pad]   = getLEO_orbit(lat,alt);

%Present important parameters:
grav_angle  = 83.5*pi/180; %(rad)
grav_alt    = 1000; %(m)
tbo_1       = inf; %(s) (if inf. will run till burnout)
tbo_2       = 300; %(s) (will try to achieve circular orbital speed after tbo_2)

coast_1     = 0; % 1-on 0-off
t_coast_1   = 300; %(s)

coast_2     = 0; % 1-on 0-off
t_coast_2   = 60*60*2; %(s)

Glob_Const  = get_const; %get the global constants

%Intial conditions:
v   = 1E-3; %velocity (m/s)
FPA = pi/2; %Flight path angle (rad)
h   = 0; %inital altitude (m)
x   = 0; %inital position (m)
X0  = [v, FPA, h, x,0]';
t0  = 0;

event.t_ign_1 = t0; %for event loging (don't touch)

%Step 1 parameters: (0.0651)
m0      = 1550E3; % Total mass(kg)
mp      = 1307E3; % Propellant mass(kg)
T_sl    = 2.2E6;%1.86E6; % Thrust at sea level (N)
T_vac   = 2.2E6; % Thrust at vacuum (N)
Isp_sl  = 332; %(s)
Ae      = pi * (1.57/2)^2; %Exit Area (m2)
Sref    = pi * (6.44/2)^2; %Reference Area (m2)
mdot    = T_sl/(Glob_Const.g0*Isp_sl); %%Propellant mass flow rate (kg/s)
```

```
max_accel_g = 6; %define maximum accel
N_engines   = 9;
X0(5,1)     = mdot*N_engines;

did_grav_manuever = 0; % 1-on 0-off
tf = tbo_1;

fprintf("\n Solving the first step:")
[t,X] =
solve_step(X0,m0,mp,mdot*N_engines,T_vac*N_engines,Ae*N_engines,Sref,t0,tf,grav_angle,grav_alt,di
d_grav_manuever,max_accel_g,Glob_Const);

event.tbo_1 = t(end);

X0      = X(end,:)'; %at bo
X0(1,1) = X0(1,1)+v_launch_pad;
m       = m0 - X(:,5); %total mass
veh.mp  = mp - X(:,5); %propellant mass

%Step 2 parameters: (0.2647)
m0              = 152E3;%131.488695E3; % Total mass(kg)
mp              = 75E3; % Propellant mass(kg)
% T_sl    = 1.86E6; % Thrust at sea level (N)
T_vac           = 1.02E5; % Thrust at vacuum (N)
Isp_sl          = 332; %(s)
Ae              = pi * (1.3/2)^2; %Exit Area (m2)
Sref            = pi * (6.44/2)^2; %Refference Area (m2)
mdot            = 16; %Propellant mass flow rate (kg/s)
max_accel_g     = 6; %define maximum accel
N_engines       = 7;

did_grav_manuever = 1; % 1-on 0-off

mf_min = m0 - mp;
if coast_1 == 1
    fprintf("\n Solving for the cruise after the first step bo:")
    X0(5,1) = 0;
    did_grav_manuever = 1;

    [t_coast_1,X_coast_1] =
    solve_step(X0,m0,mp,0,0,0,Sref,t0,t0+t_coast_1,grav_angle,grav_alt,did_grav_manuever,max_acce
    l_g,Glob_Const);
    X0      = X_coast_1(end,:)'; %at bo
    X       = [X;X_coast_1];
    t       = [t;t_coast_1+t(end)];
    m       = [m;m0 - X_coast_1(:,5)]; %total mass
    veh.mp  = [veh.mp; mp - X_coast_1(:,5)]; %propellant mass
end
reached_circ_vel    = 0; % 1-on 0-off
first_run           = 1;

fprintf("\n Solving the second step:")
X0(5,1) = mdot*N_engines;
```

```
event.t_ign_2 = t(end);
mp_temp = mp;
while reached_circ_vel ~= 1 && tbo_2 ~= 0
    [t_2,X_2] =
    solve_step(X0,m0,mp,mdot*N_engines,T_vac*N_engines,Ae*N_engines,Sref,t0,t0+tbo_2,grav_angle,g
    rav_alt,did_grav_manuever,max_accel_g,Glob_Const);

    X       = [X;X_2];
    m       = [m;m0 - X_2(:,5)]; %total mass
    veh.mp  = [veh.mp; mp_temp - X_2(:,5)]; %propellant mass

    if first_run == 1
        t = [t;t_2+t(end)];
        first_run = 0;
    else
        t = [t;t_2];
    end
    X0 = X(end,:)'; %at bo
    event.tbo_2 = t(end);

    alt = X0(3,1);
    vf  = X0(1,1);
    mp  = veh.mp(end);

    if vf >= sqrt((Glob_Const.mu_E)/(Glob_Const.R_E+alt))
        reached_circ_vel = 1;
        event.tbo_2 = t(end);
        fprintf('\nSuccess! Reached circular velocity.')
        fprintf('\nTime since ignition: %f\nSecond Burn time is
        %f',event.tbo_2,event.tbo_2-event.t_ign_2)
        fprintf('\nPropellant left: %f (T)\n',mp/1000)
    else
        t0      = t(end);
        tbo_2   = 0.5; %(s) extend the burn for a bit longer
    end

    if mp <= 0 || m(end) <= mf_min
        fprintf('\nBurned all the propellant...')
        break
    end
end
event.tbo_2 = t(end);

if coast_2 == 1
    fprintf("\n Solving for the cruise after the second step bo:")
    did_grav_manuever = 1;
    [t_coast_2,X_coast_2] =
    solve_step(X0,m0,mp,0,0,0,Sref,t(end),t(end)+t_coast_2,grav_angle,grav_alt,did_grav_manuever,
    max_accel_g,Glob_Const);
    X0      = X_coast_2(end,:)'; %at bo
    X       = [X;X_coast_2];
    m       = [m;m0 - X_coast_2(:,5)]; %total mass
    veh.mp  = [veh.mp; mp - X_coast_2(:,5)]; %propellant mass
    t       = [t;t_coast_2+t(end)];
```

```
end

%X  = [v, FPA, h, x, m_burned]';
[D,P,g,M,CD] = get_aero(X(:,3),X(:,1),Sref,Glob_Const); %get_aero(h,v,Sref,Const)

fprintf("\n\n-----------Simulation Completed--------------\n\n")
fprintf("\n Plotting results.......")
plot_res(X,t,D,m,CD,M,Glob_Const,event,veh) %for plottings

%Plot the results:
function plot_res(X,t,D,m,CD,M,Glob_Const,event,veh)
v_km_s              = X(:,1)*1.0E-3;
FPA_deg             = X(:,2)*180/pi;
h_km                = X(:,3)*1.0E-3;
x_km                = X(:,4)*1.0E-3;
m_prop_burned_T     = X(:,5)./1000;

a = (X(2:end,1)-X(1:end-1,1))./(t(2:end)-t(1:end-1));
[max_q,max_q_ind] = max(D);

ind_60km_alt = find(h_km >= 60.0,1,'first');
ind_100km_alt = find(h_km >= 100.0,1,'first');

ind_tbo_1       = find(t >= event.tbo_1,1,'first');
ind_tbo_2       = find(t >= event.tbo_2,1,'first');
ind_ign_2       = find(t > event.t_ign_2,1,'first');

if isempty(ind_60km_alt) || isempty(ind_100km_alt)
    orbit_reached = 0;
    fprintf("\n Orbit has not been reached")
    dv_loss_aero = trapz(t,D./m);
else
    orbit_reached = 1;
    dv_loss_aero = trapz(t(1:ind_100km_alt),D(1:ind_100km_alt)./m(1:ind_100km_alt));
end

leg = sprintf("Flight path");
leg = [leg;sprintf("Max-Q t=%3.2f",t(max_q_ind))];
leg = [leg;sprintf("100km t=%3.2f",t(ind_100km_alt))];
leg = [leg;sprintf("BO-1  t=%3.2f",t(ind_tbo_1))];
leg = [leg;sprintf("BO-2  t=%3.2f",t(ind_tbo_2))];

figure
hold on
title('Launch Vehicle Flight Path')
plot(x_km,h_km)
plot(x_km(max_q_ind),h_km(max_q_ind),'d')
plot(x_km(ind_100km_alt),h_km(ind_100km_alt),'d')
plot(x_km(ind_tbo_1),h_km(ind_tbo_1),'d')
plot(x_km(ind_tbo_2),h_km(ind_tbo_2),'d')
xlabel('Downrange distance (km)')
ylabel('Height above surface (km)')
legend(leg)
if x_km(1) ~= x_km(end)
```

```
    xlim([x_km(1),x_km(end)])
end



figure
m = 4;
n = 1;
i = 0;

i = i+1;
subplot(m,n,i)
hold on
title('Launch Vehicle Flight Parameters')
plot(t,v_km_s)
xlabel('Time (s)')
ylabel('Velocity (km/s)')
xlim([t(1),t(end)])

i = i+1;
subplot(m,n,i)
hold on
plot(t,h_km)
xlabel('Time (s)')
ylabel('Altitude (km)')
xlim([t(1),t(end)])

i = i+1;
subplot(m,n,i)
hold on
plot(t,x_km)
xlabel('Time (s)')
ylabel('x-track (km)')
xlim([t(1),t(end)])

i = i+1;
subplot(m,n,i)
hold on
plot(t,FPA_deg)
xlabel('Time (s)')
ylabel('FPA (deg)')
xlim([t(1),t(end)])

figure
m = 2;
n = 1;
i = 0;

i = i + 1;
subplot(m,n,i)
hold on
title('Launch Vehicle Mass and Acceleration Profile')
if orbit_reached
    plot(t(1:ind_tbo_2),[0; a(1:ind_tbo_2-1)./Glob_Const.g0])
    xlim([t(1),t(ind_tbo_2)])
```

```
else
    plot(t,[0; a./Glob_Const.g0])
    xlim([t(1),t(end)])
end
xlabel('Time (s)')
ylabel('Acceleration (g)')


i = i+1;
subplot(m,n,i)
hold on
if orbit_reached
    plot(t(1:ind_tbo_1),veh.mp(1:ind_tbo_1)./1000)
    plot(t(ind_ign_2:ind_tbo_2),veh.mp(ind_ign_2:ind_tbo_2)./1000)
    xlim([t(1),t(ind_tbo_2)])
    legend('First Step Propellant','Second Step Propellant')
else
    plot(t,m./1000)
    plot(t,veh.mp./1000)
    xlim([t(1),t(end)])
end
ylim([min(veh.mp./1000),max(veh.mp./1000)])
xlabel('Time (s)')
ylabel('Mass (T)')

if orbit_reached
    figure
    hold on
    plot(t(1:ind_60km_alt),D(1:ind_60km_alt)/1000)
    plot(t(max_q_ind),max_q/1000,'d')
    xlabel('Time (s)')
    ylabel('Drag (kN)')
    xlim([t(1),t(ind_60km_alt)])
    tit = sprintf('Ascent Profile: Drag vs Time\n dv loss due to drag is %f (m/s)\nt(max-q) =
    %f (s)',dv_loss_aero,t(max_q_ind));
    title(tit)

    figure
    hold on
    plot(x_km(1:ind_60km_alt),h_km(1:ind_60km_alt))
    plot(x_km(max_q_ind),h_km(max_q_ind),'d')
    ylabel('Altitude (km)')
    xlabel('Downrange distance (km)')
    if x_km(1) ~= x_km(ind_60km_alt)
        xlim([x_km(1),x_km(ind_60km_alt)])
    end

    tit = sprintf('Ascent Profile: Maximum Dynamic Pressure\nh = %f, (km) G-load = %fg, M =
    %f',h_km(max_q_ind),a(max_q_ind-1)/Glob_Const.g0,M(max_q_ind));
    title(tit)
    legend('Trajectory','Max-q')
    ylim([h_km(1),h_km(ind_60km_alt)])
else
    figure
```

```
    m = 2;
    n = 1;
    i = 0;

    i = i+1;
    subplot(m,n,i)
    hold on
    plot(t,D/1000)
    plot(t(max_q_ind),max_q/1000,'d')
    xlabel('Time (s)')
    ylabel('Drag (kN)')
    xlim([t(1),t(end)])
    tit = sprintf('Ascent Profile: Maximum Dynamic Pressure\nh = %f, (km) G-load = %fg, M =
    %f',h_km(max_q_ind),a(max_q_ind-1)/Glob_Const.g0,M(max_q_ind));
    title(tit)

    i = i+1;
    subplot(m,n,i)
    hold on
    plot(t,M)
    plot(t(max_q_ind),M(max_q_ind),'d')
    xlabel('Time (s)')
    ylabel('Mach number')
    xlim([t(1),t(end)])

    figure
    hold on
    plot(x_km,h_km)
    plot(x_km(max_q_ind),h_km(max_q_ind),'d')
    ylabel('Altitude (km)')
    xlabel('Downrange distance (km)')
    if x_km(1) ~= x_km(end)
        xlim([x_km(1),x_km(end)])
    end
    tit = sprintf('Ascent Profile: Maximum Dynamic Pressure\nh(max-q)=%f (km) G-load(max-q) =
    %fg',h_km(max_q_ind),a(max_q_ind-1)/Glob_Const.g0);
    title(tit)
    legend('Trajectory','Max-q')
    ylim([h_km(1),h_km(end)])
end

figure
m = 2;
n = 1;
i = 0;

i = i+1;
subplot(m,n,i)
hold on
if orbit_reached
    plot(M(1:ind_60km_alt),CD(1:ind_60km_alt))
    xlim([M(1),M(ind_60km_alt)])
    ylim([min(M(1:ind_60km_alt)),max(CD(1:ind_60km_alt))])
else
```

```
    plot(M,CD)
    ylim([min(CD),max(CD)])
    xlim([M(1),M(end)])
    if M(1) ~= max(M)
        xlim([M(1),max(M)])
    end
end
title('C_D as a function of mach number')
xlabel('M')
ylabel('C_D')

i = i+1;
subplot(m,n,i)
hold on
if orbit_reached
    plot(t(1:ind_100km_alt),M(1:ind_100km_alt))
    xlim([t(1),t(ind_100km_alt)])
    ylim([min(M(1:ind_100km_alt)),max(M(1:ind_100km_alt))])
else
    plot(t,M)
    xlim([t(1),t])
    ylim([min(M),max(M)])
end
title('Mach number as a function of time')
xlabel('time (s)')
ylabel('M')

figure
hold on
plot(t(2:end),(m_prop_burned_T(2:end)-m_prop_burned_T(1:end-1))./(t(2:end)-t(1:end-1)))
xlabel('Time (s)')
ylabel('Mass flow rate (T/s)')

if orbit_reached
    xlim([t(1),t(ind_tbo_2)])
else
    xlim([t(1),t(end)])
end
end
```

```
%{
This function propagates differential
equations of motion for a single step/stage

Author: Yevhenii (Jack) Kovryzhenko
yzk0058@auburn.edu
4/13/2021
%}
function [t,X] =
solve_step(X0,m0,mp,mdot,T_vac,Ae,Sref,t0,tf,grav_kick_angle_rad,alt_grav,did_grav_manuever,max_a
ccel_g,Glob_Const)
% Glob_Const = get_const;

% mdot        = T_sl/(Glob_Const.g0*Isp); %mass flow rate
t_bo        = mp/mdot + t0;%time of burnout
Step_Const  =
[m0,mdot,T_vac,Ae,Sref,max_accel_g,did_grav_manuever,grav_kick_angle_rad,alt_grav]'; %constant
parameters for each step

dt = 0.5;
if tf < t_bo && tf > t0
    if tf - t0 == dt
        dt = dt/2;
    end
    %run_t = [t0, tf];
    run_t = t0:dt:tf;

elseif tf == t0
    X = X0';
    t = t0;
    fprintf('\nWarning, end time is matching start time, quiting...')
    return
else
    %run_t = [t0, t_bo];
    run_t = t0:dt:t_bo;
end

mf_temp = 0;
mf = m0;
X = X0';
t = t0;
options = odeset('RelTol',1e-12,'AbsTol',1e-8,'Stats','off');
N = length(run_t)-1;
fprintf('\nStarting Sim...')
fprintf('\nProgress:              ');
for i = 1:N
    fprintf('\b\b\b\b\b\b%5.2f%%', i / N * 100);

    t0 = run_t(i);
    t1 = run_t(i+1);

%     [temp_X, t_temp] = forwardEuler (@(t,X)step_dynamics(X,Step_Const,Glob_Const,t), t0, t1,
X0, 100);
    [t_temp,temp_X]   = ode45(@(t,X)step_dynamics(X,Step_Const,Glob_Const,t),[t0,
```

```
    t1],X0,options);
    X    = [X; temp_X(2:end,:)];
    t    = [t; t_temp(2:end,1)];
    X0   = X(end,:)';
    alt = X(end,4);
    if alt >= alt_grav && Step_Const(7) ~= 1
        Step_Const(7) = 1;
    end

    if X0(2,1) < -20 * pi/180 && X0(3,1) > alt_grav && X0(3,1) < 100000
        fprintf("\n Terminating early...");
        return
    end
end
fprintf('\nDone! Exiting...')
end


%This is a short function for propagating the diff. eqns in time at a
%reduced cost and fidelity
function [y, t] = forwardEuler (f, t0, T, y0, N)
%Solve dy/dt = f(t,y) , y(t0 )= y0
h = ( T - t0 )/( N -1); % Calulate and store the step - size
t = linspace( t0 ,T , N ); % A vector to store the time values .
y = zeros (numel(y0) , N); % Initialize the Y matrix .
y(:, 1) = y0(:); % Start y at the initial value .
for i = 1:( N -1)
    y(:, i +1) = y(:, i) + h * f(t(i), y(:, i)); % Update approximation y at t+h
end
y = y';
t = t';
end
```

```
%{
This function contains all the
differential equations needed for
ascent simulation in 3-DOF:
horisontal track, altitude and
flight path angle.

Author: Yevhenii (Jack) Kovryzhenko
yzk0058@auburn.edu
4/13/2021
%}


function dX = step_dynamics(X,Step_Const,Glob_Const,t)
%X = [v, FPA, h, x, mprop_burned]';
v   = X(1,1);
FPA = X(2,1);
h   = X(3,1);
% mprop_burned = X(4,1);

%Step   = [m0,mdot,T_vac,Ae,Sref,max_accel_g]'; %constant parameters for this step
m0                  = Step_Const(1,1); %inital mass
mdot_max            = Step_Const(2,1);
T_vac               = Step_Const(3,1);
Ae                  = Step_Const(4,1);
Sref                = Step_Const(5,1);
max_accel_g         = Step_Const(6,1);
did_grav_manuever   = Step_Const(7,1);
grav_kick_angle_rad = Step_Const(8,1);
alt_grav            = Step_Const(9,1);

% if h < alt_grav
%     h = pi/2;
% end
if did_grav_manuever ~= 1 && h > alt_grav+150
    did_grav_manuever = 1;
end
if h >= alt_grav && did_grav_manuever ~= 1 && abs(FPA - pi/2) < 0.1*180/pi
    FPA = grav_kick_angle_rad;
    did_grav_manuever = 1;
end


mdot = mdot_max; %assume full thrust

%Aerodynamic Model:
[D,Pinf,g,~,~] = get_aero(h,v,Sref,Glob_Const);
% D = get_drag(h,v,Sref);

m = m0 - mdot*t; %mass of the vehicle
% T       = mdot*ve + (Pe - Pinf)*Ae; %Thrust
T = T_vac - Pinf*Ae; %Thrust

a_g   = (T./m - D./m - g.*sin(FPA))/g;
```

```
%need t throttle the engine if acceleration is too great:
if a_g > abs(max_accel_g)
    options = optimoptions('fsolve','Display','off');
    mdot = fsolve(@(mdot)
    solve_mdot(mdot,max_accel_g,Step_Const,X,t,Glob_Const),mdot_max,options);
    if mdot < 0
        mdot = 0;
        fprintf("\n Failed to achive feasible acceleration, mdot needs to be positive");
        %return;
    elseif mdot > mdot_max
        mdot = mdot_max;
        fprintf("\n Failed to achive feasible acceleration, mdot is greater than maximum");
        %return;
    end
    [D,Pinf,g,~,~]  = get_aero(h,v,Sref,Glob_Const);
    m               = m0 - mdot*t; %mass of the vehicle
    T               = T_vac - Pinf*Ae; %Thrust
end

dv_dt   = T./m - D./m - g.*sind(FPA*180/pi); %rate of change of velocity
dFPA_dt = -cosd(FPA*180/pi)*(g./v - v./(Glob_Const.R_E+h)); %rate of change of flight path angle
dh_dt   = v.*sind(FPA*180/pi); %rate of change of height (altitude)
dx_dt   = Glob_Const.R_E./(Glob_Const.R_E+h).*v.*cosd(FPA*180/pi); %rate of change of track

dX      = [dv_dt, dFPA_dt, dh_dt, dx_dt, mdot]';
end


function acc_fun = solve_mdot(mdot,max_accel_g,Step_Const,X,t,Const)
%This function solves for mass flow rate given the required acceleration
%X = [v, FPA, h, x]';
v   = X(1,1);
FPA = X(2,1);
h   = X(3,1);

%Step   = [m0,mdot,T_vac,Ae,Sref]'; %constant parameters for this step
m0      = Step_Const(1,1); %inital mass
% mdot    = Step_Const(2,1);
T_vac   = Step_Const(3,1);
% Pe      = Step_Const(3,1);
Ae      = Step_Const(4,1);
Sref    = Step_Const(5,1);


% [~,g,Pinf] = atmos_model(h);
[D,Pinf,g,~,~] = get_aero(h,v,Sref,Const);

m = m0 - mdot*t; %mass of the vehicle
% T       = mdot*ve + (Pe - Pinf)*Ae; %Thrust
T = T_vac - Pinf*Ae; %Thrust

% D = get_drag(h,v,Sref);
```

```
a_g   = (T./m - D./m - g.*sin(FPA))/g;

acc_fun = a_g - max_accel_g;
end
```

```
function [Z, Z_L, Z_U, T, P, rho, c, g, mu, nu, k, n, n_sum] = atmo(alt,division,units)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Program:    1976 Standard Atmosphere Calculator[0-1000 km]
%   Author:     Brent Lewis(RocketLion@gmail.com)
%               University of Colorado-Boulder
%   History:    Original-1/10/2007
%               Revision-1/12/2007-Corrected for changes in Matlab versions
%               for backward compatability-Many thanks to Rich
%               Rieber(rrieber@gmail.com)
%   Input:      alt:        Final Geometric Altitude[km]
%               division:   Reporting points for output arrays[km]
%                           (.01 km & Divisible by .01 km)
%               units:      1-[Metric]
%                           2-{English}
%   Default:    Values used if no input
%               alt:        1000 km
%               division:   1 km
%               units:      Metric
%   Output:     Each value has a specific region that it is valid in with this model
%               and is only printed out in that region
%               Z:          Total Reporting Altitudes[0<=alt<=1000 km][km]{ft}
%               Z_L:        Lower Atmosphere Reporting Altitudes[0<=alt<=86 km][km]{ft}
%               Z_U:        Upper Atmosphere Reporting Altitudes[86<=alt<=1000 km][km]{ft}
%               T:          Temperature array[0<=alt<=1000 km][K]{R}
%               P:          Pressure array[0<=alt<=1000 km][Pa]{in_Hg}
%               rho:        Density array[0<=alt<=1000 km][kg/m^3]{lb/ft^3}
%               c:          Speed of sound array[0<=alt<=86 km][m/s]{ft/s}
%               g:          Gravity array[0<=alt<=1000 km][m/s^2]{ft/s^2}
%               mu:         Dynamic Viscosity array[0<=alt<=86 km][N*s/m^2]{lb/(ft*s)}
%               nu:         Kinematic Viscosity array[0<=alt<=86 km][m^2/s]{ft^2/s}
%               k:          Coefficient of Thermal Conductivity
%                           array[0<=alt<=86 km][W/(m*K)]{BTU/(ft*s*R)}
%               n:          Number Density of individual gases
%                           (N2 O O2 Ar He H)[86km<=alt<=1000km][1/m^3]{1/ft^3}
%               n_sum:      Number Density of total gases
%                           [86km<=alt<=1000km][1/m^3]{1/ft^3}
%   Acknowledgements:       1976 U.S. Standard Atmosphere
%                           Prof. Adam Norris-Numerical Analysis Class
%                           Steven S. Pietrobon USSA1976 Program
%   Notes:                  Program uses a 5-point Simpson's Rule in 10
%                           meter increments.  Results DO vary by less 1%
%                           compared to tabulated values and is probably
%                           caused by different integration techniques
%   Examples:               atmo() will compute the full atmosphere in 1 km
%                           increments and output in Metric Units
%                           atmo(10) will compute the atmosphere between 0
%                           and 10 km in 1 km increments and output in
%                           Metric Units
%                           atmo(20,.1,2) will compute the atmosphere
%                           between 0 and 20 km in 100 m increments and
%                           output in English Units
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if nargin == 0
    alt = 1000;
```

```
    division = 1;
    units = 1;
elseif nargin == 1
    division = 1;
    units = 1;
elseif nargin == 2
    units = 1;
end

%   Error Reporting
if nargin > 3
    error('Too many inputs')
elseif mod(division,.01) ~= 0
    error('Divisions must be multiples of .01 km')
elseif units ~= 1 && units ~= 2
    error('Units Choice Invalid[1-Metric,2-English]')
elseif alt<0 || alt>1000
    error('Program only valid for 0<altitudes<1000 km')
end

%   Matrix Pre-allocation
if alt <= 86
    Z_L = (0:division:alt)';
    Z_U = [];
    n = [];
else
    Z_L = (0:division:86)';
    Z_U = (86:division:alt)';
    if mod(86,division) ~= 0
        Z_L = [Z_L; 86];
    end
    if mod(alt-86,division) ~= 0
        Z_U = [Z_U; alt];
    end
end
T_L = zeros(size(Z_L));
T_M_L = T_L;
T_U = zeros(size(Z_U));

%   Conversion Factor Used in 80<alt<86 km
Z_M = 80:.5:86;
M_M_0 = [1 .999996 .999989 .999971 .999941 .999909 ...
    .999870 .999829 .999786 .999741 .999694 .999641 .999579];

%   Constants
M_0 = 28.9644;
M_i = [28.0134; 15.9994; 31.9988; 39.948; 4.0026; 1.00797];
beta = 1.458e-6;
gamma = 1.4;
g_0 = 9.80665;
R = 8.31432e3;
r_E = 6.356766e3;
S = 110.4;
N_A = 6.022169e26;
```

```
%   Temperature
for i = 1 : length(Z_L)
    T_L(i,1) = atmo_temp(Z_L(i));
    T_M_L(i,1) = T_L(i,1);
    if Z_L(i) > 80 && Z_L(i) < 86
        T_L(i,1) = T_L(i)*interp1(Z_M,M_M_0,Z_L(i));
    end
end
for i = 1 : length(Z_U)
    T_U(i,1) = atmo_temp(Z_U(i));
end

%   Number Density
if alt > 86
    n = atmo_compo(alt,division);
    n_sum = sum(n,2);
else
    n = [];
    n_sum = [];
end

%   Pressure
P_L = atmo_p(Z_L);
P_U = atmo_p(Z_U,T_U,n_sum);

%   Density
rho_L = M_0*P_L./(R*T_M_L);
if ~isempty(P_U)
    rho_U = n*M_i/N_A;
else
    rho_U = [];
end

%   Speed of Sound
c = sqrt(gamma*R*T_M_L/M_0);
%   Dynamic Viscosity
mu = beta*T_L.^1.5./(T_L+S);
%   Kinematic Viscosity
nu = mu./rho_L;
%   Thermal Conductivity Coefficient
k = 2.64638e-3*T_L.^1.5./(T_L+245*10.^(-12./T_L));

%   Combine Models
T = [T_L(1:end-1*double(~isempty(T_U)));T_U];
P = [P_L(1:end-1*double(~isempty(T_U)));P_U];
rho = [rho_L(1:end-1*double(~isempty(T_U)));rho_U];
Z = [Z_L(1:end-1*double(~isempty(T_U)));Z_U];

%   Gravity
g = g_0*(r_E./(r_E+Z)).^2;

if units == 2
    unit_c = [3.048e-1 3.048e-1 3.048e-1 5/9 0.0001450377 1.6018463e1...
```

```
        3.048e-1 3.048e-1 1.488163944 9.290304e-2 6.226477504e-3...
        3.531466672e2 3.531466672e2];
    Z = Z/unit_c(1);
    Z_L = Z_L/unit_c(2);
    Z_U = Z_U/unit_c(3);
    T = T/unit_c(4);
    P = P/unit_c(5);
    rho = rho/unit_c(6);
    c = c/unit_c(7);
    g = g/unit_c(8);
    mu = mu/unit_c(9);
    nu = nu/unit_c(10);
    k = n/unit_c(11);
    n_sum = n_sum/unit_c(12);
end
```

```
function n_i_array = atmo_compo(alt,division)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Program:    High Altitude Atmospheric Composition Calculation
%   Author:     Brent Lewis(RocketLion@gmail.com)
%               University of Colorado-Boulder
%   History:    Original-1/10/2007
%               Revision-1/12/2007-Corrected for changes in Matlab versions
%               for backward compatability-Many thanks to Rich
%               Rieber(rrieber@gmail.com)
%   Input:      alt:        Geometric Altitude of desired altitude[scalar][km]
%               division:   Desired output altitudes
%   Output:     n_i_array:  Array of compositions of [N2 O O2 Ar He H] at
%                           desired reporting altitudes using equations
%                           from 1976 Standard Atmosphere
%   Note:       Only Valid Between 86 km and 1000 km
%               Division must be a multiple of 10 m;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Z_i = [86 91 95 97 100 110 115 120 150 500 1000];
step = .01;

if alt < Z_i(1) || alt>Z_i(length(Z_i))
    n_i_array = [];
    return;
end

%   Gas coefficients
alpha_i = [0; 0; 0; 0; -.4; -.25];
a_i = [0; 6.986e20; 4.863e20; 4.487e20; 1.7e21; 3.305e21];
b_i = [0; .75; .75; .87; .691; .5];
Q_i = [0; -5.809644e-4; 1.366212e-4; 9.434079e-5; -2.457369e-4];
q_i = [0; -3.416248e-3; 0; 0; 0];
U_i = [0; 56.90311; 86; 86; 86];
u_i = [0; 97; 0; 0; 0];
W_i = [0; 2.70624e-5; 8.333333e-5; 8.333333e-5; 6.666667e-4];
w_i = [0; 5.008765e-4; 0; 0; 0];

%   Gas Data
R = 8.31432e3;
phi = 7.2e11;
T_7 = 186.8673;
T_11 = 999.2356;
%   Molecular Weight & Number Density based on values at 86 km & 500 km for
%   Hydrogen
n_i_86 = [1.129794e20; 8.6e16; 3.030898e19; 1.3514e18; 7.5817e14; 8e10];
n_i_alt = n_i_86;
sum_n = [ones(3,1)*n_i_86(1);ones(2,1)*sum(n_i_86(1:3));sum(n_i_86(1:5))];
M_i = [28.0134; 15.9994; 31.9988; 39.948; 4.0026; 1.00797];
M_0 = 28.9644;

n_int = zeros(size(n_i_86));
j = 1;
n_i_array = zeros(floor((alt-86)/division)+1,6);
for i = 1 : length(Z_i)-1
```

```
    if alt > Z_i(i)
        Z_start = Z_i(i);
        if alt > Z_i(i+1)
            Z_end = Z_i(i+1);
        else
            Z_end = alt;
        end
        for Z_0 = Z_start:step:Z_end-step
            Z_1 = Z_0+step;
            if Z_1 <= Z_i(5)
                M = ones(size(M_i))*M_0;
            else
                M = [(n_i_alt(1)*M_i(1))./sum_n(1:3);...
                    sum((n_i_alt(1:3).*M_i(1:3)))./sum_n(4:5);...
                    sum((n_i_alt(1:5).*M_i(1:5)))./sum_n(6)];
            end
            sum_n = [ones(3,1)*n_i_alt(1);ones(2,1)*sum(n_i_alt(1:3));sum(n_i_alt(1:5))];
            n_int = f_n(a_i,alpha_i,b_i,M,M_i,n_int,phi,...
                Q_i,q_i,R,sum_n,U_i,u_i,W_i,w_i,Z_i,Z_0,Z_1);
            n_i_alt(1:5) = n_i_86(1:5)*T_7/atmo_temp(Z_1).*exp(-n_int(1:5));
            if Z_1 < Z_i(9)
                n_i_alt(6) = 0;
            else
                tau = int_tau(alt);
                n_i_alt(6) = (T_11/atmo_temp(Z_1))^(1+alpha_i(6))*...
                    (n_i_86(6)*exp(-tau)-n_int(6));
            end
            if mod(Z_0,division) == 0
                n_i_array(j,:) = n_i_alt';
                j = j+1;
            end

        end
    end
end
n_i_end(1:5) = n_i_86(1:5)*T_7/atmo_temp(alt).*exp(-n_int(1:5));
if alt < Z_i(9)
    n_i_end(6) = 0;
else
    tau = int_tau(alt);
    n_i_end(6) = (T_11/atmo_temp(Z_1))^(1+alpha_i(6))*...
        (n_i_86(6)*exp(-tau)-n_int(6));
end
n_i_array(j,:) = n_i_end;
```

```
function P = atmo_p(alt, T, sum_n)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Program:    Atmospheric Pressure Calculation
%   Author:     Brent Lewis(RocketLion@gmail.com)
%               University of Colorado-Boulder
%   History:    Original-1/10/2007
%               Revision-1/12/2007-Corrected for changes in Matlab versions
%               for backward compatability-Many thanks to Rich
%               Rieber(rrieber@gmail.com)
%   Input:      alt:    Geometric altitude vector of desired pressure data[km]
%               T:      Temperature vector at given altitude points
%                       Required only for altitudes greater than 86 km[K]
%               sum_n:  Total number density of atmospheric gases[1/m^3]
%   Output:     P:      Pressure vector[Pa]
%   Note:       Must compute altitudes below 86 km and above 86 km on two
%               different runs to allow line up of altitudes and
%               temperatures
%   Examples:   atmo_p(0) = 101325 Pa
%               atmo_p(0:10) = Pressures between 0-10 km at 1 km increments
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if nargin == 1
    T = [];
    sum_n = [];
end

%   Constants
N_A = 6.022169e26;
g_0 = 9.80665;
M_0 = 28.9644;
R = 8.31432e3;
r_E = 6.356766e3;
%   Geopotential/Geometric Altitudes used for Geometric Altitudes < 86 km
H = [0 11 20 32 47 51 71 84.852];
Z = r_E*H./(r_E-H);
Z(8) = 86;

%   Defined temperatures/lapse rates/pressures/density at each layer
T_M_B = [288.15 216.65 216.65 228.65 270.65 270.65 214.65];
L = [-6.5 0 1 2.8 0 -2.8 -2]/1e3;
P_ref = [1.01325e5 2.2632e4 5.4748e3 8.6801e2 1.1090e2 6.6938e1 3.9564];

%   Preallocation of Memory
P = zeros(size(alt));

for i = 1 : length(alt)
    Z_i = alt(i);
    if isempty(sum_n)
        index = find(Z>=Z_i)-1+double(Z_i==0);
        index = index(1);
        Z_H = r_E*Z_i/(r_E+Z_i);
        if L(index) == 0
            P(i) = P_ref(index)*exp(-g_0*M_0*(Z_H-H(index))*1e3/(R*T_M_B(index)));
        else
            P(i) = P_ref(index)*(T_M_B(index)/...
```

```
                (T_M_B(index)+L(index)*(Z_H-H(index))*1e3))^...
                (g_0*M_0/(R*L(index)));
        end
    else
        if i > length(alt) || i > length(P) || i > length(T) || 1 < length(R) || 1 < length(N_A)
            fprintf("\n Exeeding Array size")
        end
        P(i) = sum_n(i)*R*T(i)/N_A;
    end
end
```

```
function Temp = atmo_temp(alt)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Program:    Atmospheric Temperature Calculation
%   Author:     Brent Lewis(RocketLion@gmail.com)
%               University of Colorado-Boulder
%   History:    Original-1/09/2007
%   Input:      alt:    Geometric Altitude of desired altitude[scalar][km]
%   Output:     Temp:   Temperature at desired altitude using values from
%                       1976 Standard Atmosphere[K]
%   Note:       Only Valid Between 0 km and 1000 km
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%   Constants
r_E = 6.356766e3;
epsilon = 1e5*eps;

%   Defined temperatures at each layer
T = [288.15 216.65 216.65 228.65 270.65 270.65 ...
    214.65 186.95 186.8673 240 360 1000];
L = [-6.5 0 1 2.8 0 -2.8 -2 0 0 12 0];

%   Geopotential/Geometric Altitudes used for Geometric Altitudes < 86 km
H = [0 11 20 32 47 51 71];
Z = r_E*H./(r_E-H);
%   Geometric Altitudes used for Altitudes >86 km
Z(8:12) = [86 91 110 120 1000];

if alt < Z(1) || alt > (Z(12)+epsilon)
    error('Altitude must be 0-1000 km')
end

%   Temperature Calculation with Molecular Temperature below 86 km and
%   Kinetic Temperature above
if alt >= Z(1) && alt <= Z(8)
    Temp = interp1(Z,T,alt);
elseif alt > Z(8) && alt <= Z(9)
    Temp = T(9);
elseif alt > Z(9) && alt <= Z(10)
    a = 19.9429;
    A = -76.3232;
    T_c = 263.1905;
    Temp = T_c+A*sqrt(1-((alt-Z(9))/a)^2);
elseif  alt > Z(10) && alt <= Z(11)
    Temp = interp1(Z,T,alt);
elseif alt > Z(11)
    lambda = L(10)/(T(12)-T(11));
    xi = (alt-Z(11))*(r_E+Z(11))/(r_E+alt);
    Temp = T(12)-(T(12)-T(11))*exp(-lambda*xi);
end
```

```
function n_int = f_n(a_i,alpha_i,b_i,M,M_i,n_int,phi,...
    Q_i,q_i,R,sum_n,U_i,u_i,W_i,w_i,Z_i,Z_0,Z_1)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Program:    Gas Integral Program
%   Author:     Brent Lewis(RocketLion@gmail.com)
%               University of Colorado-Boulder
%   History:    Original-1/10/2007
%   Input:      As defined in 1976 Standard Atmosphere
%   Output:     n_int:  Integral values computed using 5-point Simpsons
%                       Rule
%   Note:       Created for running in Atmospheric Program
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%   Constants
g_0 = 9.80665;
r_E = 6.356766e3;
T_c = 263.1905;
A_8 = -76.3232;
a_8 = 19.9429;
L_K_9 = 12;
T_7 = 186.8673;
T_9 = 240;
T_10 = 360;
T_11 = 999.2356;
T_inf = 1000;

%   Molecular Diffusion Coeffiecients
K_7 = 1.2e2;

alt_j = linspace(Z_0,Z_1,5);
n_i = zeros(6,length(alt_j));
for j = 1 : length(alt_j)
    Z = alt_j(j);
    %     Temperature Values
    if Z < Z_i(2)
        T = T_7;
        dT_dZ = 0;
    elseif Z < Z_i(6)
        T = T_c+A_8*sqrt(1-((Z-Z_i(2))/a_8)^2);
        dT_dZ = -A_8/a_8^2*(Z-Z_i(2))*(1-((Z-Z_i(2))/a_8)^2)^-.5;
    elseif Z < Z_i(8)
        T = T_9+L_K_9*(Z-Z_i(6));
        dT_dZ = L_K_9;
    elseif Z >= Z_i(8)
        lambda = L_K_9/(T_inf-T_10);
        xi = (Z-Z_i(8))*(r_E+Z_i(8))/(r_E+Z);
        T = T_inf-(T_inf-T_10)*exp(-lambda*xi);
        dT_dZ = lambda*(T_inf-T_10)*((r_E+Z_i(8))/(r_E+Z))^2*exp(-lambda*xi);
    end
    %     K Values
    if Z < Z_i(3)
        K = K_7;
    elseif Z < Z_i(7)
        K = K_7*exp(1-400/(400-(Z-95)^2));
```

```
    elseif Z >= Z_i(7)
        K = 0;
    end
    %     Gravity
    g = g_0*(r_E/(r_E+Z))^2;

    %     N
    if Z <= Z_i(5)
        M_N2 = M(1);
    else
        M_N2 = M_i(1);
    end
    n_i(1,j) = M_N2*g/(T*R)*1e3;
    %     O O2 Ar He
    D = a_i(2:5).*exp(b_i(2:5).*log(T/273.15))./sum_n(2:5);
    if K ~= 0
        f_Z = (g/(R*T)*D./(D+K).*(M_i(2:5)+M(2:5)*K./D+...
            alpha_i(2:5)*R/g*dT_dZ/1e3))*1e3;
    else
        f_Z = (g/(R*T)*(M_i(2:5)+alpha_i(2:5)*R/g*dT_dZ/1e3))*1e3;
    end
    if Z <= Z_i(4)
        vdk = Q_i(2:5).*([Z;Z;Z;Z]-U_i(2:5)).^2.*exp(-W_i(2:5).*...
            ([Z;Z;Z;Z]-U_i(2:5)).^3)+q_i(2:5).*(u_i(2:5)-[Z;Z;Z;Z]).^2.*...
            exp(-w_i(2:5).*(u_i(2:5)-[Z;Z;Z;Z]).^3);
    else
        vdk = Q_i(2:5).*([Z;Z;Z;Z]-U_i(2:5)).^2.*exp(-W_i(2:5).*...
            ([Z;Z;Z;Z]-U_i(2:5)).^3);
    end
    n_i(2:5,j) = f_Z+vdk;
    %     H
    if Z_1 < 150 || Z_1 > 500
        n_i(6,j) = 0;
    else
        D_H = a_i(6)*exp(b_i(6)*log(T/273.15))/sum_n(6);
        n_i(6,j) = phi/D_H*(T/T_11)^(1+alpha_i(6));
    end

end
h = alt_j(2)-alt_j(1);
n_int = n_int+(2*h/45*(7*n_i(:,1)+32*n_i(:,2)+12*n_i(:,3)+32*n_i(:,4)+7*n_i(:,5)));
```

```
function TAU = int_tau(Z)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Program:    Tau Integral Computation for Hydrogen Composition Program
%               in Atmospheric Model
%   Author:     Brent Lewis(RocketLion@gmail.com)
%               University of Colorado-Boulder
%   History:    Original-1/10/2007
%   Input:      Z:      Altitude value
%   Output:     TAU:    Integral Value
%   Note:       This program computes the value of Tau directly with the
%               integral done by hand and only the second integration limit
%               needing to be inputed
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%   Constants
L_K_9 = 12;
T_10 = 360;
T_inf = 1000;
Z_10 = 120;
g_0 = 9.80665;
r_E = 6.356766e3;
R = 8.31432e3;
lambda = L_K_9/(T_inf-T_10);
M_H = 1.00797;

%   Value of Integration limit computed previously
tau_11 = 8.329503912749350e-004;

tau_Z = M_H*g_0*r_E^2/R*...
    log((exp(lambda*(Z-Z_10)*(r_E+Z_10)/(r_E+Z))-1)*T_inf+T_10)/...
    (lambda*T_inf*(r_E+Z_10)^2);

TAU = tau_Z-tau_11;
```