Aerospace Design Project:, Final Report

Joshua Zeitlin 2789506Z, Jaden Ferrao 2750098F, Yuki Suter 2758934S,
Arka Chatterjee 2757723C, Jack Chambré 2724756C

October 19, 2024

Contents

1 Introduction

1.1 Team Overview

The team consists of 5 members, 4 of which are majoring in Aeronautical Engineering and 1 in Aerospace Systems. Thus, the team is geared towards more revolutionary ideas and newer technologies in the Aerospace Sector. The team's different ideas and approaches have yielded a well-rounded design with each team member contributing in their own way to the project. Each team member has their own speciality and has pursued it at great length in this project. This design aims to improve multiple areas, from structures and propulsion to avionics and economics.

1.2 Mission Statement

The team aims to design a new launcher dubbed "Eos" after the Greek goddess of dawn. It is a fitting name as this new launcher aims to usher in a new generation of spacecraft to help us deliver various payloads into orbit with minimal cost and improved efficiency.

The concept hopes to improve on the Ariane 5 family of launchers with the goal of reducing costs and addressing concerns about sustainability and environmental impact. Sustainability is one of the key issues facing the aerospace sector at the moment. With mounting pressure from governments and the general public to end wasteful practices, now is the time to start focusing on re-usability.

1.3 Background

Spaceflight is a rapidly expanding industry in the UK and the economic benefits of reducing entry costs are clear. This is at an important time as reusable spacecraft are becoming common place and lower costs are opening up the industry to other sectors such as tourism. This is leading to the commercialisation of space rather than it being resigned to state sponsored endeavours.

Due to the cost and high chance of failure, achieving orbital spaceflight is difficult and there are many ways we can see it improving. There are plenty of companies trying to innovate launches to orbit. However, with more launches than ever before in 2023 (224 to be exact) there is still a 5. 4% failure rate. This is not common in other industries, which shows there is a need and a demand for new technologies.

The way we plan to improve upon current orbital vehicle designs is by using more efficient engines and fly-back recovery with wings. The benefit from using wings is a lower fuel consumption when compared to a vertical landing. Gliding is proven to be safe, reliable, and efficient.

1.4 Mission Goals

To design a,

- Primary Launcher capable of delivering a 25,000 kg payload to LEO and 12,000 kg payload to GTO.
- Secondary Launcher capable of delivering a 12,500 kg payload to LEO and a 6,000 kg payload to GTO

- Primary and Secondary Launcher that is partially re-usable, easy to recover and maintain with minimal costs and repairs.
- Primary and Secondary launcher that is more cost-effective and more sustainable by building upon existing technologies.

2 Concept development

2.1 Concept Overview

Our launcher design consists of a primary and a secondary configuration. The primary consists of a core stage and two boosters. The secondary consists of the core stage with added raptor engines and no boosters.
The boosters on the primary configuration will be recovered by way of three parachutes each. The key feature that separates this launcher design is the core-stage recovery system. Once the core stage has disengaged and decoupled, a set of wings and a set of horizontal stabilisers will fold out and lock into place. The main wing is located close to the top of the core stage, while the horizontal stabilisers are located near the engines. This set of wings will give the core stage the ability to glide through the upper atmosphere. This will slow down its reentry velocity and minimise damage upon landing, increasing its re usability.

2.2 Inspiration

The inspiration can be traced to number of designs including the F-4 Phantom, space shuttle, and Baikal fly-back booster. We wanted to come up with something that combined an already very well studied field with the cutting edge aspect of launcher design. The idea of having wings on a launcher seemed like a logical one and more cost effective than a completely new technology. Furthermore, landing an object such as a rocket launcher vertically is a lot harder than landing it horizontally and requires a lot more fuel.

2.3 Comparison to other design proposals

There are a few design proposals that have shared featured and are worth taking a look at. Namely: The Russian 'Baikal Booster', the German 'Liquid Fly-back Booster', and Airbus' 'ADELINE'.

2.3.1 Baikal Booster

The Baikal booster, shown in Fig.1, is a Fly-back Booster developed by the Krunichev space centre. It consists of a folding wing, fly-back turbojet engine, and a control system. The booster would come in to land similar to a standard aircraft [1]. The booster has a fixed tail and horizontal stabilisers so would be easier to control than a space plane type vehicle. The booster has a kerosene tank in its rear which is used during takeoff and also by the turbojet engine during landing. When in flight the booster would have a high-wing plane aerodynamic configuration [2].
This design shares a lot of similarities with ours however there are some key differences. The Baikal uses a single wing configuration but we have chosen to go for a canard configuration as we believe this will achieve greater stability and afford more control over the core stage re-entry. Another key difference is the use of a single wing on the Baikal. We have decided to go for two fold out wings as we feel this system would be cheaper to develop and would be lighter than one large rigid wing.
Overall this design is what we are going for in principle and serves as a good baseline to work off and learn from.

Figure 1: Baikal Fly-back Booster

2.3.2 Liquid Fly-back Booster

The DLR Liquid Fly-back Booster, shown in Fig.2, was a proposed booster for the Ariane 5 core stage. It had a unique design that was part space plane, part conventional booster, and part conventional aircraft. The engines for fly-back along with landing gear and control systems were stored in the nose, which helped to redistribute mass after the propellant was used up. The design had a large delta wing near the base of the booster, a small front wing, and a 'v' tail for greater maneuverability [3].
This design shares the canard configuration with our concept but has a fixed wing design. The key benefit to this is that there is more room for propellant and no complicated fold-out mechanisms. On the other hand, the risk of damage to the wings is much higher during take-off. This was one of the key reasons behind our decision to go for a fold-out wing design. This way the wings are protected within the body of the core stage and are not exposed to the harsh conditions experienced during takeoff.
This design has some interesting features and its stabiliser configuration is certainly something that we could look to emulate in our design.

Figure 2: Liquid Fly-back booster

2.3.3 ADELINE

The Adeline launcher, shown in Fig.3, is a system that uses a fixed wing with fold out winglets and deploy-able propellers. The design initially glides toward a landing strip before the winglets deploy, allowing it to steer into position. Just before landing, the propellers deploy allowing full control over the landing procedure [4]. The Adeline system differs from our design in its use of propellers instead of engines. The reasoning for this according to airbus is that propellers are far lighter and more fuel efficient, which makes for greater savings over time. The designs wing, like ours, has a high aspect ratio giving the launcher a much more controlled landing.
Something that could be taken from this design is its use of wingtips which could further increase savings.

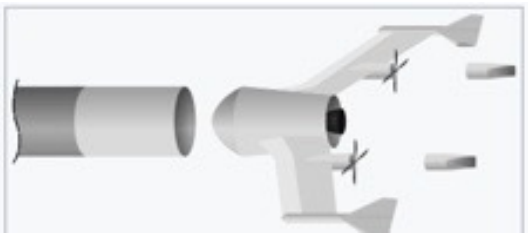

Figure 3: Advanced Expendable Launcher with Innovative engine Economy

3 Launcher performance analysis (approx. 2-5 pages)

3.1 Launcher Parameters

Section 3.1.1 and 3.1.2 show the different parameters that were used to write the code used to analyse the launcher performance. These values were obtained from various literature values (referenced in the caption) or calculated previously.

3.1.1 Primary Launcher

	Core Stage 1x Raptor 2 Engine	Liquid Rocket Booster 3x Raptor 2 Engines	Upper Stage 1x HM7B Engine
Launch Mass [kg]	242,941	185,000	19,440
Inert Mass [kg]	26,726	15,000	4,540
Oxidant Fuel Split []	3.8	3.8	4.9
Propellant Mass [kg]	216,215	170,000	14,900
Oxidant Mass [kg]	171170.6 LO_x	136543.5 LO_x	12374.6 LO_x
Fuel Mass [kg]	45044.4 CH_4	35932.5 CH_4	2525.4 LH_2
Mass Flow Rate [kg/s]	650	1950	14.8
Mf/Ms []	8.099	11.333	3.282
Isp [s]	327 (sea-level), 363 (vac)	327 (sea-level), 363 (vac)	446 (vac)
Thrust [kN]	2,260 (sea-level), 2,530 (vac)	6,780 (sea-level), 7,590 (vac)	62.7
Burn Time [s]	332.6	88.45	945

Table 1: The different parameters inputted into the code to give performance results for the primary launcher.

3.1.2 Secondary Launcher

	Core Stage 3x Raptor 2 Engines	Upper Stage 1x HM7B Engine
Launch Mass [kg]	242,141	185,000
Inert Mass [kg]	26,926	4540
Oxidant Fuel Split []	3.8	4.9
Propellant Mass [kg]	216,215	14,900
Oxidant Mass [kg]	171170.6 LO_x	12374.6 LO_x
Fuel Mass [kg]	45044.4 CH_4	2525.4 LH_2
Mass Flow Rate [kg/s]	1950	14.8
Mf/Ms []	7.224	3.282
Isp [s]	327 (sea-level), 363 (vac)	446 (vac)
Thrust [kN]	6780 (sea-level), 7590 (vac)	62.7 (vac)
Burn Time [s]	110	945

Table 2: The different parameters inputted into the code to give performance results for the secondary launcher.

3.2 Methodology

The code used to calculate the launcher performance can be found in section A. The code has been heavily modified and adapted from the JAXA H-II code provided. By adapting the code to our needs we have been able to verify the accuracy of the code and therefore our results. The code works by calculating and logging the different parameters of the launcher at different times. It does this by sampling every X seconds, were X can be set by the user upon starting the script.

The code written can be run to calculate the different graphs for both the primary launcher configuration and the secondary launcher configuration. For the primary launcher configuration it is also possible to alter the throttle percentage of the boosters, for reasons which can be seen and discussed in the results section. The script calculates the position, speeds etc from the use of simple dynamics equations. This does mean that air resistance is not accounted for, but this would vary greatly between sea level and stage separation so would require a decently large amount of additional code.

3.3 Results

3.3.1 Primary Launcher - LEO

The following graphs show the trajectory simulation for the primary launcher, with a payload mass of 25,000kg, and throttle percentage of 100%. This aligns with the goal for getting a 25,000kg payload to LEO. The graphs show that a LEO mission with this payload and configuration should be possible.

Figure 4 shows how the boosters designed for this configuration may be slightly too overpowered for the setup. This is why the code was designed such that the booster throttle value for the primary configuration could be adjusted. Upon decreasing the booster throttle

percentage, the speed graph looked to show more of an upward trend. The code was designed such that the burn time increases inversely proportional to the throttle percentage, allowing for all the fuel in the booster tanks to be utilised.

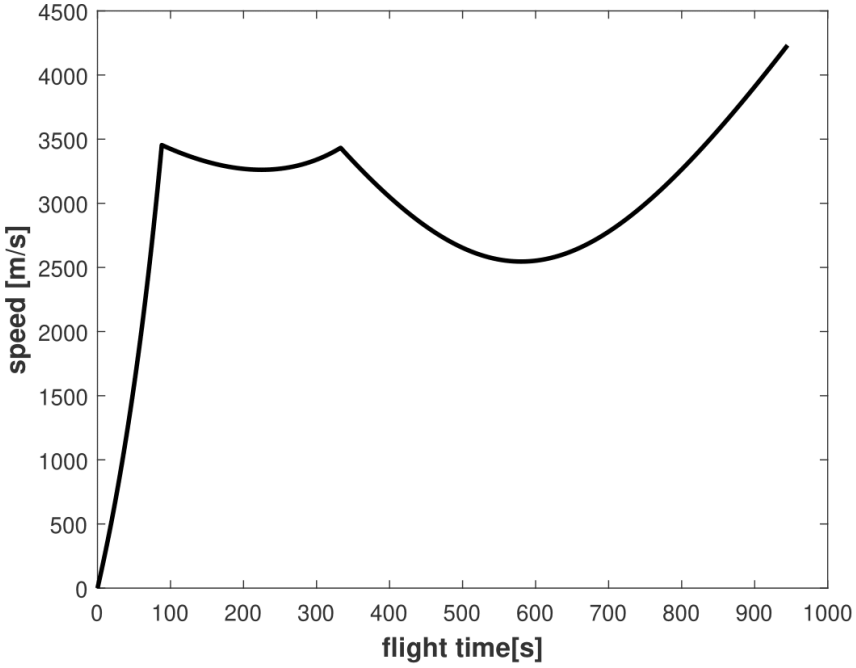

Figure 4: Graph showing how speed of the primary launcher varies with time.

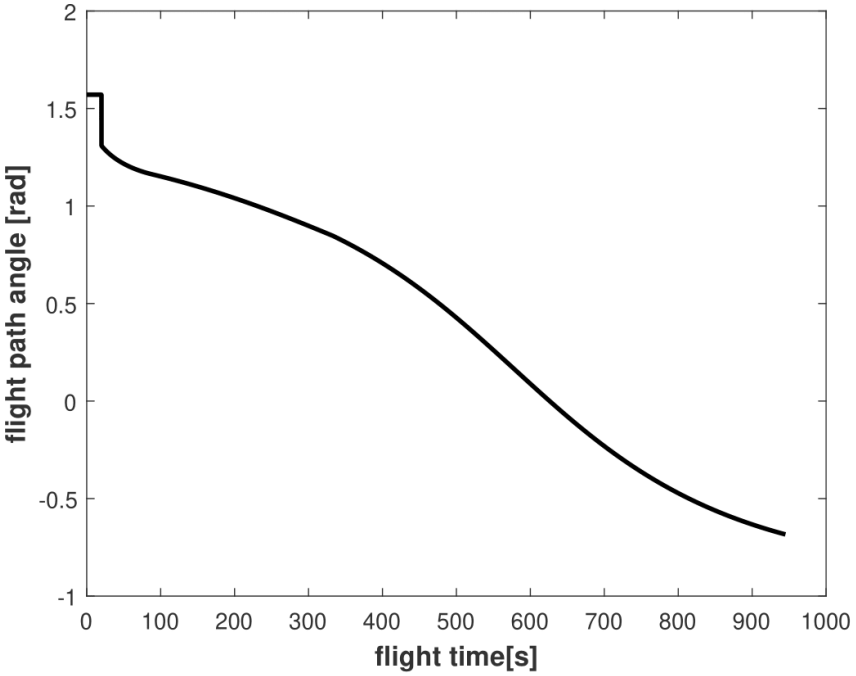

Figure 5: Graph showing how flight path angle of the primary launcher varies with time.

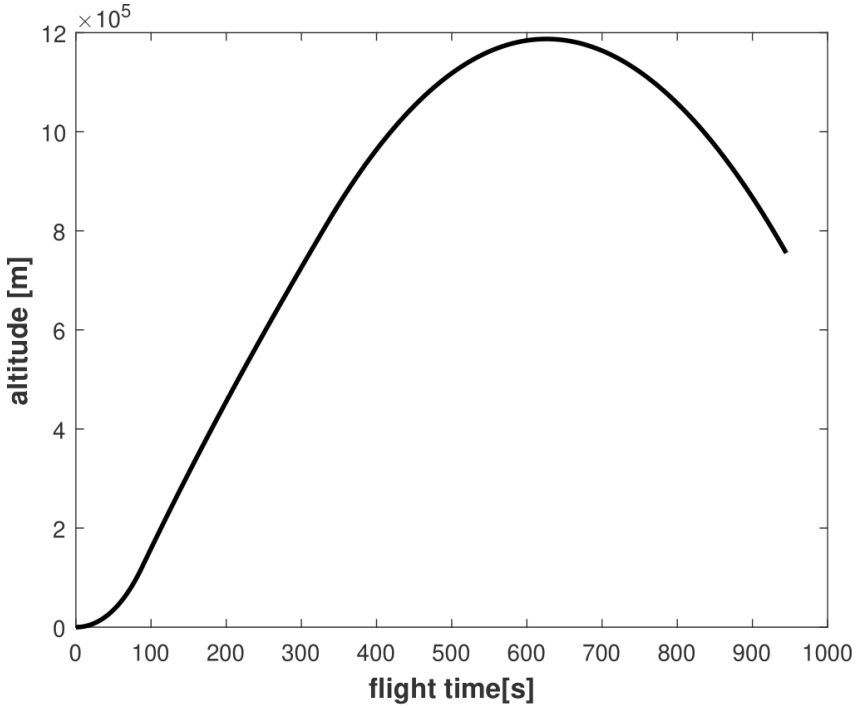

Figure 6: Graph showing how altitude of the primary launcher varies with time.

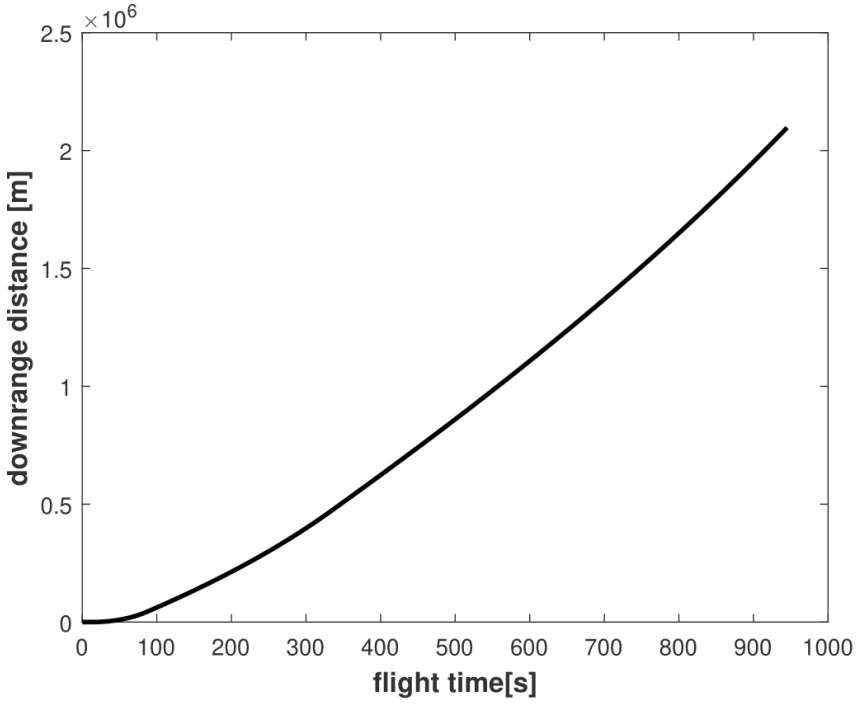

Figure 7: Graph showing how downrange distance of the primary launcher varies with time.

3.3.2 Primary Launcher - GTO

The following graphs show the trajectory simulation for the primary launcher, with a payload mass of 12,000kg, and throttle percentage of 100%. This aligns with the goal for getting a 12,000kg payload to GTO.

In the same way the boosters are overpowered for the LEO, they are for a GTO mission aswell. In this case however the increased speed this allows for is beneficial for what is needed to maintain geostationary orbit.

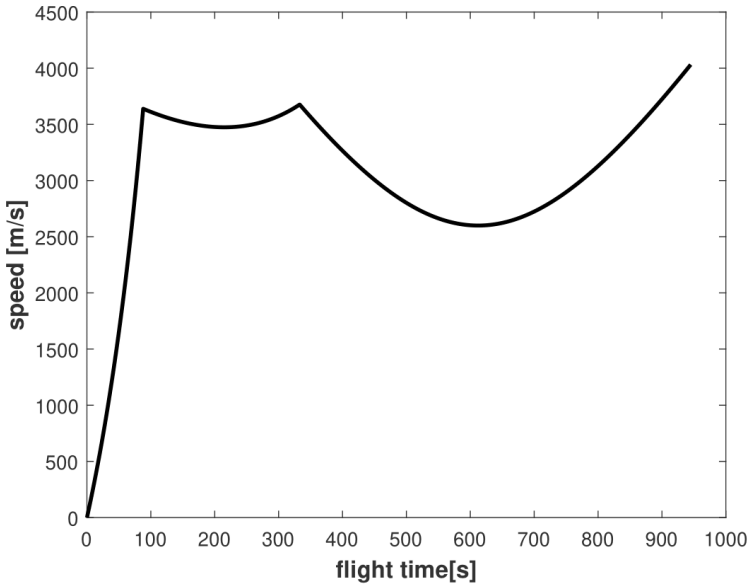

Figure 8: Graph showing how speed of the primary launcher varies with time.

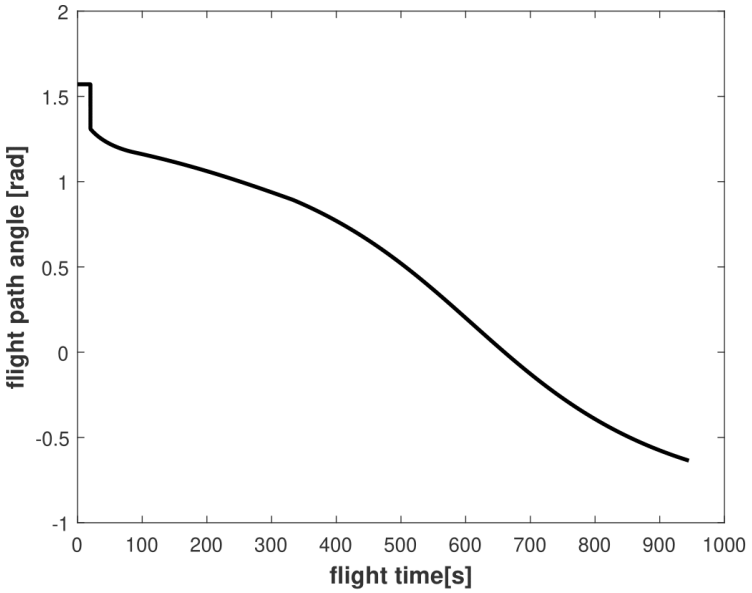

Figure 9: Graph showing how flight path angle of the primary launcher varies with time.

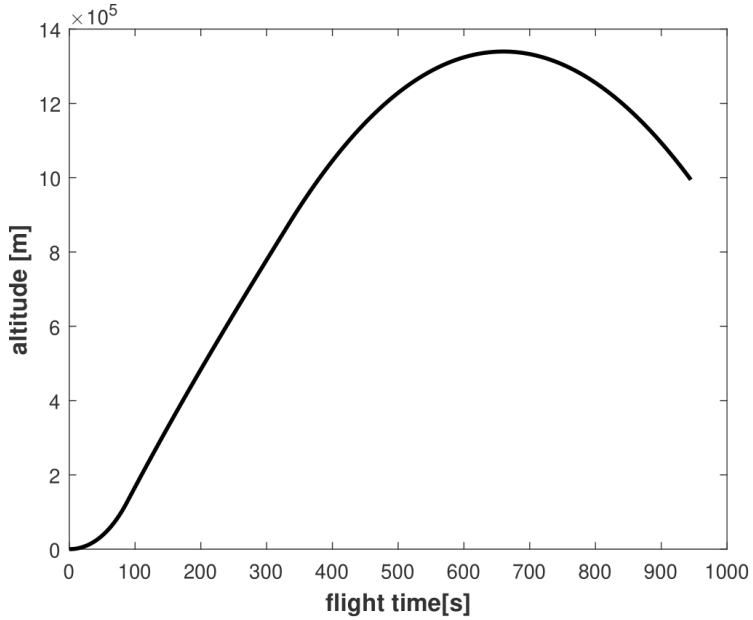

Figure 10: Graph showing how altitude of the primary launcher varies with time.

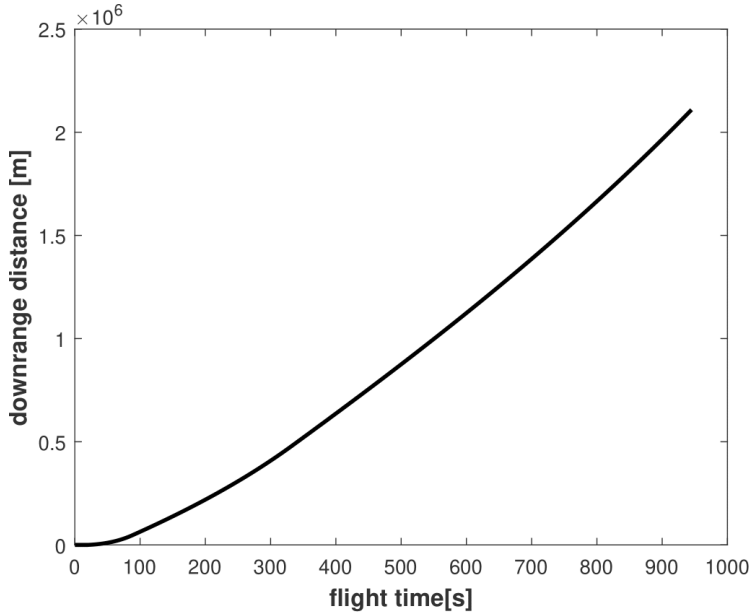

Figure 11: Graph showing how downrange distance of the primary launcher varies with time.

3.3.3 Secondary Launcher

The following graphs show the trajectory simulation for the primary launcher, with a payload mass of 12,500kg. Since there is no booster on the secondary configuration no throttle input was necessary.

The altitude graph for the secondary launcher has been omitted. This is the graph generated would show negative values of altitude, and the code not be fixed in time for submission for a correct altitude graph to be generated. The other graphs however should remain accurate so have been

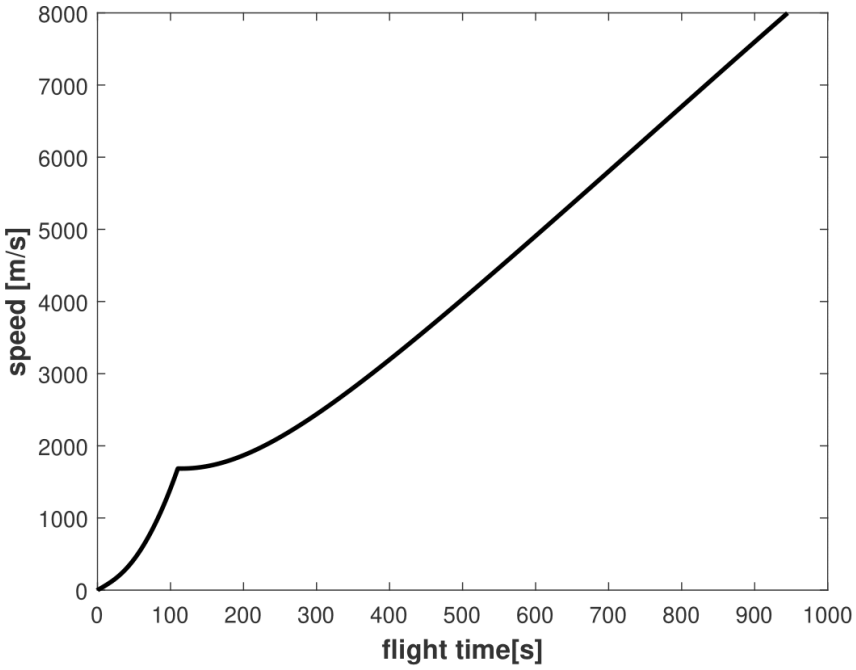

Figure 12: Graph showing how speed of the secondary launcher varies with time.

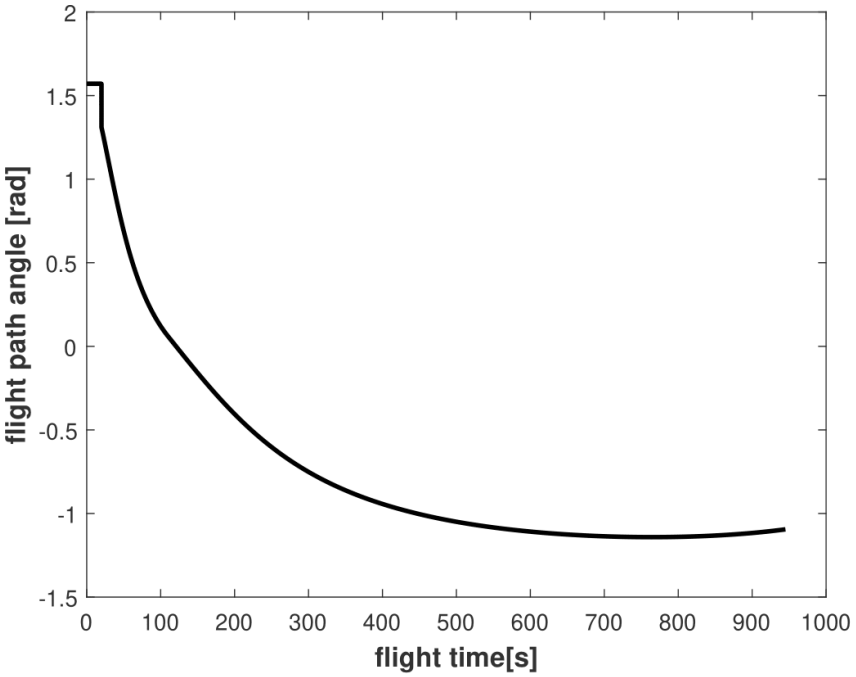

Figure 13: Graph showing how flight path angle of the secondary launcher varies with time.

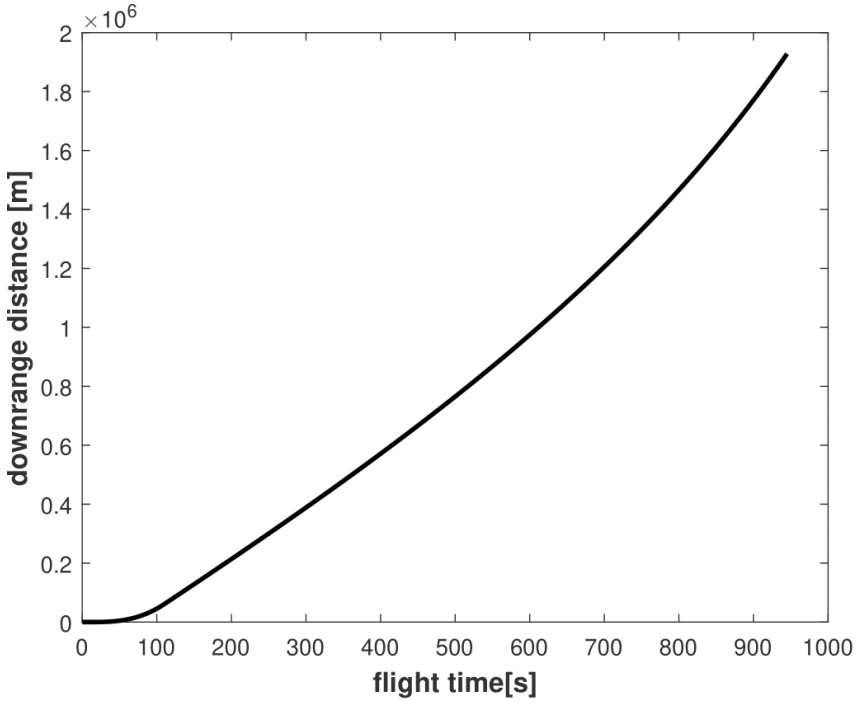

Figure 14: Graph showing how downrange distance of the secondary launcher varies with time.

4 Re-usable Core stage & Booster Analysis

4.1 Core Stage

4.1.1 Main Features

After thorough analysis, deploy able fold-out wings have been selected to aid with the core stage re-entry. Since, the wings will fold out from the core stage, the max chord length should be less than half the diameter of the core stage to accommodate 2 wings. The wingspan shall also be less than the total length of the core stage.
The choice of aerofoil for our wing is the NACA 63-210 (figure 4). The best aerofoil for gliding is one that is thin, ideally with a t/c ratio of 10%, which the NACA 63-210 has.

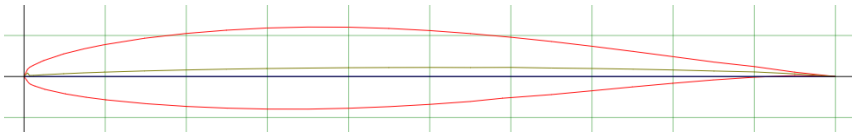

Figure 15: NACA 63-210 aerofoil

4.1.2 Wing Parameters for Primary Launcher

The empty weight of the primary core stage on descent is equal to its inert mass.

$$M_{inert} = 26726 \text{ kg}$$

The weight of the core stage is equal to the inert mass multiplied by g, the acceleration due to gravity

$$W = M_{inert} \cdot g = 26726 \cdot 9.81 = 262182 \text{ N}$$

For the subsequent calculations, it is assumed that on descent, the weight of the core stage is approximately equal to the lift force L. An important parameter in the wing design is the wing reference area S. Based on the size of the core stage and its fuel tanks, the estimated size of the wingspan b is 31.16 m. Given that the chord length of the wing must be less than half the diameter of the core stage of 5.2 m, the estimated chord length c of the wing is 2.4 m. The product of the chord length and wing span is the wing reference area S

$$S = b \cdot c = 31.16 \cdot 2.4 = 74.78 \text{ m}^2$$

Two important aspects that need to be predicted are the approach speed and distance required to stop. The stall speed of the core stage can be calculated using an altered formula for the lift coefficient:

$$C_{L,max} = \frac{L}{\frac{1}{2}\rho V_s^2 S}$$

where $C_{L,max}$ is the maximum lift coefficient and V_s is the stall speed (m/s). A typical value for the lift coefficient when landing is normally less than 1, so a reasonable value to choose for $C_{L,max}$ is 0.9 . Rearranging for V_s and assuming that L is approximately equal to W, the following is obtained:

$$V_s = \sqrt{\frac{L}{\frac{1}{2}\rho_{sl} S C_{L,max}}} = \sqrt{\frac{262182}{\frac{1}{2} \cdot 1.225 \cdot 74.78 \cdot 0.9}} = 79.8 \text{ m/s}$$

Of course, it is unsafe to fly at stall speed, so a more reasonable approach speed is $1.3V_s$, which is 103.7 m/s
To calculate the landing distance, the following formula provides a good estimate:

$$S_{landing} = 5\ (\frac{W}{S})\frac{1}{\sigma C_{L,max}} + S_a \ \ ; \ \ \sigma = \frac{\rho}{\rho_{sl}}$$

where $\frac{W}{S}$ is the wing loading (kg/m^2), and S_a is the obstacle-clearance distance, which is estimated as 305 m. Assuming that the landing site is approximately near sea level, σ is equal to 1. Plugging in the values yields the following:

$$S_{landing} = 5 \cdot \frac{26726}{74.48} \cdot \frac{1}{1 \cdot 0.9} + 305 = 2290.5 \text{ m}$$

Most major runways are 2500-3000 m in length so the core stage should not have a problem with stopping. The approach speed of 103.7 m/s is high, but because the core stage is light, there should be sufficient braking power from the spoilers on the wings to stop it in time once it lands.

4.1.3 Wing Parameters for Secondary Launcher

The only difference between the core stage of the primary and secondary launchers is that the secondary core stage has 2 additional Raptor engines to compensate for the missing boosters. The mass of the 2 extra engines is 3200 kg, so the inert mass for the secondary core stage increases to 29926 kg. The weight of this core stage is:

$$W = 29926 \cdot 9.81 = 293574 \text{ N}$$

However, all other parameters are the same as the primary launcher, so the key values for the secondary launcher are as follows:

$$V_s = \sqrt{\frac{293574}{\frac{1}{2} \cdot 1.225 \cdot 74.78 \cdot 0.9}} = 84.4 \text{ m/s}$$

$$V_{approach} = 1.3 \cdot V_s = 1.3 \cdot 84.4 = 109.7 \text{ m/s}$$

$$S_{landing} = 5 \cdot \frac{29926}{74.48} \cdot \frac{1}{1 \cdot 0.9} + 305 = 2528.3 \text{ m}$$

4.2 Boosters

4.2.1 Main Features

It has been decided that the most conventional approach of using parachutes to recover boosters is the most ideal. This will save on development costs and will greatly reduce the complexity of the project saving time and money. However, it has been decided against using buoys as the buoyant force acting on the boosters is much higher than the inert weight of them as calculated in section 4.2.4.
We have also chosen an elliptical cone with a fineness ratio of 2:1 for the booster. The lower fineness ratio helps with the huge forces the booster will undergo while also being aerodynamic during ascent.

4.2.2 Parameters of Liquid Rocket Booster

Parameter	Value
Maximum Diameter (d)	3.05 m
Maximum Radius (r)	1.525 m
Length of cylindrical section (l)	31.13 m
Height of Nosecone (h)	6.1 m
Mass of Raptor Engine	1600 kg
Number of Engines	3

4.2.3 Parachute Sizing Calculations

Using the following formula we can estimate the total area of the parachutes needed which is given by S_t.

$$D = W = \frac{\rho \cdot V^2 \cdot C_d \cdot S_t}{2}$$

W is the inert weight of the booster which is calculated by multiplying the inert mass of the rocket with the gravity acceleration constant.

$$W = 15000 \cdot 9.81 \text{ N} = 147150\text{N}$$

ρ is the density of air at sea-level

$$\rho = 1.225 \text{ kg/m}^3$$

C_d is the drag co-efficient and we have selected 1.6 as an appropriate value for a parachute for low speed descents.

$$C_d = 1.6$$

V_t is the terminal velocity at which the booster will descend. We have selected a value of 5 m/s to lower maintenance costs as well as the probability of inner components being damaged when the booster comes into contact with the sea water.

$$V_t = 5 \text{ m/s}$$

From the selected values,

$$S_t = 6006.12 \text{ m}^2$$

Each booster will have 3 parachutes. This choice was made to increase safety in-case of shock chord failure and/or parachute tearing, as the other 2 parachutes will be able to slow down the booster such that the damage is minimised. Thus, one parachute will have an area of 2002.04 m^2 and a diameter of 35.7 m.

4.2.4 Buoyant Force Calculations

Using the formula below we can obtain the buoyant force acting on our rocket booster.

$$B_f = \rho_{sw} \cdot V_{fd} \cdot g$$

ρ_{sw} is the density of sea-water and it is equal to 1025 kg/m^3.
g is the gravity constant is is equal to 9.81 m/s^2
V_{fd} is the volume of the displaced fluid and is equal to the volume of the booster.

$$V_{fd} = V_{cylinder} + V_{nosecone} + 3 \cdot V_{engine}$$

$V_{cylinder}$ is the volume of the cylindrical part of the booster and is calculated by

$$V_{cylinder} = \pi \cdot r^2 \cdot l = \pi \cdot 1.525^2 \cdot 31.13 = 227.441 \text{ m}^3$$

$V_{nosecone}$ is the volume of the nosecone and is calculated by,

$$V_{nosecone} = \frac{\pi \cdot d^2 \cdot h}{6} = \frac{\pi \cdot 3.05^2 \cdot 6.1}{6} = 29.712 \text{ m}^3$$

V_{engine} is the volume of the raptor engine and is calculated by,

$$V_{engine} = \frac{Mass\ of\ Engine}{Density\ of\ Engine}$$

Here, the mass of the engine is 1600 kg and the density of a Titanium Alloy (Ti6A14V) is used as an approximate and has a value of 4430 kg/m^3.

$$V_{engine} = \frac{1600}{4430} = 0.361 \text{ m}^3$$

Therefore the final volume calculation becomes,

$$V_{fd} = 227.441 + 29.712 + 3 \cdot 0.361 = 258.233 \text{ m}^3$$

Thus, the buoyant force is calculated to be

$$B_f = 1025 \cdot 258.233 \cdot 9.81 = 2596597.373 \text{ N}$$

Since inert weight of the booster is 147150 N and is lower than the buoyant force, it will float on water. Thus, the addition of buoys is redundant.

5 Drawing/CAD model of your launch system

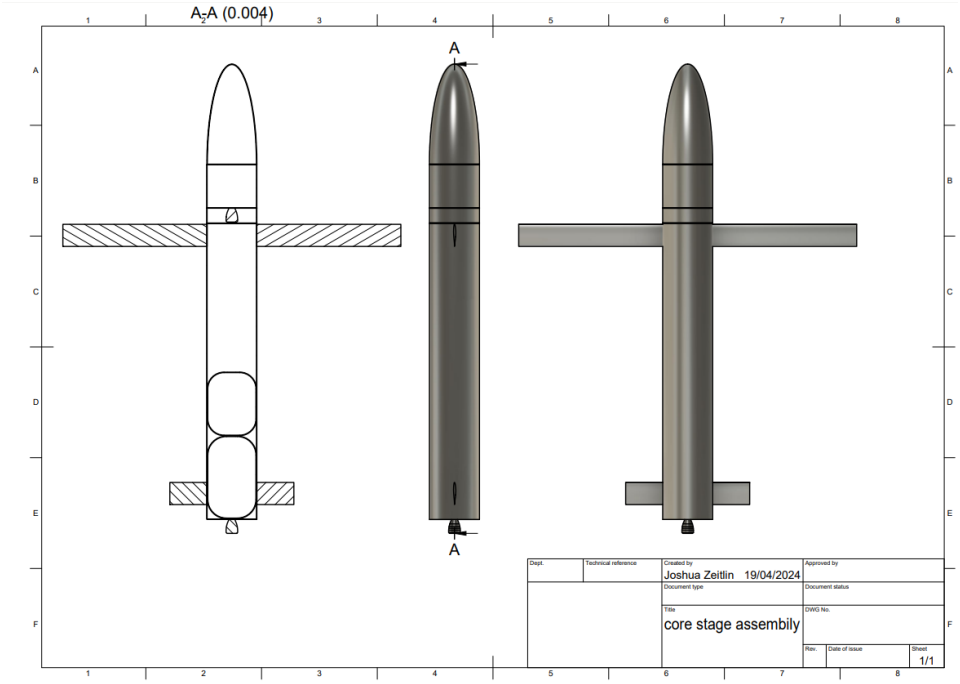

Figure 16: Core Stage - CAD

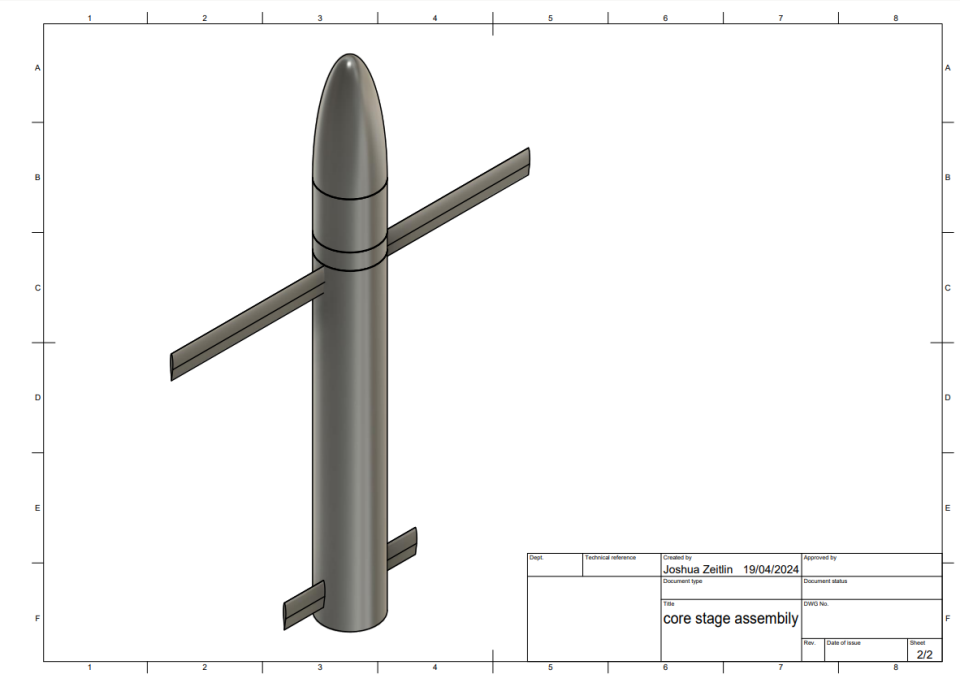

Figure 17: Core Stage Isometric View - CAD

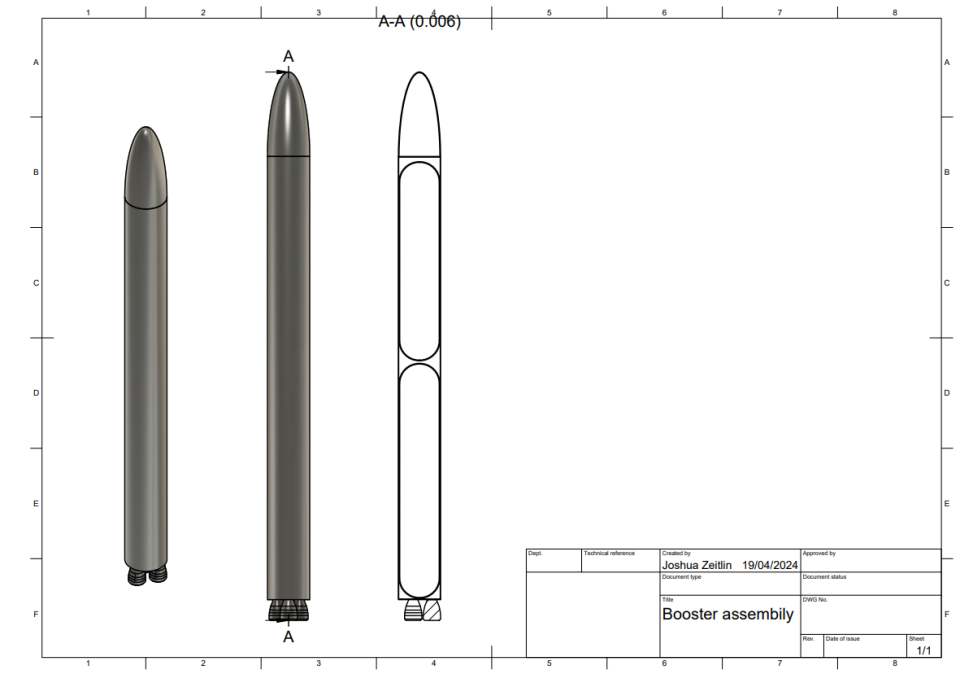

Figure 18: Booster - CAD

6 Conclusions

In this course we have learnt about launcher design and analysis as well as spacecraft dynamics. The design project taught us to think outside the box and use various tools and topics from other courses in conjunction with what we learnt in this course to design a new family of launchers.

The main challenge we faced was estimating values to use in calculations as this is a skill that only comes with experience, however we feel we have improved tremendously in this aspect after developing our design.

If we could continue on our concept design, we would improve upon the booster design as well as the weight distribution of the overall launcher. We would also liked to have done CFD analysis on our design using a fluid simulation software like Star-CCM+.

References

[1] V. Krillov, "Baikal reusable booster," *Moscow Defense Brief*, 2010.

[2] A. Zak, "Rockets: Angara family," 2008.

[3] DLR, "Astra lfbb configuration," 2015.

[4] P. B. de Selding, "Meet adeline, airbus' answer to spacex reusability," 2015.

Appendices

A Trajectory Simulation Code

```
clear;

% Constants

Rearth = 6371.0*1000.0; % Earth radius
deg2rad = 0.0174532; % used to convert degrees to radians

% Constants Over

% Configuration

tvert = 20; % assume 20 seconds of vertical flight
theta_fix = 10.0*deg2rad; % angle at which grativity turn is
    limited - start 'fixed-pitch' part of flight

dt = 0.1;

% GET LAUNCHER VARIANT

launchInput = getInput("For which launcher configuration should
    this simulation run? \n \n1: Primary Launcher\n2: Secondary
    Launcher\n", false);
payloadInput = getInput("Whats the payload? (in kg)", false);

if (launchInput == 1)
    boosterThrottleInput = getInput("Adjust throttle input? (
       Press enter to use default value of 100%) (Percentage
       integer)", true);
    if isempty(boosterThrottleInput)
        boosterThrottleInput = 1
    else
        boosterThrottleInput = boosterThrottleInput/100
    end
    Tphase1 = (2260 + 2*6780*boosterThrottleInput)*1000.0; %
       lift-off thrust
    Tphase2 = 2530.0*1000.0; % core-stage thrust
    Tphase3 = 62.7*1000.0; % upper-stage thrust

    tbo1 = 88/boosterThrottleInput; % burn-out time SRBs
    tbo2 = 333; % burn time for core
    tbo3 = 945; % burn time for upper stage

    M01 = 428056+payloadInput;
    M02 = M01 - 30000 - 332.6*tbo1; % 3175kg = payload fairing
    M03 = 25000+19440;

    mdot1 = 650 + 1950*boosterThrottleInput; % mass-flow rate
       for phase 1
```

```
42      mdot2 = 650; % mass-flow rate for phase 2
43      mdot3 = 14.8; % mass-flow rate for phase 3
44
45      calculate_primary(Tphase1, Tphase2, Tphase3, tbo1, tbo2,
            tbo3, M01, M02, M03, mdot1, mdot2, mdot3, dt, theta_fix,
             tvert, Rearth, deg2rad)
46  elseif (launchInput == 2)
47      Tphase1 = 6780*1000.0; % lift-off thrust
48      Tphase2 = 62.7*1000.0; % upper-stage thrust
49
50      tbo1 = 110; % burn-out time CORE
51      tbo2 = 945; % burn time for US
52
53      mdot1 = 1950; % mass-flow rate for phase 1
54      mdot2 = 14.8; % mass-flow rate for phase 2
55
56      M01 = 430316 + payloadInput                 ;
57      M02 = M01 - 242141; % 3175kg = payload fairing
58
59      calculate_secondary(Tphase1, Tphase2, tbo1, tbo2, M01, M02,
            mdot1, mdot2, dt, theta_fix, tvert, Rearth, deg2rad)
60  end
61
62
63
64  function calculate_primary(Tphase1, Tphase2, Tphase3, tbo1,
        tbo2, tbo3, M01, M02, M03, mdot1, mdot2, mdot3, dt,
        theta_fix, tvert, Rearth, deg2rad)
65      % Config Over
66
67      acc = Tphase1/M01; % initial acceleration
68
69
70      time(1) = dt;
71      velx(1) = 0.0;
72      velz(1) = 0.0;
73      vel(1) = 0.0;
74
75      xpos(1) = 0.0;
76      zpos(1) = 0.0;
77
78      gamma(1) = 90.0*deg2rad; % initial vertical flight
79
80      for k=2:(tvert/dt)
81          time(k) = time(k-1) + dt;
82
83          velx(k) = 0.0;
84          velz(k) = velz(k-1) + acc*dt;
85          vel(k) = sqrt(velx(k)*velx(k) + velz(k)*velz(k));
86
87          acc = Tphase1/(M01 - mdot1*time(k)) - 9.81; % update
                acceleration to account for changing mass
88
```

```
89          xpos(k) = 0.0;
90          zpos(k) = zpos(k-1) + velz(k-1)*dt + 0.5*acc*dt*dt;
91
92          gamma(k) = gamma(k-1);
93      end
94
95      % initiate gravity turn
96      %
97      % consider time from 20s to 114s
98      %
99      gamma_init = 75.0*deg2rad;
100
101     gamma(200) = gamma_init; % step-wise change of flight-path
            angle - initiates gravity turn
102
103     acc = Tphase1/(M01 - mdot1*time(200));
104
105     for k=(tvert/dt + 1):(tbo1/dt)
106         time(k) = time(k-1) + dt;
107
108         vel(k) = vel(k-1) + dt*(acc - 9.81*sin(gamma(k-1)));
109         velx(k) = vel(k)*cos(gamma(k-1));
110         velz(k) = vel(k)*sin(gamma(k-1));
111
112         xpos(k) = xpos(k-1) + velx(k)*dt + 0.5*acc*cos(gamma(k
                -1))*dt*dt;
113         zpos(k) = zpos(k-1) + velz(k)*dt + 0.5*(acc*sin(gamma(k
                -1))-9.81)*dt*dt;
114         % update flight-path angle due to centripetal and
                gravity effect
115         gamma(k) = gamma(k-1) + vel(k)*cos(gamma(k-1))*dt/(
                Rearth+zpos(k-1)) - 9.81*cos(gamma(k-1))*dt/vel(k-1)
                ;
116         acc = Tphase1/(M01 - mdot1*time(k));
117     end
118
119     % 2nd phase of launch - from 114 to 352s
120     acc = Tphase2/M02;
121
122     for k=(tbo1/dt + 1):(tbo2/dt)
123         time(k) = time(k-1) + dt;
124
125         if gamma(k-1) > theta_fix
126             % continue with gravity turn
127             vel(k) = vel(k-1) + dt*(acc - 9.81*sin(gamma(k-1)))
                    ;
128             velx(k) = vel(k)*cos(gamma(k-1));
129             velz(k) = vel(k)*sin(gamma(k-1));
130
131             xpos(k) = xpos(k-1) + velx(k)*dt + 0.5*acc*cos(
                    gamma(k-1))*dt*dt;
132             zpos(k) = zpos(k-1) + velz(k)*dt + 0.5*(acc*sin(
                    gamma(k-1))-9.81)*dt*dt;
```

```
133             % update flight-path angle due to centripetal and
                    gravity effect
134             gamma(k) = gamma(k-1) + vel(k)*cos(gamma(k-1))*dt/(
                    Rearth+zpos(k-1)) - 9.81*cos(gamma(k-1))*dt/vel(
                    k-1);
135             acc = Tphase2/(M02 - mdot2*(time(k)-tbo1));
136         else
137             % fixed-pitch part of flight
138             vel(k) = vel(k-1) + dt*acc*sin(theta_fix)*sin(gamma
                    (k-1))+dt*acc*cos(theta_fix)*cos(gamma(k-1)) -
                    dt*9.81*sin(gamma(k-1));
139             velx(k) = vel(k)*cos(gamma(k-1));
140             velz(k) = vel(k)*sin(gamma(k-1));
141
142             xpos(k) = xpos(k-1) + velx(k)*dt + 0.5*acc*cos(
                    theta_fix)*dt*dt;
143             zpos(k) = zpos(k-1) + velz(k)*dt + 0.5*(acc*sin(
                    theta_fix)-9.81)*dt*dt;
144
145             gamma(k) = gamma(k-1) + acc*(sin(theta_fix)*cos(
                    gamma(k-1)) - cos(theta_fix)*sin(gamma(k-1)))*dt
                    /vel(k-1) - 9.81*cos(gamma(k-1))*dt/vel(k-1) +
                    vel(k-1)*cos(gamma(k-1))*dt/(Rearth+zpos(k-1));
146
147             acc = Tphase2/(M02 - mdot2*(time(k) - tbo1));
148         end
149
150     end
151
152     % 3rd phase of launch - upper stage flying at fixed pitch
153     acc = Tphase3/M03;
154
155     for k=(tbo2/dt + 1):(tbo3/dt)
156         time(k) = time(k-1) + dt;
157
158         vel(k) = vel(k-1) + dt*acc*sin(theta_fix)*sin(gamma(k
                -1))+dt*acc*cos(theta_fix)*cos(gamma(k-1)) - dt
                *9.81*sin(gamma(k-1));
159         velx(k) = vel(k)*cos(gamma(k-1));
160         velz(k) = vel(k)*sin(gamma(k-1));
161
162         xpos(k) = xpos(k-1) + velx(k)*dt + 0.5*acc*cos(
                theta_fix)*dt*dt;
163         zpos(k) = zpos(k-1) + velz(k)*dt + 0.5*(acc*sin(
                theta_fix)-9.81)*dt*dt;
164
165         gamma(k) = gamma(k-1) + acc*(sin(theta_fix)*cos(gamma(k
                -1)) - cos(theta_fix)*sin(gamma(k-1)))*dt/vel(k-1) -
                 9.81*cos(gamma(k-1))*dt/vel(k-1) + vel(k-1)*cos(
                gamma(k-1))*dt/(Rearth+zpos(k-1));
166
167         acc = Tphase3/(M03 - mdot3*(time(k) - tbo2));
168     end
```

```
169     fprintf("\n\nCreating graphs, this may take some time.
            Please wait.")
170
171     plot(time,vel,'-k','Linewidth',2)
172     xlabel('flight time[s]','Fontweight','bold','Fontsize',12)
173     ylabel('speed [m/s]','Fontweight','bold','Fontsize',12)
174     exportgraphics(gcf, "output.pdf")
175
176     figure(2)
177     plot(time,gamma,'-k','Linewidth',2)
178     xlabel('flight time[s]','Fontweight','bold','Fontsize',12)
179     ylabel('flight path angle [rad]','Fontweight','bold','
            Fontsize',12)
180
181     exportgraphics(gcf,'output.pdf','Append',true)
182
183     figure(3)
184     plot(time,zpos,'-k','Linewidth',2)
185     xlabel('flight time[s]','Fontweight','bold','Fontsize',12)
186     ylabel('altitude [m]', 'Fontweight','bold','Fontsize',12)
187
188     exportgraphics(gcf,'output.pdf','Append',true)
189
190     figure(4)
191     plot(time,xpos,'-k','Linewidth',2)
192     xlabel('flight time[s]','Fontweight','bold','Fontsize',12)
193     ylabel('downrange distance [m]','Fontweight','bold','
            Fontsize',12)
194
195     exportgraphics(gcf,'output.pdf','Append',true)
196
197     fprintf("\n\nProgram Complete")
198
199     winopen('output.pdf')
200
201     grid on
202
203 end
204
205 function calculate_secondary(Tphase1, Tphase2, tbo1, tbo2, M01,
        M02, mdot1, mdot2, dt, theta_fix, tvert, Rearth, deg2rad)
206     acc = Tphase1/M01; % initial acceleration
207
208     time(1) = dt;
209     velx(1) = 0.0;
210     velz(1) = 0.0;
211     vel(1) = 0.0;
212
213     xpos(1) = 0.0;
214     zpos(1) = 0.0;
215
216     gamma(1) = 90.0*deg2rad; % initial vertical flight
217
```

```
    for k=2:(tvert/dt)
        time(k) = time(k-1) + dt;

        velx(k) = 0.0;
        velz(k) = velz(k-1) + acc*dt;
        vel(k) = sqrt(velx(k)*velx(k) + velz(k)*velz(k));

        acc = Tphase1/(M01 - mdot1*time(k)) - 9.81; % update
            acceleration to account for changing mass

        xpos(k) = 0.0;
        zpos(k) = zpos(k-1) + velz(k-1)*dt + 0.5*acc*dt*dt;

        gamma(k) = gamma(k-1);
    end
    % initiate gravity turn
    %
    % consider time from 20s to 114s
    %
    gamma_init = 75.0*deg2rad;

    gamma(200) = gamma_init; % step-wise change of flight-path
        angle - initiates gravity turn

    acc = Tphase1/(M01 - mdot1*time(200));

    for k=(tvert/dt + 1):(tbo1/dt)
        time(k) = time(k-1) + dt;

        vel(k) = vel(k-1) + dt*(acc - 9.81*sin(gamma(k-1)));
        velx(k) = vel(k)*cos(gamma(k-1));
        velz(k) = vel(k)*sin(gamma(k-1));

        xpos(k) = xpos(k-1) + velx(k)*dt + 0.5*acc*cos(gamma(k
            -1))*dt*dt;
        zpos(k) = zpos(k-1) + velz(k)*dt + 0.5*(acc*sin(gamma(k
            -1))-9.81)*dt*dt;
        % update flight-path angle due to centripetal and
            gravity effect
        gamma(k) = gamma(k-1) + vel(k)*cos(gamma(k-1))*dt/(
            Rearth+zpos(k-1)) - 9.81*cos(gamma(k-1))*dt/vel(k-1)
            ;
        acc = Tphase1/(M01 - mdot1*time(k));
    end

    acc = Tphase2/M02;

    for k=(tbo1/dt + 1):(tbo2/dt)
        time(k) = time(k-1) + dt;

        if gamma(k-1) > theta_fix
            % continue with gravity turn
```

```
263             vel(k) = vel(k-1) + dt*(acc - 9.81*sin(gamma(k-1)))
                    ;
264             velx(k) = vel(k)*cos(gamma(k-1));
265             velz(k) = vel(k)*sin(gamma(k-1));
266
267             xpos(k) = xpos(k-1) + velx(k)*dt + 0.5*acc*cos(
                    gamma(k-1))*dt*dt;
268             zpos(k) = zpos(k-1) + velz(k)*dt + 0.5*(acc*sin(
                    gamma(k-1))-9.81)*dt*dt;
269             % update flight-path angle due to centripetal and
                    gravity effect
270             gamma(k) = gamma(k-1) + vel(k)*cos(gamma(k-1))*dt/(
                    Rearth+zpos(k-1)) - 9.81*cos(gamma(k-1))*dt/vel(
                    k-1);
271             acc = Tphase2/(M02 - mdot2*(time(k)-tbo1));
272         else
273             % fixed-pitch part of flight
274             vel(k) = vel(k-1) + dt*acc*sin(theta_fix)*sin(gamma
                    (k-1))+dt*acc*cos(theta_fix)*cos(gamma(k-1)) -
                    dt*9.81*sin(gamma(k-1));
275             velx(k) = vel(k)*cos(gamma(k-1));
276             velz(k) = vel(k)*sin(gamma(k-1));
277
278             xpos(k) = xpos(k-1) + velx(k)*dt + 0.5*acc*cos(
                    theta_fix)*dt*dt;
279             zpos(k) = zpos(k-1) + velz(k)*dt + 0.5*(acc*sin(
                    theta_fix)-9.81)*dt*dt;
280
281             gamma(k) = gamma(k-1) + acc*(sin(theta_fix)*cos(
                    gamma(k-1)) - cos(theta_fix)*sin(gamma(k-1)))*dt
                    /vel(k-1) - 9.81*cos(gamma(k-1))*dt/vel(k-1) +
                    vel(k-1)*cos(gamma(k-1))*dt/(Rearth+zpos(k-1));
282
283             acc = Tphase2/(M02 - mdot2*(time(k) - tbo1));
284         end
285     end
286     fprintf("\n\nCreating graphs, this may take some time.
            Please wait.")
287
288     plot(time,vel,'-k','Linewidth',2)
289     xlabel('flight time[s]','Fontweight','bold','Fontsize',12)
290     ylabel('speed [m/s]','Fontweight','bold','Fontsize',12)
291     exportgraphics(gcf, "output.pdf")
292
293     figure(2)
294     plot(time,gamma,'-k','Linewidth',2)
295     xlabel('flight time[s]','Fontweight','bold','Fontsize',12)
296     ylabel('flight path angle [rad]','Fontweight','bold','
            Fontsize',12)
297
298     exportgraphics(gcf,'output.pdf','Append',true)
299
300     figure(3)
```

```
    plot(time,zpos,'-k','Linewidth',2)
    xlabel('flight time[s]','Fontweight','bold','Fontsize',12)
    ylabel('altitude [m]', 'Fontweight','bold','Fontsize',12)

    exportgraphics(gcf,'output.pdf','Append',true)

    figure(4)
    plot(time,xpos,'-k','Linewidth',2)
    xlabel('flight time[s]','Fontweight','bold','Fontsize',12)
    ylabel('downrange distance [m]','Fontweight','bold','
       Fontsize',12)

    exportgraphics(gcf,'output.pdf','Append',true)

    fprintf("\n\nProgram Complete")

    winopen('output.pdf')

    grid on

end

function out = getInput(prompt, blankable)
    inp = input(prompt, "s");
    inp_number = str2double(inp);

    if ~isnan(inp_number) && inp_number > 0
        out = inp_number;
    elseif inp == "" && blankable
        out = inp;
    else
        fprintf("\nPlease input a valid positive number\n");
        out = getInput(prompt);
    end
end
```