**BRNO UNIVERSITY OF TECHNOLOGY**
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**

**FACULTY OF INFORMATION TECHNOLOGY**
**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**

**DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA**
**ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ**

# DESIGN OF GUIDANCE, NAVIGATION AND CONTROL FOR VERTICAL LANDING OF A REUSABLE ROCKET BOOSTER

**NÁVRH NAVEDENÍ, NAVIGACE A ŘÍZENÍ PRO VERTIKÁLNÍ PŘISTÁNÍ OPAKOVANĚ POUŽITELNÉHO RAKETOVÉHO URYCHLOVAČE**

**MASTER'S THESIS**
**DIPLOMOVÁ PRÁCE**

**AUTHOR** **Bc. ADRIÁN KIRÁLY**
**AUTOR PRÁCE**

**SUPERVISOR** **doc. Ing. PETER CHUDÝ, Ph.D., MBA**
**VEDOUCÍ PRÁCE**

**BRNO 2020**

**Brno University of Technology**
Faculty of Information Technology

Department of Computer Graphics and Multimedia (DCGM) Academic year 2020/2021

# Master's Thesis Specification

23838

Student: **Király Adrián, Bc.**
Programme: Information Technology and Artificial Intelligence Specialization: Computer Graphics and Multimedia
Title: **Design of Guidance, Navigation and Control for Vertical Landing of a Reusable Rocket Booster**
Category: Modelling and Simulation
Assignment:

1. Create a dynamic model and research aerodynamic characteristics of a reusable rocket booster.
2. Get familiar with the design of spacecraft guidance, navigation and control system.
3. Design and create a Matlab implementation of guidance, navigation, and control for vertical landing of the booster.
4. Implement a visualization environment for an intuitive interpretation of the computed landing trajectory.
5. Evaluate achieved results and discuss potential further improvements.

Recommended literature:

- According to supervisor's recommendations.

Requirements for the semestral defence:

- Items No. 1, 2 and partially item No. 3.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor: **Chudý Peter, doc. Ing., Ph.D. MBA**
Head of Department: Černocký Jan, doc. Dr. Ing.
Beginning of work: November 1, 2020
Submission deadline: May 19, 2021
Approval date: October 30, 2020

## Abstract
This master's thesis is focused on the development of guidance, navigation and control system for a reusable rocket booster. To achieve this goal, a simulation model of the rocket was developed using the Simulink environment. A custom aerodynamic model for this simulation was created, based on data obtained from CFD software. For demonstration of the achieved results, a 3D interactive visualization tool was also created as a part of this work.



## Abstrakt
Táto diplomová práca sa zaoberá vývojom systému pre navádzanie, navigáciu, a riadenie pre znovupoužiteľný raketový urýchľovač. Pre dosiahnutie tohto cieľu bol vytvorený simulačný model rakety v prostredí Simulink. Na základe dát získaných pomocou CFD softvéru bol pre túto simuláciu vytvorený tiež vlastný aerodynamický model. Pre účely demonštrácie dosiahnutých výsledkov bol ako súčasť práce tiež naprogramovaný interaktívny 3D vizualizačný nástroj.



## Keywords
reusable rocket booster, vertical landing, guidance, controls, aerodynamics, CFD, simulation, Simulink

## Kľúčové slová
znovupoužiteľný raketový urýchľovač, vertikálne pristátie, navádzanie, riadenie, aerodynamika, CFD, simulácia, Simulink

## Reference
KIRÁLY, Adrián. *Design of Guidance, Navigation and Control for Vertical Landing of a Reusable Rocket Booster*. Brno, 2020. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor doc. Ing. Peter Chudý, Ph.D., MBA

# Rozšírený abstrakt

Táto diplomová práca sa zaoberá vývojom systému pre navádzanie, navigáciu, a riadenie pre vertikálne pristátie znovupoužiteľného raketového urýchľovača. Na rozdiel od konvenčného prvého stupňa rakety, ktorý po splnení svojej primárnej misie spadne na Zem voľným pádom je v tom prípade urýchľovač cielene navádzaný po zostupovej trajektórií až na miesto pristátia.

Pre vývoj takéhoto riadiaceho systému je pochopiteľne potrebný prístup k systému, ktorý bude riadiť. Z tohto dôvodu musel byť pre potreby práce vyvinutý simulačný model, ktorý by reflektoval správanie raketového urýchľovača na Zemi a v jej blízkom okolí.

Za týmto účelom bol vytvorený dynamický model rakety so šiestimi stupňami voľnosti, ktorý definuje pohyb urýchľovača v priestore v závislosti od síl na neho pôsobiacich. Dynamický model bol doplnený modelmi raketového pohonu a manévrovacích trysiek. Oba tieto systémy zabezpečujú ovládanie pohybu raketového urýchľovača.

Nakoľko väčšinu svojho letu strávi urýchľovač v atmosfére, nie je možné zanedbať ani vplyv aerodynamických síl. Keďže ale nebol nájdený žiadny vhodný voľne dostupný aerodynamický model ani merania na základe ktorých by ho bolo možné vytvoriť, bolo nutné využiť CFD softvér pre získanie aerodynamických charakteristík urýchľovača. Na ich základe bol vytvorený aerodynamický model využitý v simulácii.

Ďalšími nutnými súčasťami simulácie sú modely prostredia, v ktorom raketa letí. V tomto prípade bol využitý štandardný atmosferický model a model gravitačného poľa Zeme. Zatiaľ čo atmosferický model je nevyhnutný pre výpočet aerodynamických síl a momentov, potreba samostatného gravitačného modelu je spôsobená tým, že vo výškach ktoré urýchľovač počas letu dosiahne už nastáva významná zmena gravitačnej konštanty.

Následne bol s pomocou vytvoreného simulačného prostredia vyvinutý jednoduchý systém pre navádzanie, navigáciu, a riadenie, ktorý riadi let od štartu až po pristátie. Aj keď sa bohužial nepodarilo zahrnúť komplexnejšie metódy riadenia, systém podáva zaujímavé výsledky a predstavuje dobrý štartovací bod a rámec pre ďalšie experimenty v tejto oblasti.

Vizualizácia výsledkov experimentov je neodmysliteľnou súčasťou práce so simuláciami. Vzhľadom na komplexnosť pohybu so šiestimi stupňami voľnosti je vhodná vizualizácia kľučová pre správne a rýchle pochopenie výsledkov. Preto bol v rámci práce naprogramovaný aj interaktívny 3D vizualizačný nástroj.

Tento nástroj zobrazuje pohyb a orientáciu urýchľovača vzhľadom na virtuálny model Zeme doplnený o výškové mapy a satelitné fotomapy vďaka knižnici Cesium, čo zjednodušuje určovanie polohy. Pre zobrazenie negeografických veličín je vizualizačný nástroj doplnený panelmi s výpismi aktuálnych hodnôt alebo schematickými nákresmi, ktoré reflektujú aktuálny stav.

# Design of Guidance, Navigation and Control for Vertical Landing of a Reusable Rocket Booster

## Declaration

Hereby I declare that this master's thesis was prepared as an original author's work under the supervision of doc. Ing. Peter Chudý, Ph.D., MBA. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

........................
Adrián Király
May 17, 2021

## Acknowledgements

I would like to thank my supervisor doc. Ing. Peter Chudý, Ph.D., MBA for his excellent guidance and invaluable feedback and advice, my family for supporting me all this time, and my girlfriend for being there for me when I needed it the most.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

For more than 60 years, humanity has relied on conventional, expendable rocket launchers to deliver payloads and people to space. However, as each of those rockets was essentially single-use, the cost of getting to space has been exceptionally high for a long time.

There were multiple attempts to resolve this issue throughout history, most notably the *Space Shuttle*. While it became the first reusable vehicle to reach orbit and provided unprecedented capabilities, it never fulfilled its aim of reducing the cost of getting to space.

Another notable project was the *McDonnell Douglas DC-X*, which was one of the first working prototypes of a Vertical Takeoff, Vertical Landing (VTVL) vehicle. Unlike the Shuttle, it resembled conventional multi-stage rockets, but it landed vertically back to Earth after liftoff. More than 20 years later, the vision of the DC-X project became a reality when the first stage of *SpaceX Falcon 9* successfully landed back to Earth after delivering its payload to orbit.

This work aims to design and develop a Guidance, Navigation, and Control (GNC) system of such a vehicle using a software simulation of the booster and its environment. This is a complex task requiring the development of multiple interdependent parts and knowledge of various aerospace engineering areas, modeling, and simulation.

In the following chapters, a brief insight into these subjects will be provided and the basic principles explained, starting with the first part, the dynamic model of the rocket, presented in chapter 2. This chapter also covers coordinate systems and other prerequisites for the whole work.

As the booster of a reusable rocket spends a considerable portion of its flight in denser parts of Earth's atmosphere, it is necessary to consider its aerodynamics. Chapter 3 provides an introduction to aerodynamics and specifies the forces and moments that act on the vehicle. Then, a Computational Fluid Dynamics (CFD) approach is used to create a simple aerodynamic model of the rocket used in this work.

The chapter 4 describes state-of-the-art in GNC design for spacecraft, which serves as a basis for building the GNC system in this work. Finally, the chapter 5 focuses on the last areas that need to be covered before building the simulation model. This chapter briefly explains the environment models necessary for such simulations and the basic driving principle behind the continuous simulation.

Then, the chapter 6 focuses directly on the implementation details of the simulation model and GNC system created in this work. Furthermore, as it is essential to present the results of the simulations in a comprehensible manner, this section also describes a visualization tool created for displaying the computed data. Lastly, the chapter 7 analyzes the data obtained from experiments with the simulation and discusses further improvements.

# Chapter 2

# Dynamic Model of a Reusable Rocket Booster

The first step in building the GNC system for the rocket booster is to create a mathematical model of the booster itself. This makes it possible to develop the system and simulate its function without building the rocket and performing (often costly) real-world experiments.

At the beginning of the section, several coordinate systems will be described. These coordinate systems are necessary not only for developing the booster's dynamic model, but will also be used throughout the work. Then, the kinematic equations will be described, and lastly, the dynamic model of the rocket will be built in the form of equations of motion.

## 2.1 Coordinate Systems and Transformations

To describe the position and movement of a rocket, it is necessary to establish a suitable coordinate system. However, a single coordinate system may not be sufficient for all purposes. In some cases, the use of a different coordinate system may simplify certain calculations, or it may be a natural way to express certain phenomena [8]. This, however, makes it necessary to perform transformations between various coordinate systems.

This section will introduce several reference frames and coordinate systems that will be used in this work. These definitions are in line with the coordinate systems used by the MATLAB Aerospace Toolbox [34]. Furthermore, the transformations between consecutive systems will be presented, making it possible to build a transformation between any two coordinate systems.

### 2.1.1 Earth-Centered Inertial Frame

In general, an inertial reference frame is a frame of reference that is not rotating or accelerating. The Earth-Centered Inertial frame (ECI) (see fig. 2.1) is a global frame with the origin at Earth's center of mass and fixed relative to the stars, which means the Earth's rotation can be observed in this frame. This makes it a preferred frame for describing the motion of a spacecraft in near-Earth environments [10].

The ECI frame is defined as follows [10]:

- The origin $o_I$ is at the center of mass of the Earth.
- The $x_I$ axis is pointing towards the vernal equinox (a point at which the ecliptic intersects the celestial equator [1]).

- The $z_I$ axis is parallel to the Earth's rotation axis, pointing towards the Conventional Terrestrial Pole (CTP).
- The remaining $y_I$ axis completes the right-handed coordinate system.

The equatorial plane and the ecliptic slightly move over time, which means the position of the vernal equinox changes [39]. To define a truly inertial frame, the position of the vernal equinox must be fixed. This is achieved by referring to its position at a particular point in time. Most commonly, the *J2000.0 Epoch* is used, which refers to January 1, 2000, at 12:00 Terestrial Time (TT).

### 2.1.2 Earth-Centered, Earth-Fixed Frame

The Earth-Centered, Earth-Fixed frame (ECEF), shown in fig. 2.1, is similar to the ECI frame, as it has the same origin and z-axis. The major difference is that ECEF is co-rotating with the Earth, which means that the coordinates of a given point do not change over time.

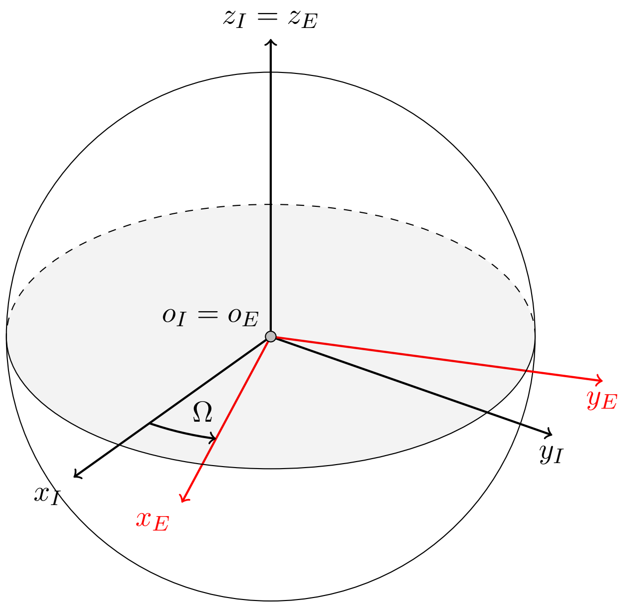



Figure 2.1: The Earth-Centered Inertial frame and Earth-Centered, Earth-Fixed frame are related through a single rotation by angle $\Omega$.

The full definition of the ECEF frame is following [21]:

- The origin $o_E$ is at the center of mass of the Earth.
- The $x_E$ axis is pointing towards the intersection of the prime meridian and the equatorial plane.
- The $z_E$ axis is parallel to the Earth's rotation axis, pointing towards the CTP.
- The remaining $y_E$ axis completes the right-handed coordinate system.

As can be seen from fig. 2.1, the ECEF and ECI frames are related by a rotation around a single axis ($x_E = x_I$), and the angle $\Omega$ of the rotation is a linear function of time [10].

#### Transformation from ECI to ECEF

The size of the angle $\Omega$ is dependent on the angular speed of the Earth $\omega_{Earth}$ and the time elapsed since the J2000.0 Epoch [10]:

$$\Omega = \omega_{Earth}(t - t_{J2000}) \tag{2.1}$$

The angular speed of the Earth is defined as [10]:

$$\omega_{Earth} = 7.292\,115\,167 \times 10^{-5}\mathrm{rad\,s^{-1}} \tag{2.2}$$

We can then construct a transformation matrix $\boldsymbol{T_I^E}$ for transformation from ECI to ECEF as a rotation $\boldsymbol{R_{z_E}}(\Omega)$ around the $z_E$ axis by angle $\Omega$:

$$\boldsymbol{T_I^E} = \boldsymbol{R_{z_E}}(\Omega) = \begin{bmatrix} \cos\Omega & \sin\Omega & 0 \\ -\sin\Omega & \cos\Omega & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

### 2.1.3 Local Geographic Frame

The Local Geographic Frame is an intuitive frame suitable for cases when the vehicle is on or near the Earth's surface, as it represents the Earth's surface as a flat plane tangent to a given point on Earth (see fig. 2.2. This assumption is adequate mostly in scenarios whose total extent is smaller than some tens of kilometers [17].

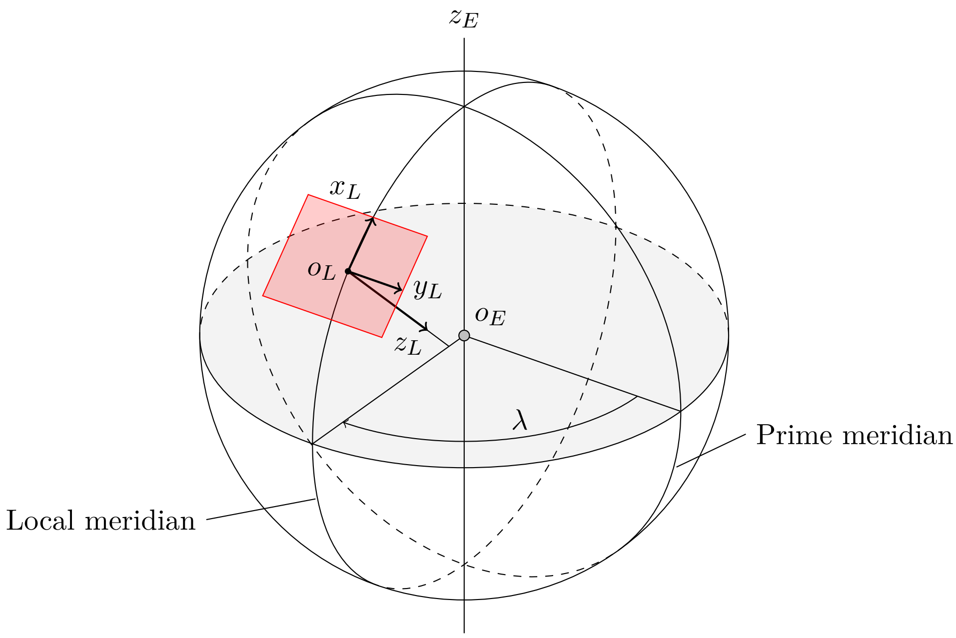



Figure 2.2: The Local Geographic Frame and North-East-Down coordinate system.

There are multiple ways to define axes in a local geographic frame. The most commonly used definition in aerospace is the North-East-Down (NED) coordinate system (depicted in fig. 2.2), defined as follows [17]:

- The origin $o_L$ is fixed to a given point on the surface of the Earth.

- The $x_L$ axis points towards the north, parallel to the local horizontal plane.
- The $y_L$ axis points towards the east, parallel to the local horizontal plane.
- The $z_L$ axis points vertically down.

### Transformation from ECEF to NED

Assume that $\boldsymbol{P_E}$ is a position in ECEF coordinate system and $\boldsymbol{P_r}$ is the position of the NED coordinate system origin ($o_L$). Then, the position of the point $\boldsymbol{P_E}$ can be transformed to NED coordinates $\boldsymbol{P_L}$ using the following equation [4]:

$$\vec{P_L} = \boldsymbol{T_E^L}\left(\vec{P_E} - \vec{P_r}\right) \tag{2.4}$$

The transformation matrix $\boldsymbol{T_E^L}$ is given by [4]:

$$\boldsymbol{T_E^L} = \begin{bmatrix} -\sin\Phi_r\cos\lambda_r & -\sin\Phi_r\sin\lambda_r & \cos\Phi_r \\ -\sin\lambda_r & \cos\lambda_r & 0 \\ -\cos\Phi_r\cos\lambda_r & -\cos\Phi_r\sin\lambda_r & -\sin\Phi_r \end{bmatrix} \tag{2.5}$$

## 2.1.4 Body Fixed Frame

To describe the orientation of a vehicle relative to its initial position, a sequence of rotations around the roll, pitch, and yaw axes is commonly used [10]. The angles of these rotations are called the *Euler angles*. It is crucial to preserve the order of rotations, as applying rotations in an incorrect order may result in a different vehicle orientation.

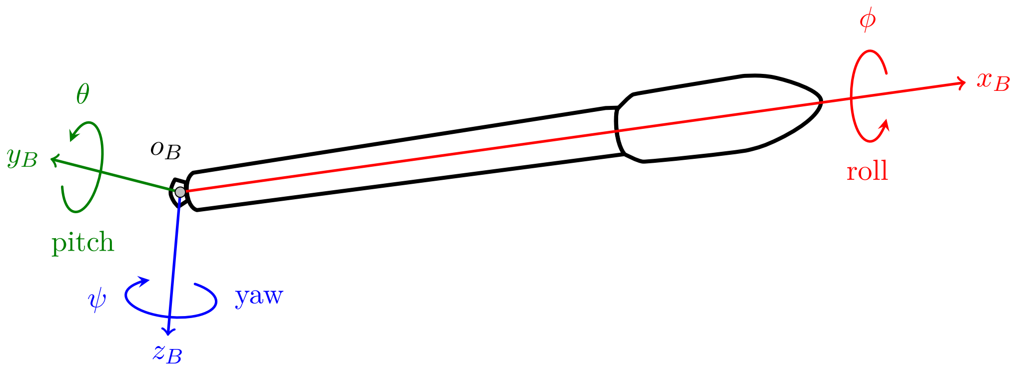



Figure 2.3: The Body Fixed Frame.

These axes also form the Body Fixed Frame (BFF) (see fig. 2.3) of the vehicle. This frame is also the frame of choice for specifying the positions of various internal components and other significant points inside the rocket.

The exact definition of the frame may be slightly different on some launch vehicles, but generally, the following definition applies [19, 30]:

- The origin $o_B$ lies at a fixed place along the longitudinal axis of the vehicle. For launch vehicles, the origin is usually situated below the gimbal plane of the stage.
- The $x_B$ axis is aligned with the longitudinal axis and points towards the nose.

- The $y_B$ axis usually points towards a certain significant feature on the surface of the rocket (such as the fin).

- The $z_B$ axis completes the right-handed coordinate system.

#### Transformation from NED to BFF

If we coincide the origins of the NED and BFF coordinate systems to same point $o = o_L = o_B$, then we can formulate the transformation matrix $T_L^B$ from NED to BFF as [4]:

$$\begin{aligned} \boldsymbol{T_L^B} &= \boldsymbol{R_1}(\phi)\,\boldsymbol{R_2}(\theta)\,\boldsymbol{R_3}(\psi) \\ &= \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\cos\theta \end{bmatrix} \end{aligned} \tag{2.6}$$

where $\phi, \theta, \psi$ denote the roll, pitch, and yaw angles of the vehicle, respectively.

## 2.2 Kinematic Differential Equations

The Kinematic Differential Equations deal with the time-dependent relationship between two reference frames [37]. In other words, these equations describe how the relative orientation of two frames changes over time. It is important to note that kinematics study the orientation of the body without involving any of the forces acting on the body. As such, the relations presented in this section are purely mathematical [37].

There are multiple ways to express the orientation of the body, which also means that there are different ways to express the kinematic differential equations. The most intuitive one is using the *Euler angles*. While it is a simple-to-understand method, it has one significant limitation, which can be resolved using a quaternion expression of the kinematic equation.

### 2.2.1 Euler Angles

Total angular velocity $\vec{\omega}$ can be expressed in terms of the basis vectors $\vec{i}, \vec{j}, \vec{k}$ of a Body Fixed Frame:

$$\vec{\omega} = p\vec{i} + q\vec{j} + r\vec{k} \tag{2.7}$$

where $p$, $q$, and $r$ are the components of the angular velocity [37]. Then, the time derivatives of Euler angles $\dot{\phi}, \dot{\theta}, \dot{\psi}$ (roll, pitch, and yaw rates) can be related to $p, q, r$ using a series of rotations [37]:

$$\vec{\omega} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + R_x(\phi) \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R_x(\phi) R_y(\theta) \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \tag{2.8}$$

Now, from eq. (2.8) we can obtain the final kinematic differential equation:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.9}$$

The inverse relationship can be obtained by inverting the matrix in eq. (2.9):

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{2.10}$$

As mentioned earlier, there is a limitation when using Euler angles—the eqs. (2.9) and (2.10) become singular when $\theta = \pi/2$ [37]. This inherent property of Euler angles may not be an issue in some scenarios but can be fully resolved only by using a different representation of kinematic equations.

### 2.2.2 Quaternions

In short, quaternions are an extension of complex numbers with numerous applications. They are particularly suitable for working with rotations in three-dimensional space, as they both solve the issue of singularities encountered when using the Euler angles and are also computationally more efficient [15].

A rotation around a basis vector $\vec{u}$ by angle $\theta$ can be performed by first constructing a rotation quaternion in the form of

$$q = q_0 + \boldsymbol{q} = \cos\frac{\theta}{2} + \vec{u}\sin\frac{\theta}{2} \tag{2.11}$$

Subsequently, a vector $\vec{v}$ can be rotated by the following operation:

$$q\vec{v}q^* \tag{2.12}$$

where $q^*$ is the conjugate of quaternion $q$ defined as $q^* = q_0 - q_1\vec{i} - q_2\vec{j} - q_3\vec{k}$ [15].

Consequently, quaternions can be used to express the kinematic equations, while gaining all benefits mentioned earlier. The following are the kinematic differential equations expressed with quaternions [31]:

$$\begin{bmatrix} \dot{q_0} \\ \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{2.13}$$

While quaternions are immensely useful during the calculations, they are not easy to interpret. Thus, it is often convenient to express the quaternion as Euler angles, which can be done using the following equations [25]:

$$\phi = \arctan\left(\frac{2\left(q_0q_1 + q_2q_3\right)}{q_0^2 - q_1^2 - q_2^2 + q_3^2}\right) \tag{2.14}$$

$$\theta = \arcsin\left(2\left(q_0q_2 - q_1q_3\right)\right) \tag{2.15}$$

$$\psi = \arctan\left(\frac{2\left(q_0q_3 + q_1q_2\right)}{q_0^2 + q_1^2 - q_2^2 - q_3^2}\right) \tag{2.16}$$

## 2.3 Equations of Motion

The equations of motion describe the translational and rotational behavior of the vehicle in time, depending on the forces and moments acting on the body of the vehicle [31]. The basis of these equations is the concepts and relationships presented earlier in this chapter.

In many cases, a single plane is sufficient to describe the flight path of a rocket booster. This means that the rocket needs only 3 Degrees of Freedom (DOF), which significantly simplifies the equations of motion. However, this work utilizes all 6 DOF in simulation to accurately describe any possible case. Furthermore, as the distances covered by the rocket booster during the launch are significant, the flat Earth assumption is not viable for this use. Hence, the vehicle's position is expressed using the ECEF in equations of motion.

The following equation describes the translational motion using the ECEF [31, 35]:

$$\vec{F_b} = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = m \left\{ \dot{\vec{V_b}} + \vec{\omega_b} \times \vec{V_b} + \boldsymbol{T_E^B} \cdot \vec{\omega_e} \times \vec{V_b} + \boldsymbol{T_E^B} \cdot \left[ \vec{\omega_e} \times \left( \vec{\omega_e} \times \vec{X_f} \right) \right] \right\} \tag{2.17}$$

where:

- $\vec{F_b}$ is given in the BFF,
- $m$ is the current mass of the vehicle,
- $\vec{V_b} = \begin{bmatrix} u & v & w \end{bmatrix}^T$ is the velocity vector of the vehicle in the BFF,
- $\vec{\omega_b}$ are the angular rates with respect to ECI given in BFF,
- $\vec{\omega_e}$ is the Earth rotation rate,
- and $\boldsymbol{T_E^B}$ is the transformation matrix from the ECEF axes to BFF axes.

Similarly, the rotational behavior of the rocket can be described by the following equation [31, 35]:

$$\vec{M_b} = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = \vec{I} \dot{\vec{\omega_b}} + \vec{\omega_b} \times \left( \vec{I} \vec{\omega_b} \right) + \dot{I} \vec{\omega_b} \tag{2.18}$$

where the moments $\vec{M_b}$ given in the BFF are related to angular rates $\omega_b$, and the inertia tensor $I$.

# Chapter 3

# Aerodynamic Characteristics of a Reusable Rocket Booster

The booster of a reusable rocket spends most of its time in denser layers of the atmosphere, where the effects of aerodynamic forces cannot be neglected. More importantly, the aerodynamic effects on the booster during its descent are significant and must be considered. Furthermore, some of the reusable boosters even use aerodynamic forces to steer during the descent. [7].

The beginning of this chapter will focus on the general description of aerodynamic forces and moments and their theory. Then, the process of determination of aerodynamic coefficients for the reusable rocket booster will be discussed.

## 3.1 Aerodynamic Forces and Moments

Among the major forces that act on the rocket during its flight in the atmosphere are the aerodynamic forces. They are mechanical forces generated by the relative motion of a rocket in the air. As such, these forces and moments depend on the orientation of the rocket relative to the airflow [31].

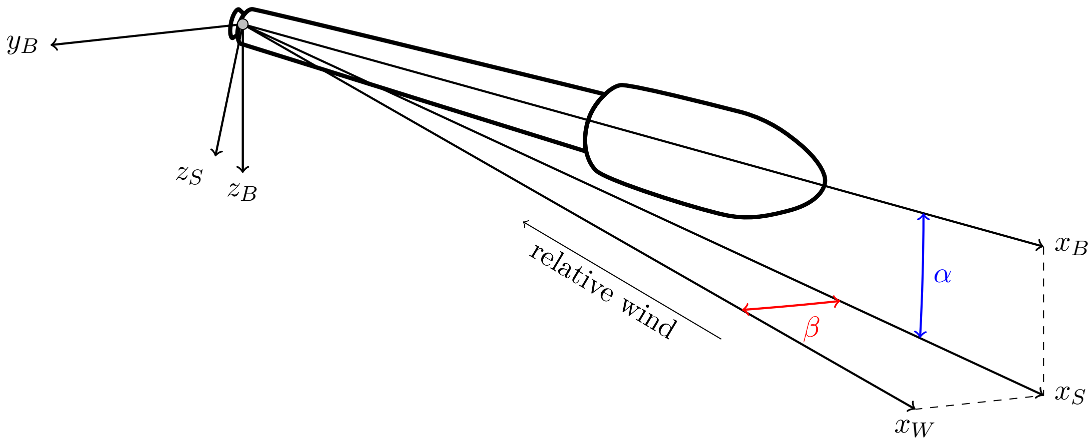



Figure 3.1: The aerodynamic angles $\alpha$ and $\beta$ and their relation to Body Fixed Frame

To determine the aerodynamic forces and moments acting on the vehicle, it is necessary to express its orientation relative to the airflow. This is done by introducing two new coordinate systems; both originated at the same point as the BFF.

The first one is the *stability axes* coordinate system. It is obtained from the BFF by a single rotation around the $y_B$ axis through angle $\alpha$, also known as angle of attack (AoA) [31]. The second coordinate system is the *wind-axes* system, which is obtained from the stability axes by a rotation around the $z_S$ axis. The size of this rotation is the angle of sideslip (AoS), denoted as $\beta$ [31].

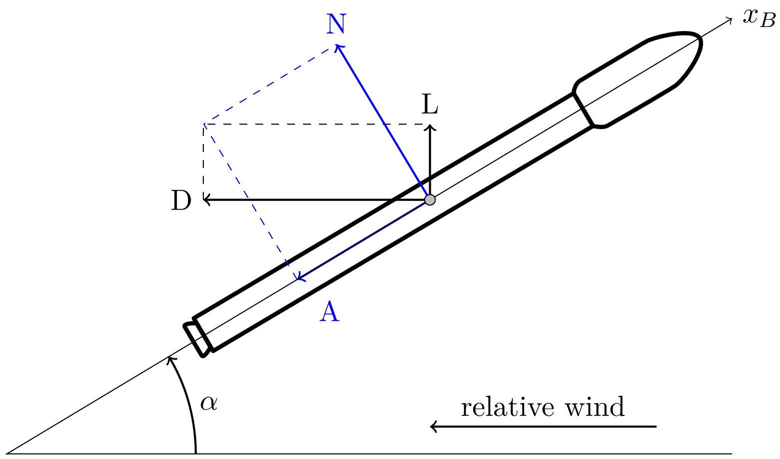



Figure 3.2: The aerodynamic forces acting on the rocket

The aerodynamic forces acting on the rocket can be decomposed into multiple components. There are several ways to perform this decomposition, which differ in the coordinate system in which the forces are decomposed, as can be seen in [28].

The most common ways of decomposition are shown in fig. 3.2. Historically, the preferred way was to express the forces in the wind-axes coordinate system as the lift force $L$ and drag force $D$ [28].

However, for symmetric bodies such as rockets, it is often beneficial to use the BFF to decompose the aerodynamic forces [28]. In this case, the components are the normal force $N$ and axial force $A$.

As the relationship between these decompositions is purely geometrical, the components can be transformed using the following equations [31]:

$$N = L\cos(\alpha) + D\sin(\alpha) \tag{3.1}$$

$$A = -Lsin(\alpha) + D\cos(\alpha) \tag{3.2}$$

The inverse relation also applies:

$$L = N\cos(\alpha) - A\sin(\alpha) \tag{3.3}$$

$$D = N\sin(\alpha) + A\cos(\alpha) \tag{3.4}$$

The magnitude of the aerodynamic forces depends on the aerodynamic coefficients, which in turn depend on a large number of other variables, most significantly on the aerodynamic angles $\alpha$ and $\beta$, the Mach number, altitude, and others. Last but not least, these coefficients depend on the shape of the rocket [31].

As with the forces, the aerodynamic coefficients can also be expressed in multiple coordinate systems. Equations for the transformations of these coefficients are analogous to eqs. (3.1) to (3.4). The relation between various choices of input axes and the corresponding aerodynamic coefficients can be seen in table 3.1.

Table 3.1: Aerodynamic coefficients used for various choices of axes [35].

| **Used axes** | **Force coefficients** | **Moment coefficients** |
|---|---|---|
| Body | axial $C_A$, sideforce $C_S$, normal $C_N$ | roll $C_\ell$, pitch $C_m$, yaw $C_n$ |
| Stability | drag $C_D$, sideforce $C_S$, lift $C_L$ | roll $C_\ell$, pitch $C_m$, yaw $C_n$ |
| Wind | drag $C_D$, cross-wind $C_C$, lift $C_L$ | roll $C_\ell$, pitch $C_m$, yaw $C_n$ |

Assuming the coefficients are in the BFF axes, then, the aerodynamic forces $A, S, N$ and moments $\ell, m, n$ are expressed by the equations [31]:

$$A = \vec{q}SC_A \tag{3.5}$$

$$S = \vec{q}SC_S \tag{3.6}$$

$$N = \vec{q}SC_N \tag{3.7}$$

$$\ell = \vec{q}SbC_\ell \tag{3.8}$$

$$m = \vec{q}ScC_m \tag{3.9}$$

$$n = \vec{q}SbC_n \tag{3.10}$$

where $S, b, c$ are the reference area, length, and span, respectively. The quantity $\vec{q}$ is the dynamic pressure, which is effectively a product of the density $\rho$ and square of the velocity $V$:

$$\vec{q} = \frac{1}{2}\rho V^2 \tag{3.11}$$

## 3.2 Determination of Aerodynamic Coefficients for Reusable Rocket Booster

As was shown in the previous section, the calculation of forces and moments acting on the vehicle is relatively trivial if the aerodynamic coefficients are known. Unfortunately, the values of those coefficients are highly dependent on the shape of the vehicle, aerodynamic angles, and Mach number. It is apparent that the determination of the aerodynamic coefficients is a complex task and necessitates the application of special techniques.

While the topic of the determination of aerodynamic coefficients is far out of the scope of this work, for the sake of completeness, a brief description of several methods will be presented in this section.

### 3.2.1 Experimental Methods

The traditional method of obtaining flight coefficients is by performing an experiment in a wind tunnel using a scaled-down model of the vehicle. During the test, the model is mounted on a rigid test fixture (commonly known as a „sting“), and the air is made to flow past the model. Using various probes and sensors, many different parameters can be measured [31].

The obvious disadvantage of this method is the time complexity and cost associated with building the scale model of the vehicle and setting up the whole experiment. Therefore, it is not viable for use in this work.

### 3.2.2 Computational Fluid Dynamics

Thanks to the advancements in computing power in recent years, it has become increasingly more viable (and in some cases preferable) to use computer simulations to determine the aerodynamic characteristics of a vehicle [14]. This approach is commonly known as Computational Fluid Dynamics.

In CFD, a set of nonlinear equations known as *the Navier-Stokes equations* is numerically solved. Based on Newton's second law of motion and energy conservation, these equations describe the dynamics of a fluid particle in terms of its mass, momentum, and energy [14].

In order to solve these equations on a computer, it is necessary to truncate the computational domain using boundary conditions and discretize it with methods such as Finite-Difference Method (FDM) or Finite-Volume Method (FVM).

## 3.3 Aerodynamic Model of the Booster

As the aerodynamic characteristics of the chosen reusable booster (or any other reusable booster for that matter) are not publicly disclosed, nor any aerodynamic models are available, it was necessary to create a custom aerodynamic model. Fortunately, while concrete aerodynamic data are not available, at least the behavior of a reusable rocket booster can be found in various works such as [16, 18]. For this, it was needed to perform an aerodynamic study of the rocket using CFD.

After experimentation with various CFD solvers, the student edition of *Ansys Fluent* was chosen for its ability to handle supersonic flows. This is important for this application, as the booster reaches supersonic speeds both during ascent and descent.

Naturally, the CFD solver needs a CAD model of the vehicle of sufficient quality to perform the calculations. Unfortunately, only mesh models of the selected booster are freely available, which are not suitable for this use. Therefore, it was also needed to create the CAD model itself.

The final model, shown in fig. 3.3, was created using *Autodesk Fusion 360* according to the specifications of the booster used in this work (see section 6.1) and available visual reference [30]. The model is intentionally simplified, as its purpose is to obtain only the general aerodynamic characteristics. Furthermore, a simpler, symmetric model allows certain assumptions, which significantly speed up the solver's computation.

This model was then used to create and set up the simulation domain for the calculation in *Ansys Fluent*. In this case, this was done by creating an enclosure with the size of $375 \times 204 \times 204\,\mathrm{m}$. As the model of the rocket booster is symmetrical, it was possible to reduce the size of the enclosure by half by creating a symmetry face in the $xy$ plane, as can be seen in fig. 3.4. Also, note the smaller green area, where a finer mesh will be generated to accommodate the wake region.

The mesh was generated using the built-in *Fluent Mesher* using the watertight geometry workflow. Some of the default values were tuned in order to generate a valid mesh suitable for this purpose. Also, the boundary conditions were assigned in this step (shown in fig. 3.5).

After successfully generating the mesh, it was possible to start with the experiments. As mentioned earlier, while a finished aerodynamic model or concrete measurements are

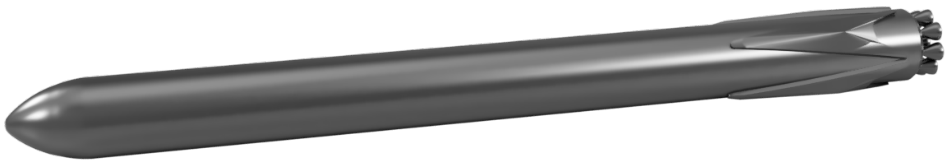


Figure 3.3: The CAD model of the booster used for aerodynamic studies.

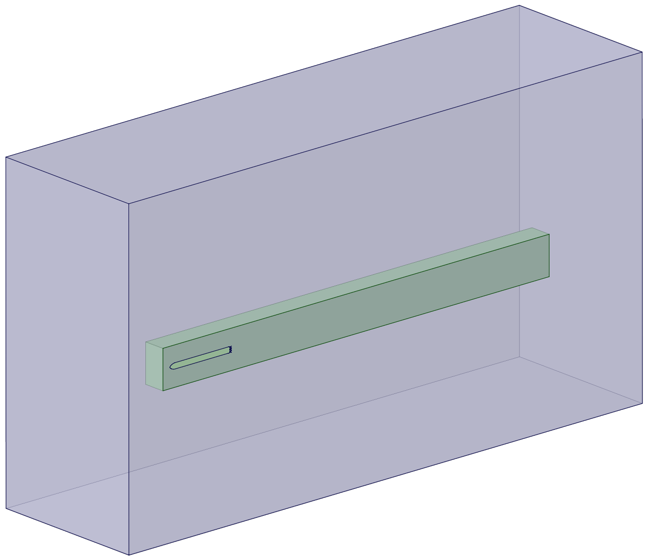


Figure 3.4: CFD simulation domain consisting of enclosure (blue) and wake region (green).

hard to obtain, a general aerodynamic characteristic of a reusable rocket booster can be obtained. This, together with the fact that the CFD simulations are highly computationally intensive and time-consuming, led to the decision to reduce the number of simulations and create the aerodynamic model with the help of publicly available data.

For example, the dependency of the aerodynamic coefficients (mostly the axial coefficient $C_A$) on the Mach number was determined by repeatedly running the simulation with AoA $\alpha = 0$ and different inlet velocities. The results, presented in table 3.2, are consistent with the expected behavior of the drag coefficient described in literature [31]. Most notably, the transonic drag rise caused by the presence of shockwaves can be distinguished in fig. 3.6. The formation of shockwaves itself can also be seen in fig. 3.7 near the nose cone and the tips of the landing legs.

The data obtained from the simulation together with information available from the *CALLISTO* project [18] form the basis for determining the axial force coefficient $C_A$ at AoA $\alpha = 0$. The normal, sideforce, and moment coefficients are neglected at $\alpha = 0$, as their values are close to zero at all velocities.

While the AoA and AoS do not change significantly during the ascent, they should be taken into account in the aerodynamic model. This is achieved by adding components

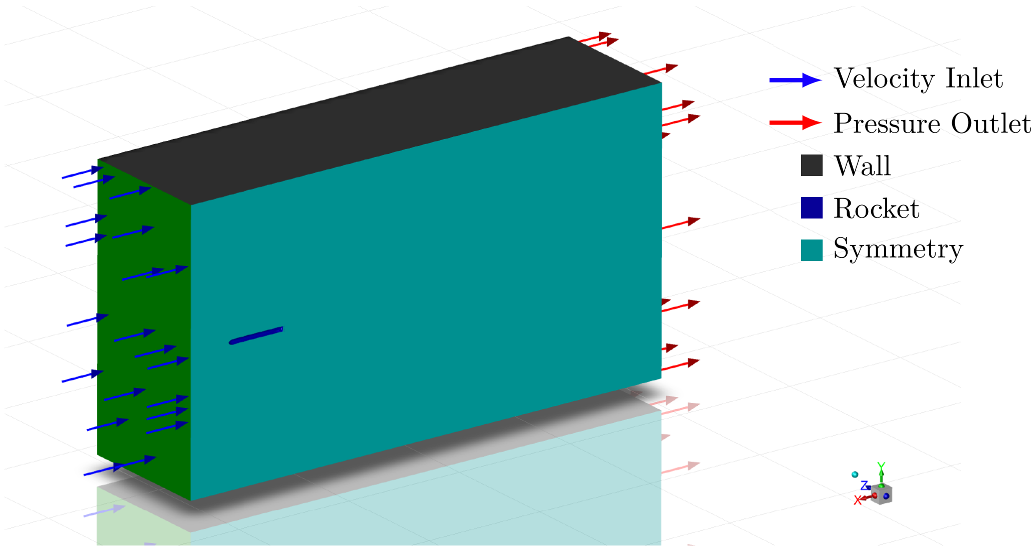



Figure 3.5: Assignment of the boundary conditions.

Table 3.2: Aerodynamic coefficients of the rocket booster determined at $\alpha = 0$ and specified inlet velocities. Coefficients $C_S$ and $C_m$ are omitted, as $C_S = C_N$ and $C_m = C_n$ due to symmetry.

| | | Force Coefficients | | Moment Coefficients | |
|---|---|---|---|---|---|
| **Velocity** | **Mach speed** | $C_A$ | $C_N$ | $C_\ell$ | $C_n$ |
| 100 m s$^{-1}$ | 0.303 | 0.170 99 | −0.000 502 | 0.000 643 | −0.003 005 |
| 200 m s$^{-1}$ | 0.606 | 0.175 23 | −0.000 031 | 0.000 622 | −0.000 253 |
| 300 m s$^{-1}$ | 0.909 | 0.234 78 | −0.001 220 | 0.000 998 | −0.003 025 |
| 450 m s$^{-1}$ | 1.364 | 0.394 24 | 0.001 108 | 0.000 631 | −0.004 850 |
| 500 m s$^{-1}$ | 1.515 | 0.403 79 | −0.000 573 | 0.000 479 | −0.007 522 |
| 600 m s$^{-1}$ | 1.818 | 0.393 38 | −0.000 630 | 0.000 224 | −0.008 293 |
| 700 m s$^{-1}$ | 2.121 | 0.393 38 | −0.000 630 | 0.000 224 | −0.008 293 |

linearly dependent on $\alpha$ and $\beta$ to the original value of the axial force coefficient $C_{A_0}$:

$$C_A = C_{A_0} + \alpha C_{A_\alpha} + \beta C_{A_\beta} \tag{3.12}$$

$$C_S = \beta C_{S_\beta} \tag{3.13}$$

$$C_N = \alpha C_{N_\alpha} \tag{3.14}$$

Similarly, the moment coefficients are related to both aerodynamic angles and to rotation rates $p, q, r$:

$$C_\ell = p C_{\ell_p} \tag{3.15}$$

$$C_m = q C_{m_q} + \alpha C_{m_\alpha} \tag{3.16}$$

$$C_n = r C_{n_r} + \beta C_{n_\beta} \tag{3.17}$$

It is important to note that this approximation is sufficient only for low values of $\alpha$ and $\beta$. This is not an issue, as they are kept to a minimum during ascent. The values of $\alpha$ and

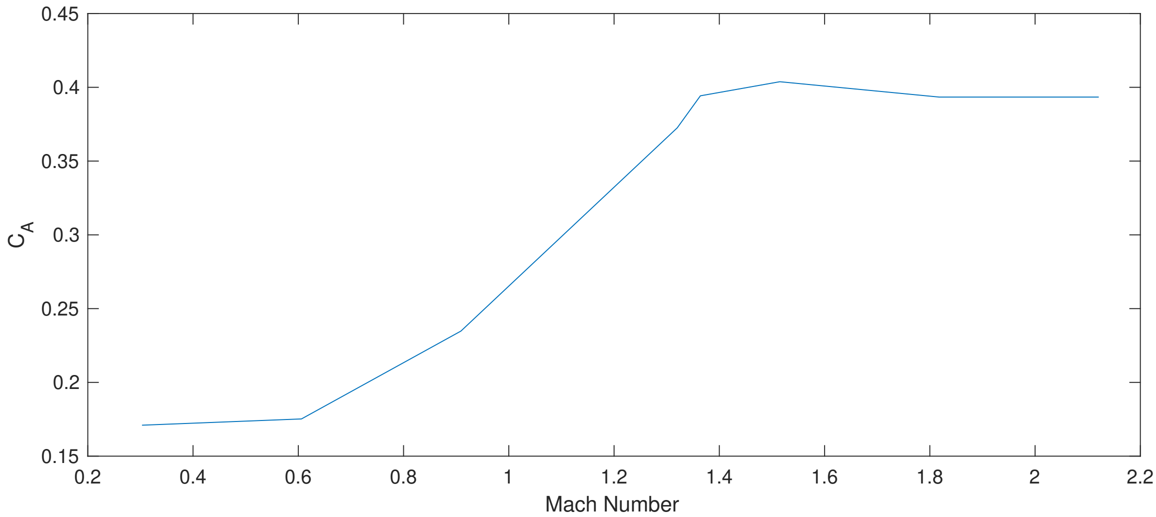



Figure 3.6: Dependency of axial coefficient on the Mach number.

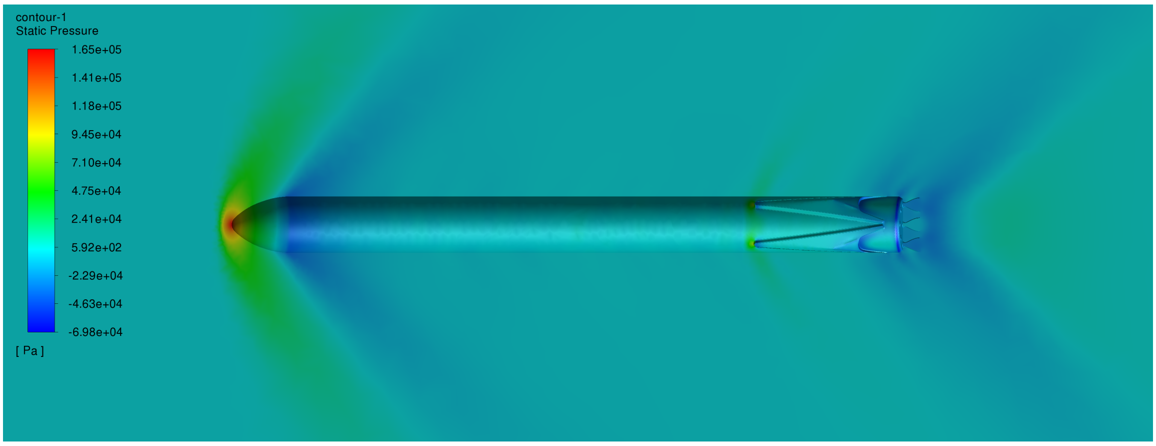



Figure 3.7: Countour plot of the static pressure at inlet velocity of $450\,\mathrm{m\,s^{-1}}$.

$\beta$ change significantly during the turnaround (which orients the vehicle for descent), but as it happens far above the atmosphere, it produces no aerodynamic effects.

Thus, the only case which needs to be taken care of is the descent through the atmosphere during which the vehicle has $\alpha = 180\,°$. This is resolved by simply flipping the direction of the $x_B$ axis for purposes of the aerodynamic model when the $\alpha$ is larger than 90°:

$$\alpha_{\text{Aero}} = \left\{ \begin{array}{ll} 180\,° - \alpha & \text{for } \alpha > 90\,° \\ -180\,° - \alpha & \text{for } \alpha < -90\,° \\ \alpha & \text{otherwise} \end{array} \right\} \tag{3.18}$$

and analogically for $\beta$.

# Chapter 4

# Design of Spacecraft Guidance, Control, and Navigation Systems

The GNC subsystem is at the heart of every rocket booster. In a nutshell, its task is to successfully achieve mission goals by following the flight trajectory that leads to the target orbit and respond to disturbances from the boosters' environment. Nowadays, with the rise of reusable launch systems, its purpose is extended with the secondary task of safely landing the rocket booster back to the surface of the Earth.

The acronym GNC is an all-encompassing term covering a broad range of subsystems [23]. The **Guidance** part refers to systems responsible for establishing the trajectory the booster needs to follow. The purpose of **Navigation** is to find the vehicle's current position and predict a future state. Finally, **Control** refers to the determination of actions necessary to match the current state (obtained by navigation) with the desired path (guidance). The following sections will focus on all three aspects of the GNC subsystem, their components, and the algorithms they use.

## 4.1 Guidance

The task of the guidance is to calculate the path the rocket needs to follow to reach a specific target. In expendable rockets, it is only necessary to determine the trajectory for reaching the target orbit. For the sake of completeness, a brief description of the launch trajectory is included in this section.

However, a reusable rocket booster also needs guidance to the landing site on the surface of the Earth. This task is much more challenging than it appears, primarily due to the extreme conditions the vehicle goes through during landing and very tight margins for error. An algorithm for the calculation of the optimal landing path will be described later in this section.

### 4.1.1 Launch Trajectory

Determination of the launch trajectory is a complex optimization problem. In order to find an optimal solution, it is necessary to consider a large number of factors and constraints, determined by both the mission requirements (such as the target orbit) and the design and construction of the launch vehicle [24]. Despite the complexity of this task, all vertically launched launch vehicles (LVs) follow a common pattern during their ascent, which can be seen in fig. 4.1.

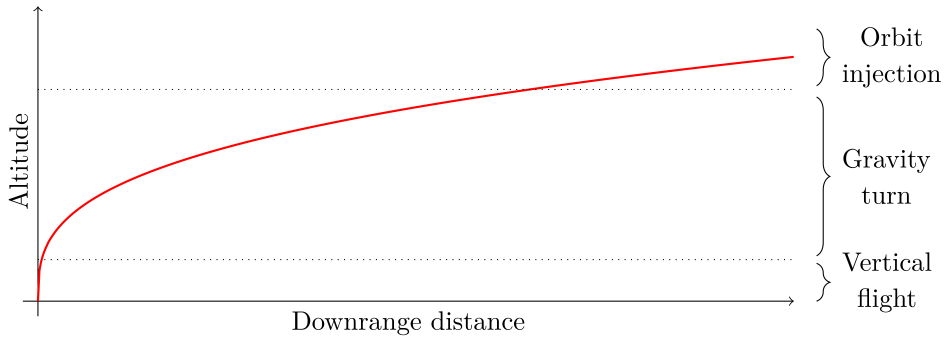



Figure 4.1: Launch trajectory of a vertically launched launch vehicle.

The LV is accelerated vertically from the launch site and remains in vertical (or near-vertical flight) for the first roughly 15 s, during which the vehicle ascends through the denser parts of the atmosphere [24].

Due to the aerodynamic heating considerations, any turn maneuvers are usually performed when the vehicle leaves the denser atmosphere [37]. Then, the vehicle initiates a pitch maneuver, which puts it into a *gravity-turn* trajectory.

In the *gravity-turn* trajectory, the thrust vector is kept in the direction opposite to the velocity vector, and the AoA is close to zero. This lowers the aerodynamic stress, making it possible to minimize the structural mass of the vehicle [37].

During the ascent, the vehicle reaches a point called *Max Q*, when it is subjected to maximum dynamic pressure [24]. Most LVs temporarily throttle down during this phase to reduce the aerodynamic loads produced on the vehicle [33].

The last step of the launch is the final guidance using the upper stage engines into the target orbit, which occurs after the vehicle leaves the atmosphere [24].

### 4.1.2 Booster Landing Trajectory Optimization

Landing a spacecraft, in general, is a challenging task. Landing a VTVL rocket booster on the surface of the Earth is even more difficult due to several reasons. First, Earth's denser atmosphere causes large drag forces during descent, resulting in extreme heating. High-altitude winds cause significant disturbances to the trajectory, which, combined with the need for precision and a small margin for error, make precision landing on Earth very difficult [3].

To perform such a challenging task, several maneuvers and mechanisms are required. This is shown in fig. 4.2, which presents a simplified scheme of the phases of flight and landing of such reusable booster [26]. However, a guidance algorithm for the calculation of the landing trajectory is also required.

While the exact specifics and implementation of the algorithms used in the actual reusable rocket boosters are not public, for example, the *SpaceX Falcon 9* is known to use an algorithm based on lossless convexification [2].

In this method, the planetary soft landing is first formulated as an optimal control problem with nonconvex control constraints, such as the limited throttling range of the landing engine. The aim is to find an optimal thrust profile and the corresponding translational state trajectory, which guides the booster to a predefined landing location [2].

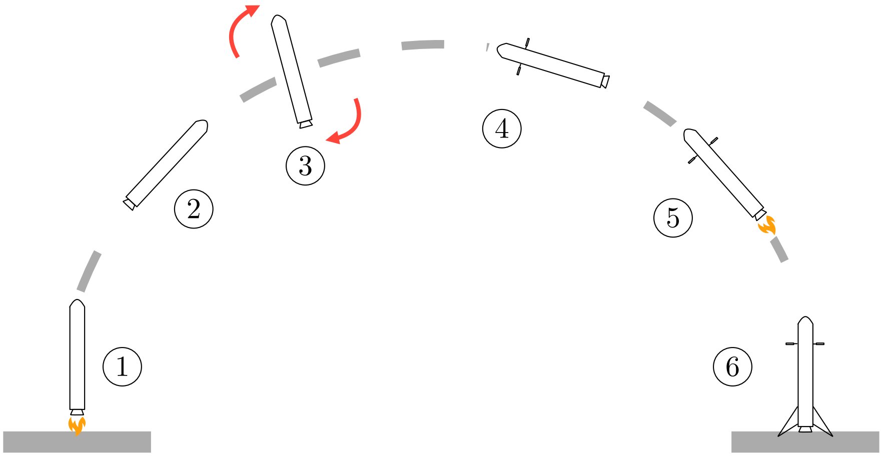



Figure 4.2: Landing of a reusable rocket booster. The booster starts the flight as usual with the liftoff (1) and provides thrust until the Main Engine Cut Off (MECO) (2). Shortly after that, the first stage rotates with RCS thrusters to position itself for descent (3) and the grid fins are deployed (4). When the booster enters the atmosphere, it commences the entry burn, which slows it down to prevent aerodynamic damage (5). Shortly before touchdown, the landing legs deploy, and the booster lands propulsively on Earth (6).

The paper presents a method of lossless convexification of this problem, which makes it possible to solve it using the interior point methods [2]. This can also be done in real-time, making it possible to recalculate the landing trajectory on-fly [3].

## 4.2 Navigation

The main purpose of navigation is to determine the vehicle's position and orientation in space using a set of sensors. In the case of modern LVs, the sensor set usually consists of a Global Positioning System (GPS) receiver and an Inertial Measurement Unit (IMU) with accelerometers and gyroscopes [22].

### 4.2.1 Accelerometers

An accelerometer is an inertial sensor that indirectly measures the specific force needed to prevent a known proof mass from moving when its case is being accelerated [31]. While the number of different types of accelerometers is large, the basic principle is the same.

The accelerometer contains a proof mass of known weight that is free to move in the accelerometers' sensitive axis. When an accelerating force acts on the accelerometer case in the sensitive axis, the proof mass will not immediately change its velocity. This causes a displacement of the proof mass relative to the case, measured by the pick-off [11].

### 4.2.2 Gyroscopes

A gyroscope is an inertial sensor that measures the angular rate around an axis. Unlike accelerometers, there are three main types of gyroscopes, each of which uses a different basic principle [11].

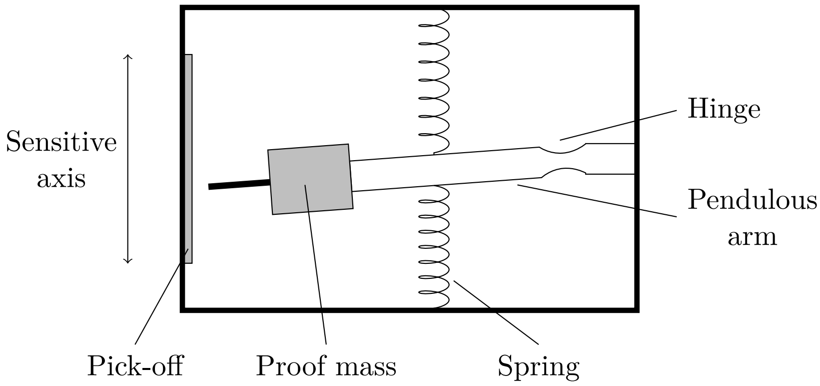



Figure 4.3: Simple mechanical pendulous accelerometer.

Originally, most gyroscopes were spinning-mass gyros. The basic principle of these gyroscopes employs the conservation of angular momentum. Due to their high complexity, power consumption, and other disadvantages stemming from their mechanical nature, they have been superseded by optical and vibratory gyros [11].

Optical gyroscopes are commonly used in aerospace applications [11, 22]. There are two main kinds: the Ring Laser Gyroscope (RLG) and the Interferometric Fiber-Optic Gyro (IFOG). Both of these employ the same core principle—the *Sagnac effect*.

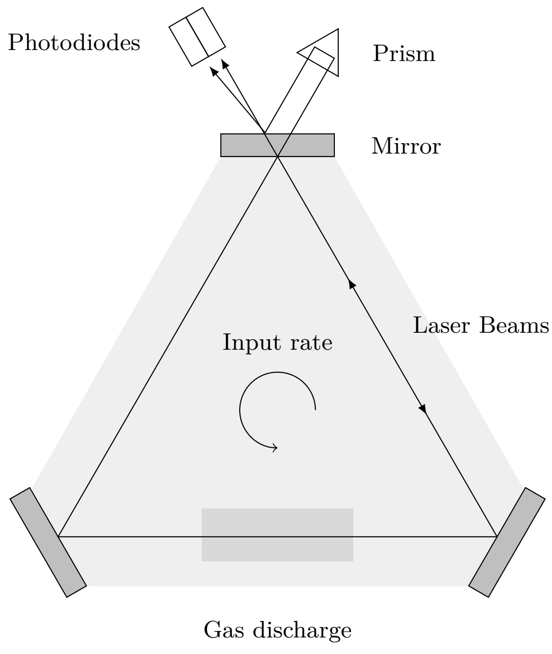



Figure 4.4: The Ring Laser Gyroscope

The Ring Laser Gyroscope (RLG), shown in fig. 4.4, is formed by a (usually) triangle-shaped laser cavity with mirrors in each of the corners. There are two laser beams in the cavity—one in each direction. If the gyroscope is stationary, both beams have the same wavelength. However, if the gyro rotates around its sensitive axis, the path of the beam in the direction of rotation is longer, which results in an increase of the wavelength. The opposite happens for the beam in the other direction [11].

Due to the scattering inside the cavity, the wavelengths of the two beams do not diverge at low angular rates. This issue is resolved in most RLGs by the dither wheel, which applies low-amplitude, high-frequency vibrations around the sensitive axis [11].

The second type of optical gyroscope, the Interferometric Fiber-Optic Gyro (IFOG), was initially developed as a lower-cost solution. However, nowadays, its performance is on par with the RLG, and its reliability is higher than both RLG and spinning-mass gyro [11].

### 4.2.3 Global Navigation Satellite Systems

The accuracy of accelerometers and gyroscopes decreases over time during their use. Fortunately, the launch only takes a few minutes, during which the accumulated error on the sensors is often still acceptable [6]. However, today's missions often require better accuracy, which cannot be reached by IMU alone [22]. Most importantly, such high accuracy is necessary for the vertical landing of the rocket booster [3]. To achieve the required accuracy, most LVs utilize some kind of Global Navigation Satellite System (GNSS). While there are currently multiple competing satellite navigation systems, the basic principle, which will be described in this section, is the same.

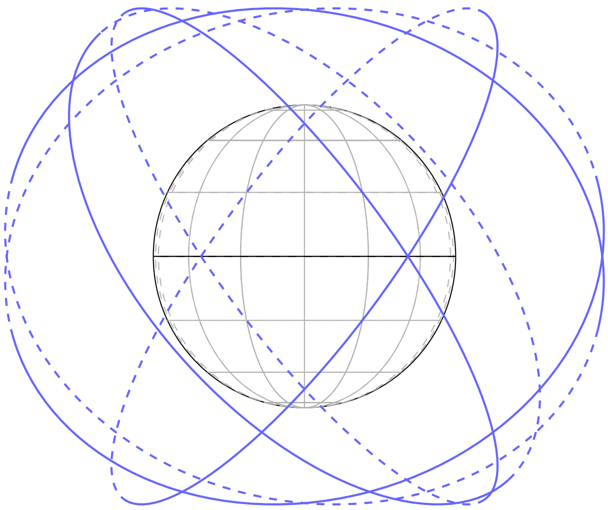


Figure 4.5: Orbital planes of the Global Positioning System (GPS).

A GNSS consists of an extensive network of satellites, known as a *constellation*, positioned in several orbital planes in a way that ensures the visibility of 5–14 satellites from most places on the Earth at most times if there is a clear line of sight [11]. An example of such satellite constellation is shown in fig. 4.5.

The satellites broadcast multiple signals with various data. Among them, timing messages and information about satellite orbits, which make it possible to calculate the current position of a satellite [11].

As a result, the position of a GNSS receiver can be determined by performing a range measurement from the satellite signal by constructing a sphere with radius $r$ (determined from the signal strength) with center at the satellite's position. The receiver then may lie anywhere in the walls of this sphere. By adding a measurement from a second satellite, the area of possible receiver positions is reduced to the intersection of the two spheres—a circle. Third satellite reduces this further to only two points on the circle. This ambiguity can be resolved by adding further measurements, or in some cases, by simply eliminating the non-viable point (*e.g.*, when it lies inside the Earth) [11].

## 4.3 Control

This section aims to describe all of the various methods that the rocket booster can use to change and maintain its orientation and position. This is necessary not only for achieving the correct trajectory that results in the desired orbit, but also for the landing phase of a reusable booster. The last part of this section is then dedicated to the control algorithms used to drive the mechanical actuators of the rocket's controls.

### 4.3.1 Propulsion

Even though the primary use of the rocket engine is not control, many modern rocket engines are equipped with a Thrust Vector Control (TVC) system, which adds the ability to manipulate the direction of the engine thrust. For this reason, the description of a rocket engine and the TVC is included in this chapter.

A rocket engine produces thrust by ejecting high-temperature propellant gas at high speeds. Per Newton's third law of motion, this causes the motion of a rocket in the direction opposite to the direction of the ejected gas [13].

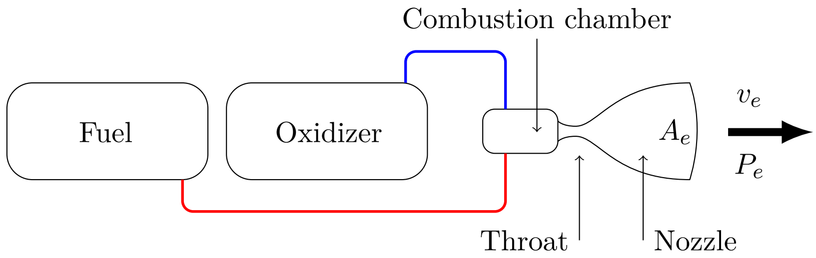



Figure 4.6: Simplified scheme of a liquid bi-propellant engine.

In this work, a liquid bi-propellant rocket engine will be considered, as it is the most common type of rocket engine used for the first stage. As can be seen in fig. 4.6, in these engines, liquid fuel and oxidizer combine in the combustion chamber.

The hot gas then leaves the combustion chamber and passes through the converging portion of the nozzle, where it is accelerated to sonic speed. To further increase the velocity of the gas, it must be expanded in the diverging part of the nozzle [13].

The rocket engine's thrust depends on the difference between the exhaust gas pressure $P_e$ and the ambient pressure $P_0$ around the rocket engine–for ideal performance, the pressures should be equal. The exhaust gas pressure $P_e$ is affected by the cross-sectional area of the exit plane $A_e$.

These characteristics of a rocket engine can be modeled using the *Thrust equation*:

$$F_t = \dot{m}v_e + A_e(P_e - P_0) \tag{4.1}$$

where $F_t$ is the thrust force of a rocket engine, $\dot{m}$ is the mass flow rate, and $v_e$ is the effective exhaust velocity at the exit plane when $P_e = P_0$.

To control the direction of the thrust, several different methods of TVC exist. For rocket engines, the most common solution is to gimbal the nozzle of an engine. This can be achieved, for example, by fixing the top of the engine to the vehicle through a spherical joint. The direction of the engine is then controlled using two actuators, which are mounted parallel to the pitch and yaw axis of the vehicle [13]. A simplified scheme of such a method of attachment can be seen in fig. 4.7.

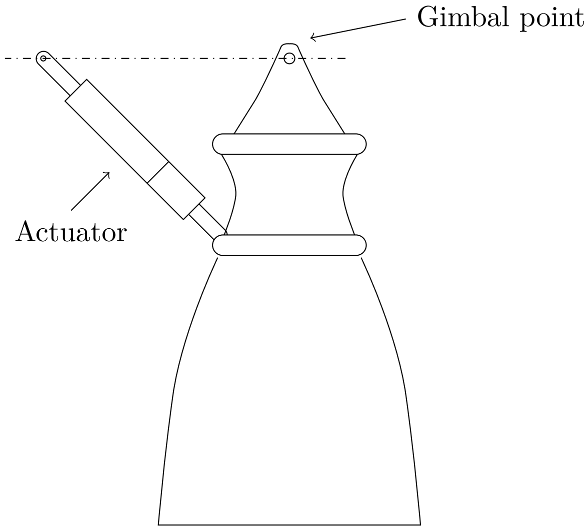



Figure 4.7: Gimbal mechanism of a rocket engine.

When the engine gimbals, the direction of the thrust changes in the same way. This can be expressed mathematically by rotating the thrust vector (originally coincident with the vehicle $-x$ axis) by angles $\delta_\theta$ and $\delta_\psi$, which represent the gimbal deflection in the pitch and yaw axes. This can be achieved by constructing a rotation matrix in a similar way as in section 2.1:

$$\vec{F}_t = \begin{bmatrix} F_{t_x} \\ F_{t_y} \\ F_{t_z} \end{bmatrix} = \begin{bmatrix} \cos\delta_\theta \cos\delta_\psi & \sin\delta_\psi & -\sin\delta_\theta \cos\delta_\psi \\ -\cos\delta_\theta \sin\delta_\psi & \cos\delta_\psi & \sin\delta_\theta \sin\delta_\psi \\ \sin\delta_\theta & 0 & \cos\delta_\theta \end{bmatrix} \cdot \begin{bmatrix} F_t \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} F_t \cos\delta_\psi \cos\delta_\theta \\ -F_t \cos\delta_\theta \sin\delta_\psi \\ F_t \sin\delta_\theta \end{bmatrix} \tag{4.2}$$

The eq. (4.2) transforms the gimbaled thrust force into the BFF, which is then added to other forces acting on the vehicle. However, gimbaling the engine also produces a rotational moment about the vehicles center of gravity (CG), which also needs to be considered. Considering the forces are applied at the point $\vec{X_{cp}}$ and the center of gravity (CG) is located at point $\vec{X_{cg}}$ (both positions specified in the BFF), the resulting moments can be calculated as [35]:

$$\vec{M}_t = \begin{bmatrix} M_{t_x} \\ M_{t_y} \\ M_{t_z} \end{bmatrix} = \vec{F}_t \times \left(\vec{X_{cg}} - \vec{X_{cp}}\right) \tag{4.3}$$

The gimbal mechanism can only move the engine in the vehicle $y$ and $z$ axes, which means that only the pitch and yaw can be controlled with a single gimbaled engine. However, suppose the stage has two or more gimbaled engines. In that case, it is possible to generate torque about the $x$ axis by differentially gimbaling the engines as shown in fig. 4.8 [13].

### 4.3.2 Reaction Control System

The purpose of the Reaction Control System (RCS) is to provide attitude and stability control outside the atmosphere when other means of control (such as aerodynamic) are

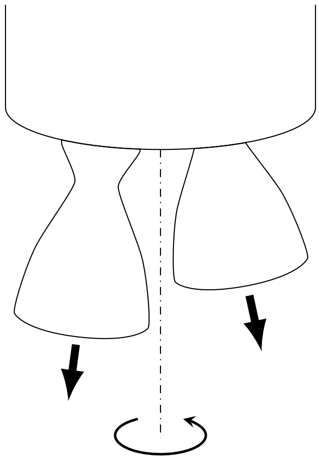


Figure 4.8: Roll control using two gimbaled engines.

not available [29]. The RCS thrusters are essentially small rocket engines positioned in such a way that firing them creates a rotational or translational[1] movement of the vehicle. Commonly, the thrusters are arranged in a common assembly (or „block“) with separate nozzles, as can be seen in fig. 4.9.

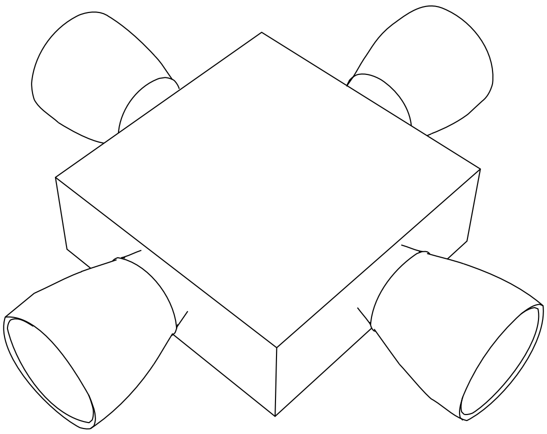


Figure 4.9: RCS thruster assembly.

It is apparent that an RCS thruster needs to be started and stopped many times during the mission, making solid rocket engines completely unsuitable for this purpose. While liquid rocket engines do not have this limitation, their reliability in such scenarios is of concern. The best choices for this objective are hypergolic rocket engines and cold gas thrusters [29].

As the RCS thrusters are rocket engines, the same principles and equations that apply to rocket engines, apply to thrusters. In other words, the thrust of an RCS thruster can be calculated by the *Thrust equation* (eq. (4.1)), and the moments about the CG can be calculated by eq. (4.3).

[1] Translational movement of the vehicle is required only on vehicles that need to dock in space.

### 4.3.3 PID Controller

The PID controller is one of the most well-known and widely used control mechanisms used in most industrial applications. Despite the availability of many more sophisticated algorithms nowadays, PID remains in use thanks to its simplicity in design and implementation [36].

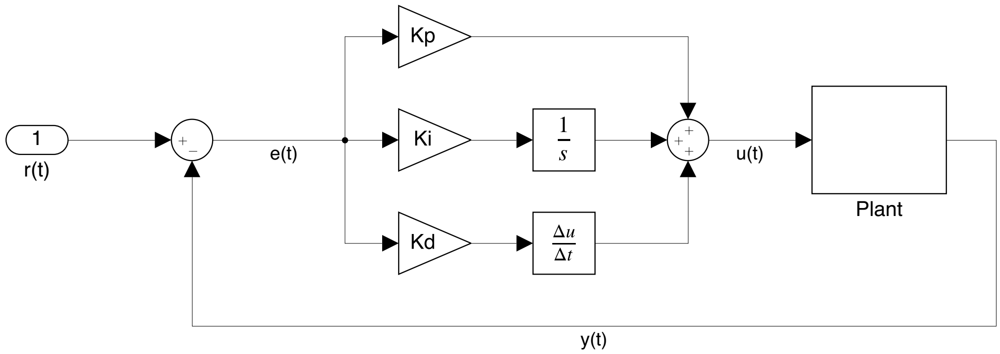



Figure 4.10: PID controller block scheme.

As the name suggests, the PID controller consists of proportional, integral, and derivative terms. The controller takes the difference between the actual output of the controlled plant $y(t)$ and a reference signal $r(t)$ as an input and outputs a feedback control signal $u(t)$ comprised of the sum of the three terms [36]:

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt} \tag{4.4}$$

where the $K_p, K_i, K_d$ represent the proportional, integral, and derivative components of the controller. A block scheme representing the eq. (4.4) can be seen in fig. 4.10. Each of the terms in the PID controller serves a different purpose [36]:

- The proportional term responds to the immediate value of the error signal $e(f)$.
- The integral term accumulates the error $e(f)$ over time, thus eliminating any steady-state errors.
- The derivative term reflects the rate of change of the error $e(f)$ with the aim to generate a stronger response to a quickly growing error.

The coefficients of the PID terms can be determined by various means. Among the classical ways to tune the PID controller is the *Ziegler-Nichols* method, in which the controller is initially set to proportional mode ($K_i$ and $K_d$ are set to zero). The value of $K_p$ is raised gradually from a small value until the control signal starts oscillating [36]. At this point, the value of $K_p$ and the oscillation period are noted and used to obtain the remaining values for the controller from table 4.1.

Table 4.1: Ziegler-Nichols tuning method rules [36].

| | $K_p$ | $K_i$ | $K_d$ |
|---|---|---|---|
| P | $0.50K_0$ | | |
| PI | $0.45K_0$ | $1.2\frac{K_0}{P_0}$ | |
| PID | $0.60K_0$ | $2\frac{K_0}{P_0}$ | $\frac{K_0 P_0}{8}$ |

# Chapter 5

# Simulating Launch and Landing

Previous chapters described various aspects that need to be considered to create a model of a rocket itself. However, such a model on its own is not sufficient to create a simulation of the flight, as the rocket is affected by and interacts with its environment during the entire flight. Therefore, the rocket model must be complemented by models of various aspects of the environment, such as gravity or atmosphere, described in this chapter.

## 5.1 Atmospheric Model

As was mentioned earlier in chapter 3, the LV spends a significant portion of time in the atmosphere during the launch and landing. Thus, the atmosphere plays a significant role during the flight as it affects multiple aspects, such as the aerodynamic forces or performance of the rocket engine.

The properties of the atmosphere, such as temperature, speed of sound, air pressure, and air density, are not constant but are dependant on time and space [27]. These dependencies can be approximately described using an atmospheric model.

One of the most commonly known models is the *U.S. Standard Atmosphere*. It is an idealized steady-state representation of the atmosphere, which uses a set of equations and tables to provide approximations of various parameters of the atmosphere [20]. The full extent of this model exceeds the scope of this work; therefore, only a small subset of these equations will be presented.

The first important parameter provided by the atmospheric model is the temperature. For altitudes up to 86 km, the altitude is given by a series of seven successive equations in the following general form [20]:

$$T = T_{0,b} + L_{M,b}\left(h - h_b\right) \tag{5.1}$$

where the $T_{0,b}$ for the first layer ($b = 0$) is equal to 288.15 K, and the successive values are obtained from previous calculations of the eq. (5.1). The molecular-scale temperature gradient $L_{M,b}$ and layer reference altitude $h_b$ are given by table 5.1 for each value of $b$.

The pressure, which is used, for example, in the thrust equation (see section 4.3.1), is given for altitudes up to 86 km by the equation

$$P = P_b \left[\frac{T}{T + L_{M,b}\left(h - h_b\right)}\right]^{\frac{gM_0}{R^* L_{M,b}}} \tag{5.2}$$

where $P_b = 10132$ Pa is the reference pressure at sea level, $g$ is the gravitational acceleration, $M_0 = 0.028964\,\mathrm{kg\,mol^{-1}}$ is the molar mass of air at sea level, and $R^*$ is the universal gas constant [20]. The variable $L_{M,b}$ represents molecular-scale temperature gradient, which can be found in table 5.1.

Table 5.1: Atmospheric model variables used for calculation of temperature and pressure [20].

| $b$ | $h_b$ | $L_{M,b}$ |
|---|---|---|
| 0 | 0 | −6.5 |
| 1 | 11 | 0.0 |
| 2 | 20 | 1.0 |
| 3 | 32 | 2.8 |
| 4 | 47 | 0.0 |
| 5 | 51 | −2.8 |
| 6 | 71 | −2.0 |
| 7 | 84.852 | |

From the values obtained in eqs. (5.1) and (5.2), the air mass density, which is necessary for the aerodynamic equations, can be simply calculated [20]:

$$\rho = \frac{PM}{RT} \tag{5.3}$$

Lastly, we need to calculate the speed of sound $a$, which affects the vehicle's aerodynamic characteristics. The equation used in the *U.S. Standard Atmosphere* model is the following:

$$a = \left(\frac{\gamma R^* T}{M_0}\right)^{\frac{1}{2}} \tag{5.4}$$

where $\gamma$ is the ratio of specific heat of air at constant pressure, defined to be exactly 1.40 [20]. The speed of sound then can be used to calculate the Mach number for any velocity vector $V$ by using the following equation:

$$Mach = \frac{\sqrt{V \cdot V}}{a} \tag{5.5}$$

While the presented equations apply only for the lower parts of the atmosphere, the *U.S. Standard Atmosphere* provides approximations for altitudes up to 1000 km. The values computed by the atmospheric model for the lower parts of the atmosphere can be seen in fig. 5.1.

## 5.2 Earth Gravity Model

As the reusable rocket booster never leaves Earth's gravity field, gravity has a significant effect on the vehicle during the whole flight. Furthermore, the vertical distance covered by the booster is large enough to necessitate the use of a geopotential model to calculate the gravitational acceleration.

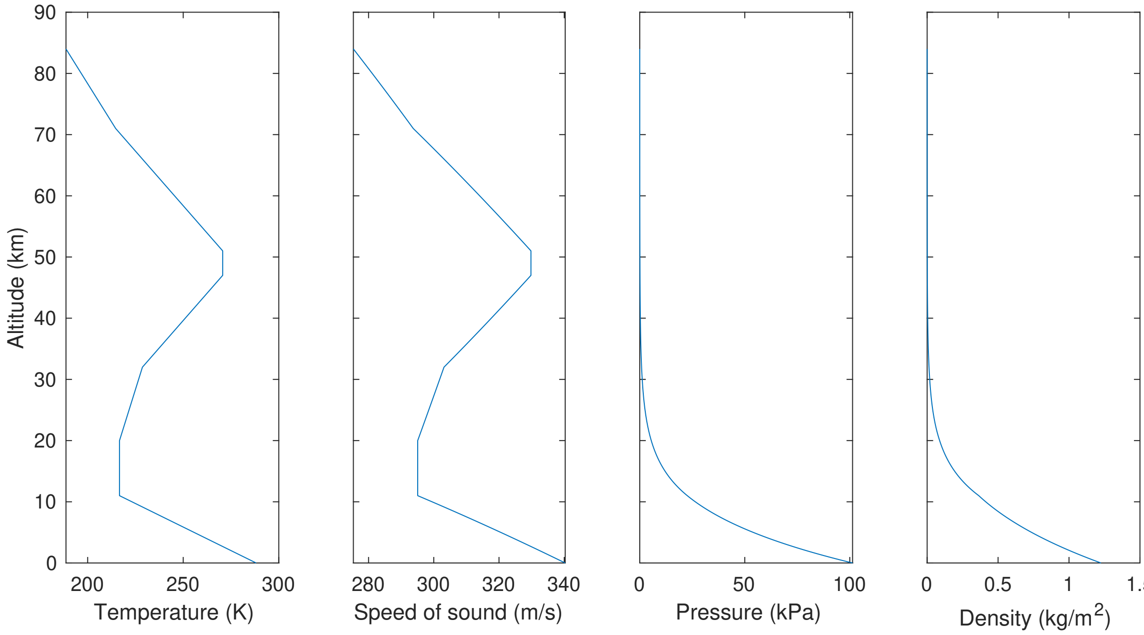



Figure 5.1: Temperature, speed of sound, pressure, and air mass density calculated by the U.S. Standard Atmosphere model for altitudes up to 84 km.

Most of such models are based on a spherical harmonic expansion given by the following expression [9]:

$$U(\Phi,\lambda,h) = \frac{\mu}{r}\Bigg\{-1+\sum_{n=2}^{\infty}\Bigg[\left(\frac{R_E}{r}\right)^n J_n P_{n0}\cos\Phi + \sum_{m=1}^{n}\left(\frac{R_E}{r}\right)^n (C_{nm}\cos m\lambda + S_{nm}\sin m\lambda)\, P_{nm}\cos\Phi\Bigg]\Bigg\} \tag{5.6}$$

This equation expresses the gravitational potential $U(\Phi,\lambda,h)$ at a specified latitude $\Phi$, longitude $\lambda$, and distance from the center of the planet $r$, where $\mu$ is the standard gravitational parameter of the planet (in the case of Earth, $\mu = 3.986004415 \cdot 10^{14}\,\mathrm{m}^3/\mathrm{s}^2$), $R_E$ is the equatorial radius of the Earth, and $P_{nm}$ are Legendre polynomials [9].

Finally, the coefficients $J_n$, called *zonal harmonic coefficients*, reflect the mass distribution of Earth independently of longitude [9]. Similarly, $C_{nm}$ are the *tesseral harmonic coefficients* for $n \neq m$, and $S_{nm}$ are the *sectoral harmonic coefficients* for $n = m$.

Table 5.2: Approximate values of low-order zonal, tesseral, and sectoral harmonic coefficients from the JGM-3 model [32].

| $n$ | $J_n$ |
|---|---|
| 2 | $-0.1083 \times 10^{-2}$ |
| 3 | $0.2532 \times 10^{-5}$ |
| 4 | $0.1619 \times 10^{-5}$ |
| 5 | $0.2277 \times 10^{-6}$ |
| 6 | $-0.5396 \times 10^{-6}$ |

| $n$ | $m$ | $C_{nm}$ | $S_{nm}$ |
|---|---|---|---|
| 2 | 1 | $-0.2414 \times 10^{-9}$ | $0.1543 \times 10^{-8}$ |
| 2 | 2 | $0.1575 \times 10^{-5}$ | $-0.9039 \times 10^{-6}$ |
| 3 | 1 | $0.2193 \times 10^{-5}$ | $0.2680 \times 10^{-6}$ |
| 3 | 2 | $0.3090 \times 10^{-6}$ | $-0.2114 \times 10^{-6}$ |
| 3 | 3 | $0.1006 \times 10^{-6}$ | $0.1972 \times 10^{-6}$ |

The values of these coefficients have been determined mostly experimentally from measurements of Earth-orbiting spacecraft. As a result, there is a growing number of Earth gravity models with various accuracy [9]. As an example, an extract from the JGM-3 model with approximate values of low-order coefficients is presented in table 5.2.

## 5.3 Basic Principles of Continuous Time System Simulation

In the previous chapters, multiple mathematical models for describing the behavior of the rocket booster and the environment were presented. These models are expressed as sets of algebraic and differential equations (most notably the equations of motion, see section 2.3), which describe the rate of change of the state variables [38]. Such systems are called *continuous time systems*.

In general, all these equations can be converted to an Ordinary Differential Equations (ODEs) in the form of:

$$\dot{z}_i(t) = f_i\left(z_1(t), \cdots, z_n(t), u_1(t), \cdots, u_m(t)\right) \tag{5.7}$$

where $z_i(t)$ represents a state variable, $\dot{z}_i(t)$ is the rate of the given state variable, $u_i(t)$ represents the system's input. Then, the function $f_i$ defines the rate of change of the state variable $z_i$ depending on the current state of the system and inputs [38].

It is obvious that the value of any given state variable $z_i$ at time $t$ can be obtained by solving the equation. Unfortunately, the analytical solution of these equations rarely exists, which means they have to be approximated numerically [38].

There are various methods for solving systems of ODEs. The simplest one is the *Euler's method*, which unfortunately generates relatively large errors, making it unsuitable for practical applications. Thus, in practice, one-step methods from the *Runge-Kutta* family of methods or multi-step methods such as *Adams-Bashforth* are utilized.

### 5.3.1 The Runge-Kutta Methods

The *Runge-Kutta methods* are a family of one-step numerical integration methods. As they are one-step methods, they use only only one previous value of state variables $(\boldsymbol{z}(t_{k-1}))$ to calculate the new values $(\boldsymbol{z}(t_k))$.

One of the best-known Runge-Kutta methods is the fourth-order Runge-Kutta method. This is an explicit method, which evaluates the function $\boldsymbol{f}()$ at various points around $t_k$. The $\boldsymbol{z}(t_k))$ is calculated as a weighted average of those values [38]:

$$\boldsymbol{z}(t_{k+1}) = \boldsymbol{z}(t_k) + \frac{h}{6} \cdot (\boldsymbol{k_1} + 2\boldsymbol{k_2} + 2\boldsymbol{k_3} + \boldsymbol{k_4}) \tag{5.8}$$

where

$$\boldsymbol{k_1} = \boldsymbol{f}(\boldsymbol{z}(t_k), t_k) \tag{5.9}$$

$$\boldsymbol{k_2} = \boldsymbol{f}\left(\boldsymbol{z}(t_k) + h \cdot \frac{\boldsymbol{k_1}}{2}, t_k + \frac{h}{2}\right) \tag{5.10}$$

$$\boldsymbol{k_3} = \boldsymbol{f}\left(\boldsymbol{z}(t_k) + h \cdot \frac{\boldsymbol{k_2}}{2}, t_k + \frac{h}{2}\right) \tag{5.11}$$

$$\boldsymbol{k_4} = \boldsymbol{f}\left(\boldsymbol{z}(t_k) + h \cdot \boldsymbol{k_3}, t_k + h\right). \tag{5.12}$$

# Chapter 6

# Implementation

The previous chapters have covered the theoretical foundations necessary for building a reusable rocket booster model and designing and implementing a GNC system for it. This chapter describes the implementation details of the final model and GNC system created as a part of this work.

To interpret the results of the simulation, it is necessary to choose an appropriate means of visualization. While two-dimensional plots remain a valuable tool for the interpretation of many values, they are insufficient for describing the position and movement of a vehicle with 6 DOF.

Therefore, a 3D visualization tool, which provides a convenient way to display the position and movement of the booster, was created for this purpose. In addition, it also visualizes the state of various parts of the launcher, such as the engine throttle. The last section of this chapter describes this visualization tool and its implementation.

## 6.1 Characteristics of the Simulated Reusable Rocket Booster

The reusable rocket booster chosen for simulation is based on the first stage of the SpaceX Falcon 9 rocket. In order to simplify the model, only the booster is considered in the whole simulation, *i.e.*, the rocket has no second stage and payload. To ensure good aerodynamic characteristics during ascent, a nose cone has been added to the top of the booster. A general shape and dimensions of the used booster can be seen in fig. 6.1.

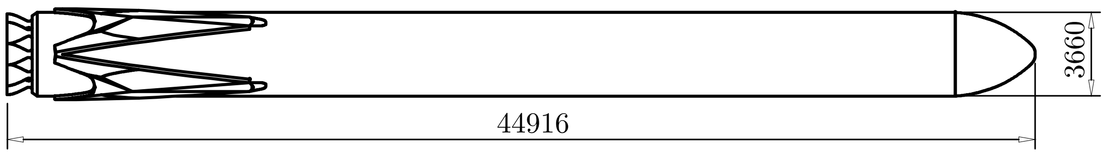



Figure 6.1: Dimensions of the simulated rocket booster.

Like the Falcon 9, the booster has nine gimbaled liquid bi-propellant engines with a total thrust of more than 8300 kN in a vacuum. The thrust and gimbal angles of each engine can be controlled individually. The engines are arranged in a structure called *octaweb* [30], which can be seen in fig. 6.2.

As the number of parameters defining the rocket booster is quite large, it has been omitted from this section. The complete overview of the values used in the simulation can be found in appendix A.

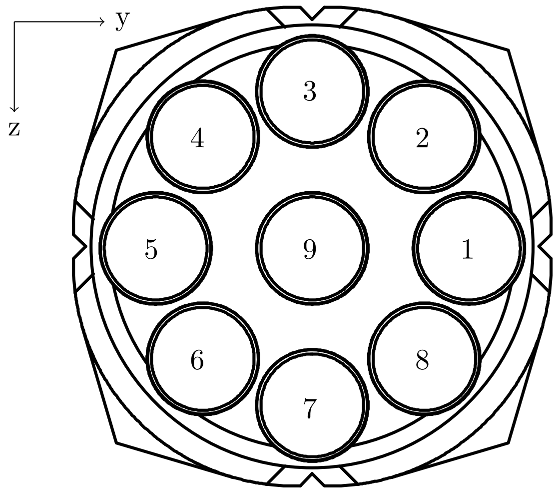


Figure 6.2: Octaweb engine structure.

## 6.2 Model

The whole model has been implemented using the *Simulink* simulation environment in MATLAB R2020b. In Simulink, the model is built using block diagrams, making it easier to understand and debug. Additionally, the already vast library of Simulink blocks can be extended by various blocksets suited towards a specific field. In this work, the *Aerospace Blockset* is used extensively in all instances when it provides an appropriate block. Finally, the control subsystem also utilizes the *Stateflow* add-on for implementing the control logic.

The process of building the model has been iterative, starting from a simple 3 DOF model prototype; the final model was created by gradually adding features and components, such as the aerodynamics or control subsystem.

The resulting model is quite complex due to the large number of aspects that had to be considered. To preserve clarity and make the model easy to understand, its components have been separated into several Simulink subsystems interconnected by buses. This can be seen in the overall structure of the model showcased in fig. 6.3. The various subsystems will be described further in this section, except for the *CZML Export* block, which is described in section 6.3.3.

### 6.2.1 Equations of Motion and Mass Calculator

The core of the system lies in the equations of motion, which describe the dynamics of the rocket booster. The equations described in section 2.3 are implemented in the *Custom Variable Mass 6DOF ECEF* block from the *Aerospace Blockset*. The outputs of the block are routed into the `States` bus along with several other values computed by other blocks, such as the aerodynamic angles, Mach number, and dynamic pressure.

Even though the *Aerospace Blockset* provides several options for handling the vehicle's mass (such as fixed mass or simple variable mass), these have shown to be insufficient for this application. Therefore, a custom *Mass Calculator* block was implemented as seen in fig. 6.4, which integrates the mass flow of the engines to obtain the current mass of the vehicle. The calculated mass is used to estimate the inertia tensor and CG by linear interpolation, which are then used by the equations of motion.

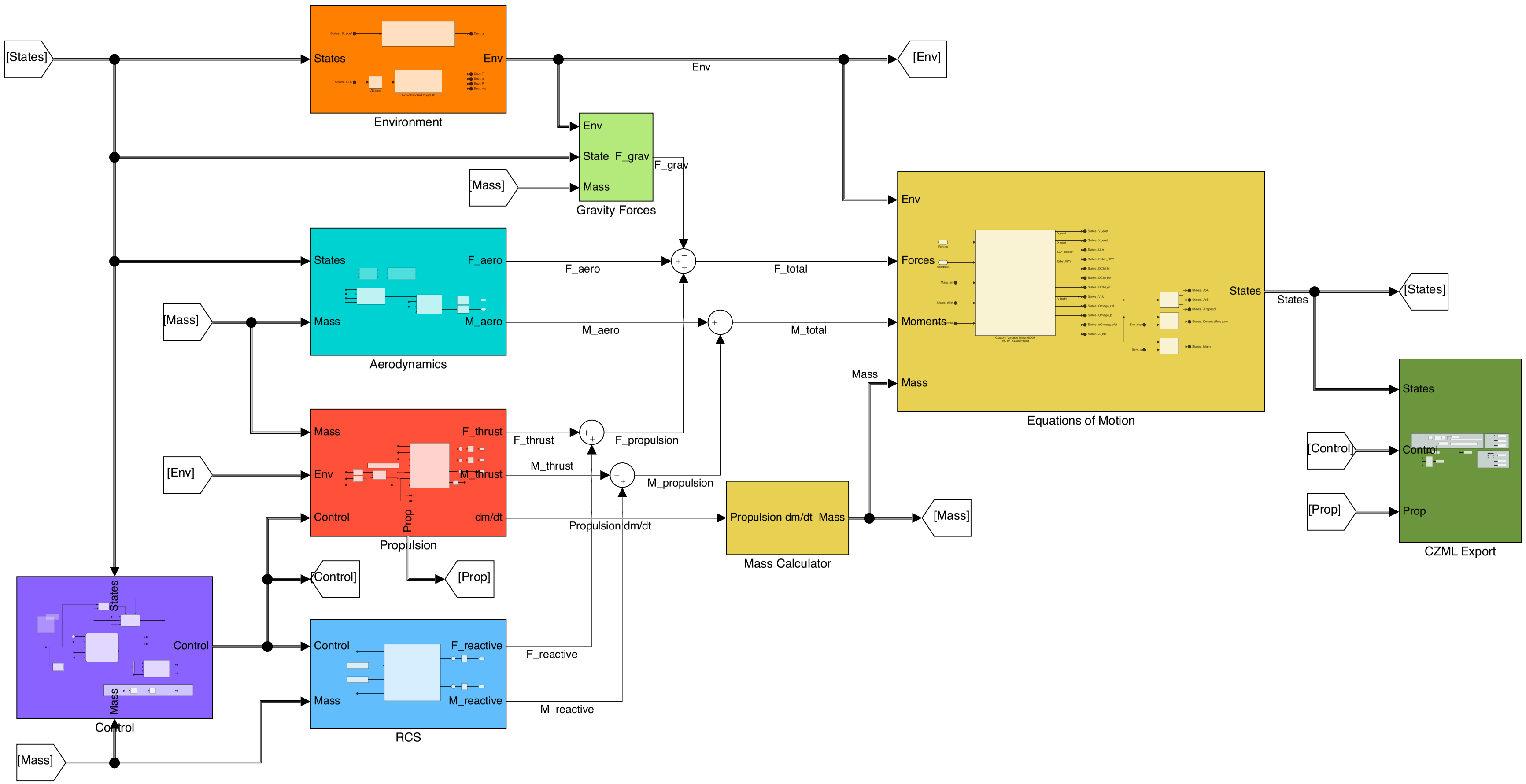



Figure 6.3: The overall structure of the Simulink model.

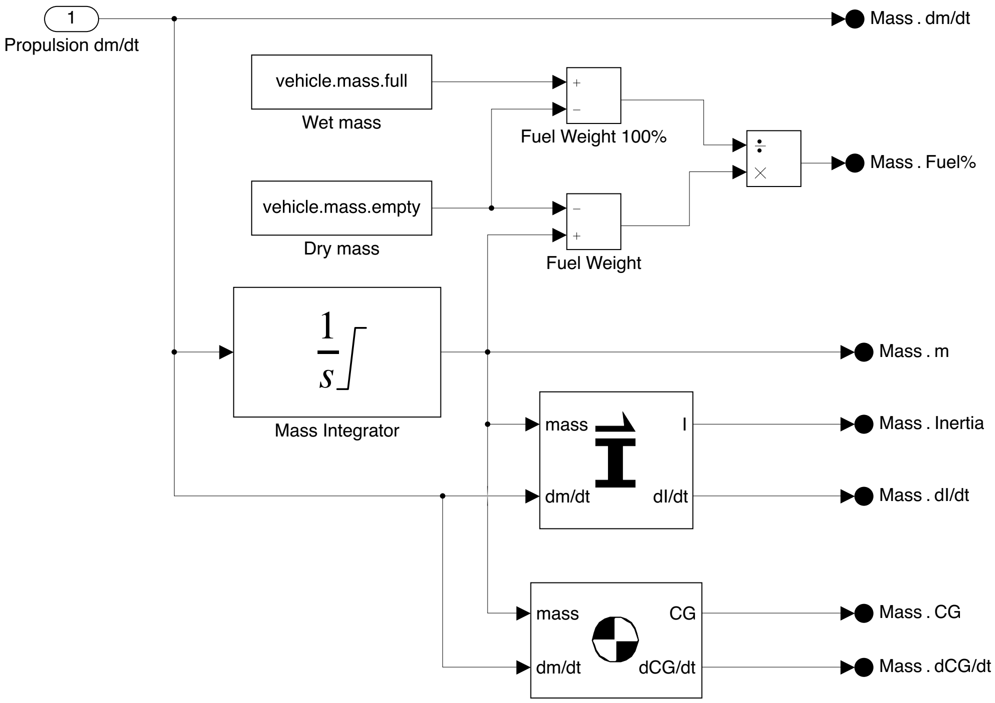



Figure 6.4: Implementation of the mass calculator block.

### 6.2.2 Environment Models and Gravity Forces Transformation

The *Environment* block consists of the gravity model and the atmosphere model, both implemented using standard blocks from *Aerospace Blockset*. While the gravity model uses the zonal harmonic gravity model from section 5.2, the atmospheric model utilizes the MIL-HDBK-310 model, as its implementation was better suited for high-altitude use.

The gravity model outputs the gravitational acceleration in ECEF frame. In accordance with Newton's second law of motion, by multiplying the gravitational acceleration with the current mass of the vehicle, the gravitational forces are obtained. Then, these forces need to be transformed in the BFF axes, which are used by the equations of motion. These operations are performed by the *Gravity Forces block* shown in fig. 6.5.

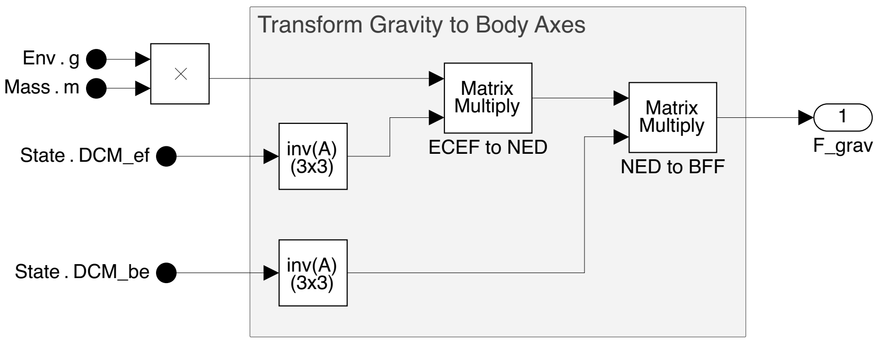



Figure 6.5: Implementation of the gravity forces block.

### 6.2.3 Propulsion and RCS Models

The blocks for propulsion and RCS subsystems are principally similar, as they both simulate each engine/RCS block individually. This is achieved using the *For Each Subsystem*, which evaluates its contents multiple times for each set of input values.

Inside the *For Each Subsystem* are contained models of the simulated entities. In the case of a rocket engine, it contains the *Thrust equation*, implementation of throttling, and calculation of the forces and moments caused by the gimbal.

The RCS implementation is simplified as the nozzles are grouped into RCS blocks. Furthermore, the implementation of the *Thrust equation* is not necessarily required as the RCS thrusters are fired mostly in a vacuum, where their thrust can be considered constant.

### 6.2.4 Aerodynamic Model

The model utilizes the standard *Aerodynamic Forces and Moments* block to calculate the forces and moments due to aerodynamics. This block essentially implements the equations in section 3.1, which need to be supplied with dynamic pressure and aerodynamic coefficients.

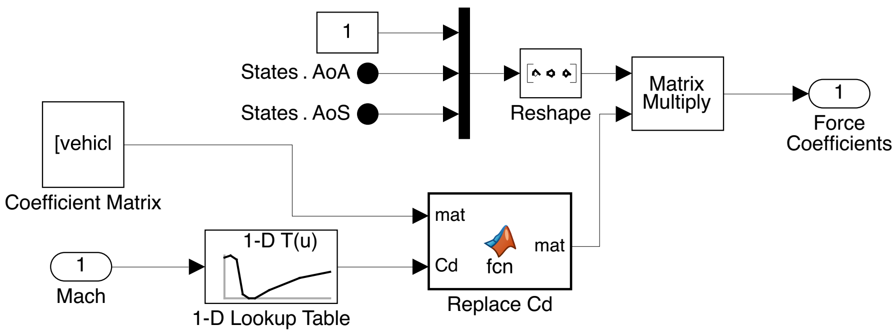



Figure 6.6: Calculation of the force coefficients.

Calculation of the coefficients is done by a custom block, which performs linear interpolation of the aerodynamic data computed by CFD and adds the approximation of components caused by the AoA and AoS. The subsystem for calculation of the force coefficients can be seen in fig. 6.6.

### 6.2.5 Control Subsystem

The *Control* block implements the guidance and controls part of the GNC and consists of multiple subsystems. The *Guidance* Stateflow block is the core of the whole system, as it determines the global state of the booster in response to flight data and controls other parts of the *Control* block. The relationship between various parts of the controls can be seen in fig. 6.7

The *Gimbal Control* subsystems consits of PID controllers that drive the engines' gimbal mechanism so that the vehicle is in the desired orientation and stable during the ascent. Similarly, the roll stabilization is performed by another PID controller, which minimizes the angular velocity in the $x_B$ axis. On the other hand, the controls of the RCS thrusters

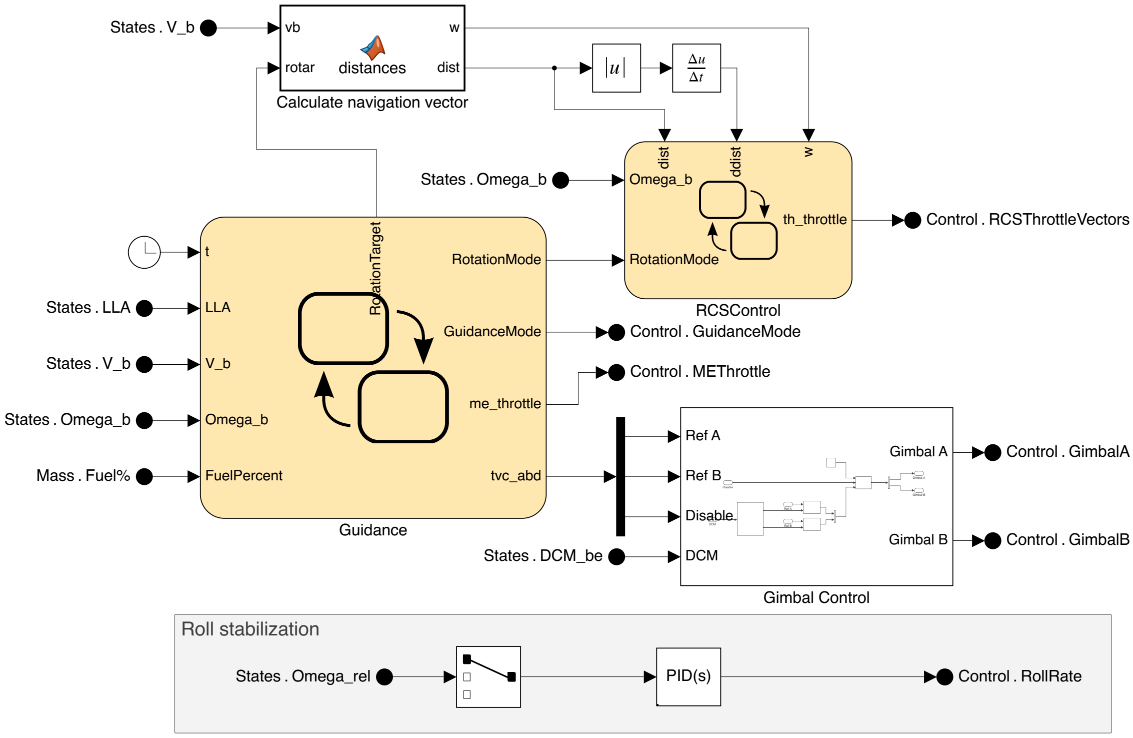



Figure 6.7: Control block and its parts.

are implemented as a Stateflow block. This is more suitable than a PID, as the thrusters only fire in short impulses instead of continuous throttled operation.

## 6.3 Visualization

Visualization is one of the most important aspects of any simulation. While it does not affect the simulation directly, it provides a way to interpret the results by a human. The importance of visualization is even higher in the case of 6 DOF simulation, where a set of 6 variables defines the position and orientation. In such cases, it is appropriate to use interactive 3D visualization, which clearly shows the position and orientation of the vessel.

Even though Simulink's *Aerospace Blockset* provides an interface for using the *FlightGear* flight simulation as a visualization tool for the simulation, this solution has several drawbacks, such as the quality and large size of world scenery files and more complicated workflow for creating custom vehicle models. For this reason, it was decided to use a different visualization tool despite the need to implement a custom interface for Simulink.

The final visualization tool is implemented with the use of the open-source JavaScript library *CesiumJS*. The main benefit is that it provides a high-quality 3D model of the Earth with streamed height data and satellite imagery [5], which provides a helpful context for determining the position of the booster relative to the Earth's surface. Furthermore, thanks to its roots in the aerospace industry, it has good support for visualizing time-dynamic data and supports 3D models in the GL Transmission Format (glTF) format [5], making it an ideal fit for this application.

### 6.3.1 Interfacing CesiumJS with Simulink

As mentioned earlier, there is no existing interface between CesiumJS and Simulink. Fortunately, with the appropriate transformations and modifications, the simulation outputs can be used in CesiumJS, which makes building the interface relatively straightforward.

Cesium can be used both for visualizing precalculated and streaming data. As the simulation built in this work does not use any user interactions, it was decided that streaming the data is unnecessary. Furthermore, precalculating the entire simulation makes it possible to speed up or down the visualization.

Besides the traditional JavaScript Application Programming Interface (API) provided by Cesium, it is also possible to use the Cesium Language (CZML) format for describing the time-dynamic scene visualized by Cesium, thus reducing the amount of the client-side code. CZML, described in the next section, is a JavaScript Object Notation (JSON) based format, which means it can be easily generated using the MATLAB's `jsonencode` function.

### 6.3.2 CZML Format Specification

The CZML document consists of a single JSON array, which contains one or more object literal elements called *packets*. Each packet contains properties with data about a single object in the scene, such as its position or 3D model. A CZML document should also contain a „document“ packet, with information that applies to the whole file, such as the clock settings [28]. The basic structure of the file can be seen in listing 6.1.

Listing 6.1: Basic CZML file structure.

```json
[
  {
    "id": "document",
    "name": "Simulation Result",
    "version": "1.0",
    "clock": {
      "interval": "2021-04-01T12:00:00Z/2021-04-01T12:00:01Z",
      "currentTime": "2021-04-01T12:00:00Z",
    }
  },
  {
    "id": "F9/S1",
    "name": "Stage 1",
    "model": {
      "show": true,
      "gltf": "bv12x.glb"
    },
    ...
  }
]
```

Since the data visualized are time-dynamic, proper timestamping is essential. One way this can be achieved is shown in listing 6.2, where the initial time and date are specified in the `epoch` property in the ISO8601 format. Then, each sample is tagged with the time elapsed since the epoch in seconds. The samples are then arranged sequentially in a single array as `[Time, Sample, Time, Sample, ...]`.

Listing 6.2: Specification of position in CZML.

```
"position": {
  "epoch": "2021-04-01T12:00:00Z",
  "cartesian": [
    0.0, 917841.53, -5530570.17, 3031350.82,
    0.2, 917841.55, -5530570.29, 3031350.89,
    0.4, 917841.57, -5530570.41, 3031350.95,
    ...
  ],
  "interpolationAlgorithm": "LAGRANGE",
  "interpolationDegree": 1,
  "referenceFrame": "FIXED"
}
```

In case of position (listing 6.2), the samples are in the `cartesian` property, which is used for specifying the position as a three-dimensional Cartesian value in meters, relative to the `referenceFrame`, which in this case is ECEF.

Listing 6.3: Specification of orientation in CZML.

```
"orientation": {
  "epoch": "2021-04-01T12:00:00Z",
  "unitQuaternion": [
    0.0, -0.15, -0.18, -0.61, 0.73,
    0.2, -0.16, -0.17, -0.62, 0.73,
    0.4, -0.15, -0.16, -0.63, 0.74,
    ...
  ],
  "interpolationAlgorithm": "LAGRANGE",
  "interpolationDegree": 1
},
```

The rocket's orientation is specified similarly, as can be seen in listing 6.3. However, the `unitQuaternion` property is used in place of `cartesian`, which specifies the orientation as a unit quaternion $(x, y, z, w)$ relative to the ECEF. A method for obtaining the orientation in this format is presented in section 6.3.3.

Lastly, the CZML format also makes it possible to include custom properties that are not specified in the language specification [28]. The custom properties are specified in the same way as the native ones. However, to prevent future naming conflicts, all custom properties should be nested under the `properties` value of the CZML packet. An example of a complete valid CZML document is presented in appendix B.

### 6.3.3 Preparing Simulink Model Data for Export

The interface between Simulink and Cesium is implemented as a MATLAB function, which takes variables outputted by Simulink, creates appropriate structures and then exports them to JSON. Most of the data from the Simulink model are suitable for direct export, such as the vehicle position, as the equations of motion (see section 2.3) already use the ECEF frame. In these cases, it is sufficient to send the data from Simulink using the *To Workspace* block.

In other cases, it is necessary to perform some transformations to obtain values suitable for use with Cesium. For example, the *6DOF ECEF Equations of Motion* block outputs the orientation both as the Euler angles and transformation matrices, but not in the quaternion form.

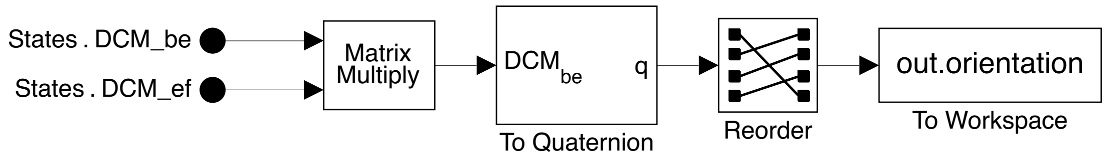



Figure 6.8: Simulink blocks for creating an orientation quaternion suitable for use in Cesium.

The transformation matrices $DCM_{be}$ and $DCM_{ef}$, which describe transformations from NED to BFF, and from ECEF to NED can be combined by matrix multiplication into a single matrix which describes orientation of the vehicle with regards to the ECEF frame.

The resulting transformation matrix is then converted to a quaternion using a block from the *Aerospace Blockset*. Lastly, it is important to change the order of the quaternion components, as the quaternion conventions used by the Aerospace Blockset and Cesium differ. While Cesium uses the $(x, y, z, w)$ order, quaternions in the Aerospace Blockset are specified as $(w, x, y, z)$ [35, 28].

Finally, all desired parameters are exported to MATLAB's workspace using the *To Workspace* blocks. As the amount of generated data can be quite large, it is useful to specify the *Decimation* parameter of the block. The frequency of samples is usually sufficient even after decimation, and can be improved using Cesium's interpolation feature.

After the simulation ends, the `StopFcn` model callback is used to automatically call the function responsible for exporting the final JSON file, which is then loaded in the visualization tool.

### 6.3.4 Implementation of the Visualization Tool

For the sake of simplicity, the visualization tool is implemented as an HTML file with few JavaScript dependencies. Beside the *CesiumJS*, the visualization tool also uses the *Golden-Layout* library, which is a multi-screen web layout manager [12]. This allowed creation of a customizable tiled user interface, which can be seen in fig. 6.9. This is especially important given that the amount of information that can be displayed beside the 3D visualization is substantial. The viewer also displays the past flight trajectory, which can be seen in fig. 6.10.

The implementation itself is simple; the CZML file from Simulink is loaded using the Fetch API, then passed to Cesium, which then displays the entities defined in CZML. Cesium also handles the control of playback time. On each time change, new values of telemetry (already interpolated by Cesium) are obtained from the API and sent to the various components which display them. This way, the consistency of the visualization and displayed data is ensured.

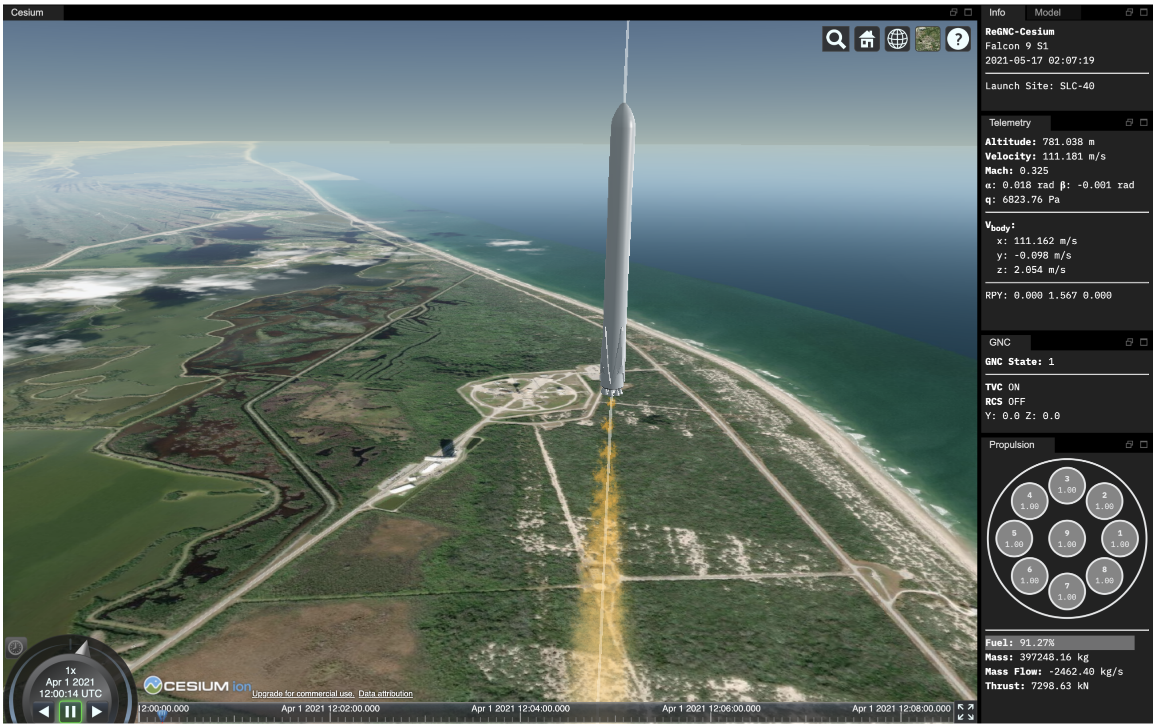



Figure 6.9: Main view of the visualization tool.

Figure 6.10: Visualization of the flight trajectory.

# Chapter 7

# Results

By performing simulations with the model built in the previous chapters and logging its outputs, it is possible to obtain many data about all aspects of the flight. The main focus of this chapter is the analysis of these data, both from the launch portion of the flight and the subsequent landing portion.

## 7.1 Evaluation of the Launch and Landing Performance

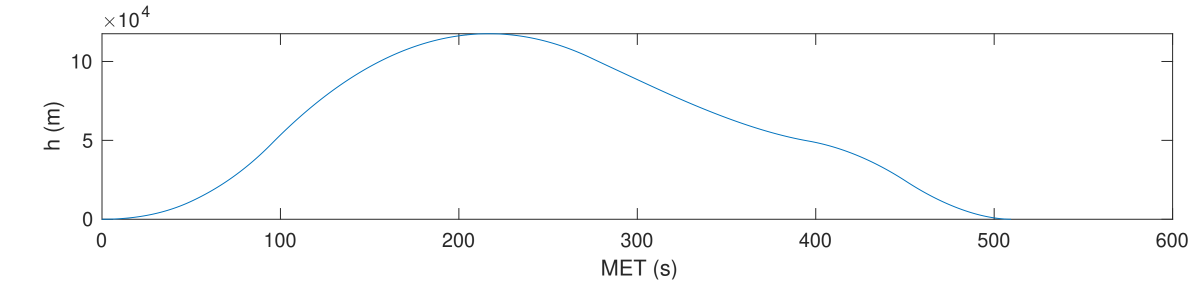
Even though the launch is not a primary concern of this work, it cannot be omitted as it serves multiple purposes. First, it defines the initial conditions for the landing phase depending on the parameters of the booster and launch guidance. Second, it is also helpful during the development of the simulation, as it can be used to validate the model as a whole before implementing the landing phase.

Lets explore the nominal mission profile created for this booster. The launch occurs from the *Cape Canaveral Space Launch Complex 40*, which is one of the two launch facilities used for *Falcon 9*, by igniting all nine engines. At roughly 30 s Mission Elapsed Time (MET) the engines throttle down in order to reduce the aerodynamic loads before reaching the point of maximum dynamic pressure at 50 s MET (see fig. 7.1e). The engines continue to work until 95 s MET, when MECO occurs. After that, the booster continues to gain altitude until about 215 s MET, which can be seen in fig. 7.1a.

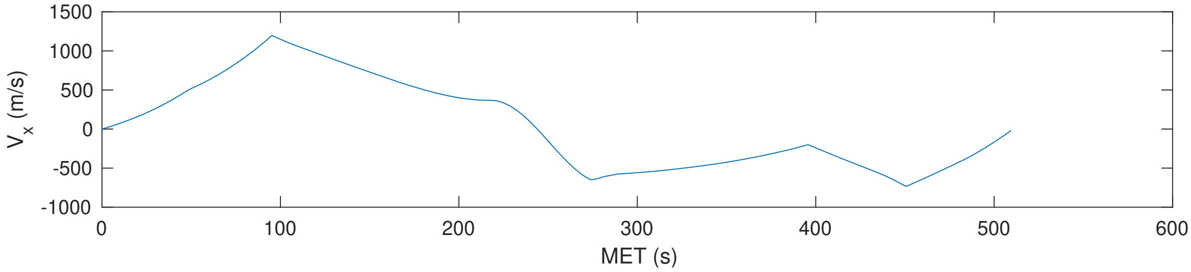
Figure 7.1c shows the increase of thrust force due to the change of atmospheric pressure during the ascent, just as expected. Also noteworthy is fig. 7.1b, which shows the vertical speed of the booster. As can be seen, the speed increases steadily before the MECO. This is caused mainly by the decrease of mass, absence of drag force, and increased thrust in vacuum.

After reaching the highest point of the trajectory, the booster then starts the descent and reorients itself into an engine-first orientation. The fig. 7.1c shows that a braking burn with three engines is performed around 273 s MET with the aim to reduce speed in the atmosphere in order to reduce the aerodynamic forces. Finally, the landing burn starts at 450 s MET, at the end of which the booster lands.

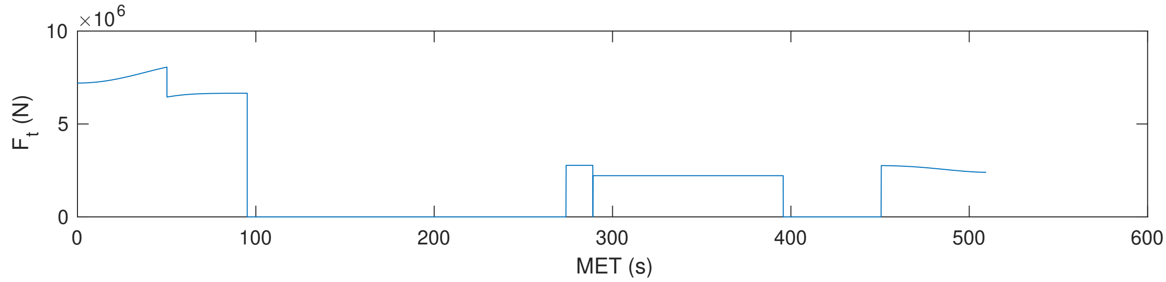
The landing spot is located approximately 87.5 km west from the launch site. The vertical speed of the booster at the moment of landing is $-25.39\,\mathrm{m\,s^{-1}}$, which is not ideal, but sufficient in this case. The booster lands with a generous reserve of 16.65 % of fuel (see fig. 7.1d). This was expected, as the booster launched without the second stage and payload.



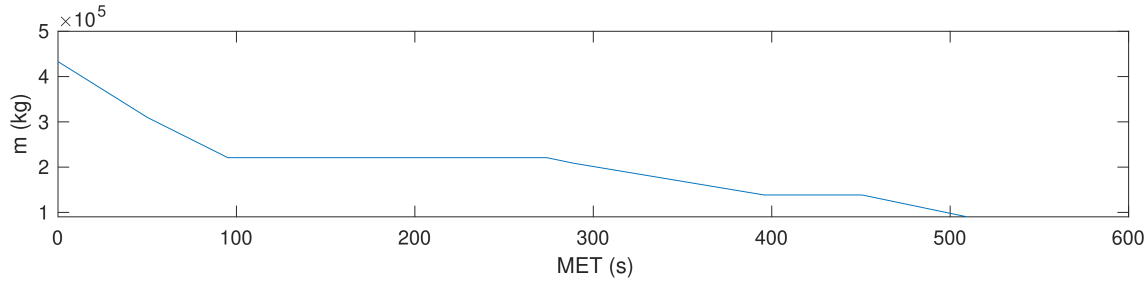

(a) Altitude



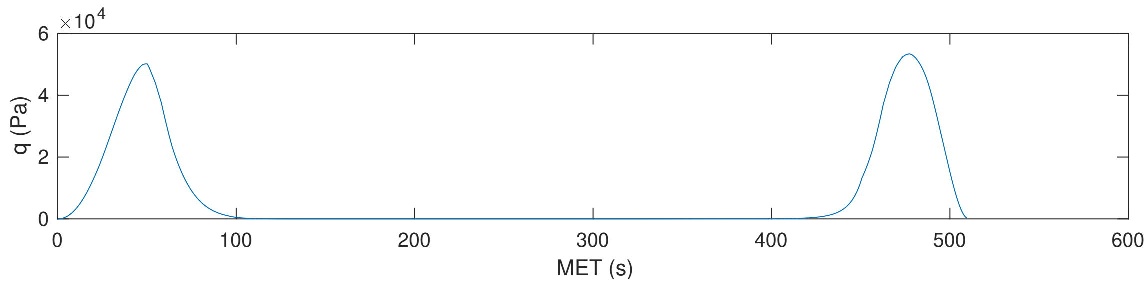

(b) Vertical speed




(c) Thrust force




(d) Mass




(e) Dynamic pressure

Figure 7.1: Plots of variables obtained by the simulation.

## 7.2 Potential Future Improvements

As can be seen in the previous chapters, building a simulation model and GNC system of a reusable booster involves many tasks and engineering areas. Unfortunately, exploring those subjects in more detail would far exceed the extent of this work. Thus, numerous simplifications have had to be done to make the scope of the work manageable.

The simulation model itself presents a good framework and a starting point for exploring the characteristics of the reusable rocket booster and building a GNC system for such a vehicle. While care was taken to capture all the essential aspects of such simulation, there is still much room for potential improvements. Among the aspects that would benefit the most are aerodynamics, which could be substantially improved with a more detailed CFD study. This could be coupled with the improvement of the atmospheric model with gust and turbulence models, which would present additional challenges for the development of the GNC.

The upgrades in GNC could be numerous too, starting from minor improvements such as the implementation of models for sensors and actuators, up to enhancements in guidance, which could enable, for example, pinpoint precision landing and return to the launch site.

Lastly, the visualization environment could be improved mainly by implementing more particle effects and rocket model animations. There is also room for user interface improvements, such as the implementation of gauges and plots.

# Chapter 8

# Conclusion

In this work, a simple Guidance, Navigation, and Control (GNC) system capable of launching and landing a reusable rocket booster was developed. To achieve this, it was necessary to develop a suitable simulation model, which could be used to develop the GNC and for experiments with the already completed system.

First, the theoretical foundations necessary for this work were explained, such as the coordinate systems, dynamics, and basic principles of aerodynamics. Then, the focus shifted to an overview of the design of spacecraft GNC. Finally, the theoretical part concluded with explanations of the environment models and simulation theory.

The simulation model developed in this work implements 6 DOF variable mass dynamic model, custom propulsion and RCS models, and standard environment models. As no suitable aerodynamic model was available, a simple custom model was created as part of this work using the data obtained from CFD studies performed on a CAD model of the booster.

To display the data in an intuitive and comprehensible manner, a suitable visualization was required. Thus, in the last part, a custom 3D visualization tool was developed using the *CesiumJS* library. The position and orientation of the vehicle are shown on an interactive 3D model of the Earth with high-quality satellite imagery, which provides a useful context for interpretation of the visualized data.

# Bibliography


[1] *IAU Nomenclature for Fundamental Astronomy*. 2007. Available at: https://syrte.obspm.fr/iauWGnfa/index.html.

[2] Acikmese, B., Carson, J. M. and Blackmore, L. Lossless Convexification of Nonconvex Control Bound and Pointing Constraints of the Soft Landing Optimal Control Problem. *IEEE Transactions on Control Systems Technology*. november 2013, vol. 21, no. 6, p. 2104–2113. DOI: 10.1109/TCST.2012.2237346. ISSN 1063-6536, 1558-0865. Available at: http://ieeexplore.ieee.org/document/6428631/.

[3] Blackmore, L. Autonomous Precision Landing of Space Rockets. *The Bridge*. 2016, vol. 46, no. 4, p. 15–20. ISSN 0737-6278.

[4] Cai, G., Chen, B. M. and Lee, T. H. *Unmanned rotorcraft systems*. New York: Springer, 2011. Advances in industrial control. ISBN 9780857296344.

[5] Cesium GS, Inc.. *CesiumJS*. Available at: https://cesium.com/platform/cesiumjs/.

[6] Davies, M., ed. *The standard handbook for aeronautical and astronautical engineers*. New York: McGraw-Hill, 2003. McGraw-Hill standard handbooks. ISBN 9780071362290.

[7] Dumont, E., Ecker, T., Chavagnac, C., Witte, L., Windelberg, J. et al. CALLISTO - Reusable VTVL launcher first stage demonstrator. In:. Sevilla, Spanien: [b.n.], May 2018. Available at: https://elib.dlr.de/119728/.

[8] Durham, W. *Aircraft flight dynamics and control*. Chichester, West Sussex: John Wiley & Sons, Inc, 2013. ISBN 9781118646786 9781118646793 9781118646809.

[9] Fortescue, P. W., Stark, J. and Swinerd, G., ed. *Spacecraft systems engineering*. 3rd edth ed. New York: J. Wiley, 2003. ISBN 9780470851029 9780471619512.

[10] Grewal, M. S., Weill, L. R. and Andrews, A. P. *Global positioning systems, inertial navigation, and integration*. 2nd edth ed. Hoboken, N.J: Wiley-Interscience, 2007. ISBN 9780470041901. OCLC: ocm70259027.

[11] Groves, P. D. *Principles of GNSS, inertial, and multisensor integrated navigation systems*. 2nd edth ed. Boston: Artech House, 2013. GNSS technology and application series. ISBN 9781608070053. OCLC: ocn820530994.

[12] Hempel, W. *GoldenLayout - a multi-window JavaScript layout manager for Webapps*. Available at: http://golden-layout.com/.

[13] Iaco Veris, A. de. *Fundamental Concepts on Liquid-Propellant Rocket Engines*. 2021. ISBN 9783030547042.

[14] Jamshed, S. *Using HPC for computational fluid dynamics: a guide to high performance computing for CFD engineers*. Amsterdam: Elsevier/Academic Press, 2015. ISBN 9780128015674. OCLC: ocn918792203.

[15] Jia, Y.-B. Quaternions. *Com S 477/577 Notes*. 2020. Available at: http://web.cs.iastate.edu/~cs577/handouts/quaternion.pdf.

[16] Klevanski, J., Ecker, T., Riehmer, J., Reimann, B., Dumont, E. et al. Aerodynamic Studies in Preparation for CALLISTO - Reusable VTVL Launcher First Stage Demonstrator. In:. Bremen, Germany: [b.n.], October 2018. Available at: https://elib.dlr.de/122062/.

[17] Koks, D. *Using Rotations to Build Aerospace Coordinate Systems*. DSTO-TN-0640. DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION SALISBURY (AUSTRALIA) SYSTEMS SCIENCES LAB, august 2008. Available at: https://apps.dtic.mil/docs/citations/ADA484864.

[18] Marwege, A., Riehmer, J., Klevanski, J., Gülhan, A., Ecker, T. et al. First Wind Tunnel Data of CALLISTO - Reusable VTVL Launcher First Stage Demonstrator. In:. Madrid, Spanien: [b.n.], July 2019. Available at: https://elib.dlr.de/128629/.

[19] National Aeronautics and Space Administration. *Project Apollo Coordinate System Standards*. Washington, D.C.: [b.n.], 1965. Available at: http://www.ibiblio.org/apollo/Documents/19700076120.pdf.

[20] National Aeronautics and Space Administration. *U.S. Standard Atmosphere*. NASA-TM-X-74335. 1976. Available at: https://ntrs.nasa.gov/citations/19770009539.

[21] Noureldin, A., Karamat, T. B. and Georgy, J. *Fundamentals of Inertial Navigation, Satellite-based Positioning and their Integration*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. ISBN 9783642304651 9783642304668. Available at: http://link.springer.com/10.1007/978-3-642-30466-8.

[22] Oliver, T. E., Park, T., Anzalone, E., Smith, A., Strickland, D. et al. Space Launch Systems Block 1B Preliminary Navigation System Design. In:. 2018. Available at: https://ntrs.nasa.gov/citations/20180002039.

[23] Ortega, G. *ESA Guidance, Navigation and Control Systems*. 2014. Available at: http://www.et.tu-dresden.de/ifa/fileadmin/user_upload/www_files/Aktuelles/2014-01-27_ESA_Guidance__Navigation__and_Control_Systems.pdf.

[24] Pisacane, V. L., ed. *Fundamentals of space systems*. 2nd edth ed. Oxford ; New York: Oxford University Press, 2005. The Johns Hopkins University/Applied Physics Laboratory series in science and engineering. ISBN 9780195162059.

[25] Rose, D. *Quaternions*. 2015. Available at: http://danceswithcode.net/engineeringnotes/quaternions/quaternions.html.

[26] Ross, J. *SpaceX Falcon 9 Downrange Propulsive Landing (No Boostback)*. Available at: https://zlsadesign.com/infographic/trajectory/spacex-falcon9-booster-dpl-no-boostback/.

[27] Saha, K. *The Earth's Atmosphere: Its Physics and Dynamics*. Berlin: Springer, 2008. ISBN 9783540784265. OCLC: ocn213114189.

[28] Scott, J. *Lift & Drag vs. Normal & Axial Force*. 2004. Available at: http://www.aerospaceweb.org/question/aerodynamics/q0194.shtml.

[29] Sforza, P. M. *Manned Spacecraft Design Principles*. Oxford: Butterworth-Heinemann, 2016. Elsevier aerospace engineering series. ISBN 9780128044254. OCLC: ocn946603824.

[30] Space Exploration Technologies Corp.. *Falcon User's Guide*. 2020. Available at: https://www.spacex.com/media/Falcon_Users_Guide_082020.pdf.

[31] Stevens, B. L., Lewis, F. L. and Johnson, E. N. *Aircraft control and simulation: dynamics, controls design, and autonomous systems*. Third editionth ed. Hoboken, N.J: John Wiley & Sons, 2016. ISBN 9781118870983. OCLC: ocn935444384.

[32] Tapley, B. D., Schutz, B. E. and Born, G. H. *Statistical Orbit Determination*. Amsterdam ; Boston: Elsevier Academic Press, 2004. ISBN 9780126836301.

[33] The Boeing Company. *Space Shuttle Main Engine Orientation*. 1998. Available at: http://large.stanford.edu/courses/2011/ph240/nguyen1/docs/SSME_PRESENTATION.pdf.

[34] The MathWorks, Inc.. *About Aerospace Coordinate Systems - MATLAB & Simulink*. 2020. Available at: https://www.mathworks.com/help/aeroblks/about-aerospace-coordinate-systems.html.

[35] The MathWorks, Inc.. *Aerospace Blockset™ User's Guide*. 2021. Available at: https://www.mathworks.com/help/pdf_doc/aeroblks/aeroblks.pdf.

[36] Wang, L. *PID Control System Design and Automatic Tuning using MATLAB/Simulink*. 1st ed. Wiley, march 2020. ISBN 9781119469346 9781119469414. Available at: https://onlinelibrary.wiley.com/doi/book/10.1002/9781119469414.

[37] Wie, B. *Space vehicle dynamics and control*. 2nd edth ed. Reston, VA: American Institute of Aeronautics and Astronautics, 2008. AIAA education series. ISBN 9781563479533. OCLC: ocn232786911.

[38] Zeigler, B. P., Muzy, A. and Kofman, E. *Theory of Modeling and Simulation: Discrete Event & Iterative System Computational Foundations*. 3rdth ed. USA: Academic Press, Inc., 2018. ISBN 9780128133705.

[39] Zipfel, P. H. *Modeling and simulation of aerospace vehicle dynamics*. 2nd edth ed. Reston, Va: American Institute of Aeronautics and Astronautics, 2007. AIAA education series. ISBN 9781563478758. OCLC: ocm78072255.

# List of Acronyms

| | |
|---|---|
| AoA | Angle of Attack |
| AoS | Angle of Sideslip |
| API | Application Programming Interface |
| BFF | Body Fixed Frame |
| CAD | Computer-Aided Design |
| CFD | Computational Fluid Dynamics |
| CG | Center of Gravity |
| CTP | Conventional Terrestrial Pole |
| CZML | Cesium Language |
| DOF | Degrees of Freedom |
| ECEF | Earth-Centered, Earth-Fixed frame |
| ECI | Earth-Centered Inertial frame |
| FDM | Finite-Difference Method |
| FVM | Finite-Volume Method |
| glTF | GL Transmission Format |
| GNC | Guidance, Navigation, and Control |
| GNSS | Global Navigation Satellite System |
| GPS | Global Positioning System |
| IFOG | Interferometric Fiber-Optic Gyro |
| IMU | Inertial Measurement Unit |
| JSON | JavaScript Object Notation |
| LV | Launch Vehicle |
| MECO | Main Engine Cut Off |
| MET | Mission Elapsed Time |
| NED | North-East-Down |
| ODE | Ordinary Differential Equation |
| PID | Proportional–Integral–Derivative |

| | |
|---|---|
| RCS | Reaction Control System |
| RLG | Ring Laser Gyroscope |
| TT | Terestrial Time |
| TVC | Thrust Vector Control |
| VTVL | Vertical Takeoff, Vertical Landing |

# List of Symbols

| | |
|---|---|
| $a$ | Speed of Sound |
| $\alpha$ | Angle of Attack |
| $A$ | Axial Force |
| $\beta$ | Angle of Sideslip |
| $C_A, C_S, C_N$ | Axial, sideforce, and normal force coefficients |
| $C_\ell, C_m, C_n$ | Roll, pitch, and yaw moment coefficients |
| $D$ | Drag Force |
| $g$ | Gravitational acceleration |
| $L$ | Lift Force |
| $m$ | Mass |
| $\dot{m}$ | Mass flow |
| $N$ | Normal Force |
| $P$ | Pressure |
| $\Phi, \lambda, h$ | Latitude, longitude, and altitude |
| $\phi, \theta, \psi$ | Roll, pitch, and yaw angles |
| $p, q, r$ | Rotation rates around the axes |
| $\rho$ | Density of air mass |
| $R_a(\alpha)$ | Rotation around axis $a$ by angle $\alpha$ |
| $S$ | Side Force |
| $T$ | Temperature |
| $\boldsymbol{T_A^B}$ | Transformation from coordinate system $\boldsymbol{A}$ to $\boldsymbol{B}$ |

# Appendix A

# Parameters of the Rocket Booster

This section presents the vehicle parameters used as the input values for the model implemented in this work. These parameters are either obtained from official sources [30], unofficial sources, or approximated from other available information.

## Mass Properties

| | **Dry** | **Wet** |
|---|---|---|
| Mass | 22 000 kg | 433 100 kg |
| Inertia[1] | $2.68 \times 10^6 \cdot I_3$ kg m$^{-2}$ | $4.95 \times 10^6 \cdot I_3$ kg m$^{-2}$ |
| Center of Gravity | $\begin{bmatrix} 15 & 0 & 0 \end{bmatrix}^T$ m | $\begin{bmatrix} 25 & 0 & 0 \end{bmatrix}^T$ m |

## Aerodynamic Properties

| **Force Coefficients** | |
|---|---|
| $C_{A_\alpha}$ | 0.05 |
| $C_{A_\beta}$ | 0.05 |
| $C_{S_\beta}$ | 0.05 |
| $C_{N_\alpha}$ | 0.05 |
| **Moment Coefficients** | |
| $C_{\ell_p}$ | 0.8 |
| $C_{m_\alpha}$ | 0.05 |
| $C_{m_q}$ | 0.8 |
| $C_{n_\beta}$ | 0.05 |
| $C_{n_r}$ | 0.8 |
| **Geometry** | |
| Reference Area | 10.5209 m$^2$ |
| Reference Length | 3.66 m |
| Reference Span | 3.66 m |

[1] $I_3$ denotes the $3 \times 3$ identity matrix.

## Engine Parameters

| **Merlin 1D** | |
|---|---|
| Flow area | $A_e = 1.227185\,\mathrm{m}^2$ |
| Static pressure at exhaust | $P_e = 84424\,\mathrm{Pa}$ |
| Effective exhaust velocity | $v_e = 3000\,\mathrm{m\,s^{-1}}$ |
| Mass flow | $\dot{m} = 273.6\,\mathrm{kg\,s^{-1}}$ |
| Minimum throttle level | $40\,\%$ |
| Maximum gimbal angle | $\delta_{max} = 5\,°$ |

## RCS Thruster Parameters

| **Parameter** | **Value** |
|---|---|
| Maximum Thrust | $2000\,\mathrm{N}$ |
| RCS block positions | $\begin{bmatrix} 39 & 1.83 & 0 \end{bmatrix}$ |
| | $\begin{bmatrix} 39 & 0 & -1.83 \end{bmatrix}$ |
| | $\begin{bmatrix} 39 & -1.83 & 0 \end{bmatrix}$ |
| | $\begin{bmatrix} 39 & 0 & 1.83 \end{bmatrix}$ |

# Appendix B

# Example CZML Document


```json
[
  {
    "id": "document",
    "name": "Simulation Result",
    "version": "1.0",
    "clock": {
      "interval": "2021-04-01T12:00:00Z/2021-04-01T12:00:01Z",
      "currentTime": "2021-04-01T12:00:00Z",
      "multiplier": 1,
      "range": "LOOP_STOP",
      "step": "SYSTEM_CLOCK_MULTIPLIER"
    }
  },
  {
    "id": "F9/S1",
    "name": "Stage 1",
    "model": {
      "show": true,
      "gltf": "http://localhost:8001/bv12x.glb"
    },
    "point": {
      "color": {
        "rgba": [245, 0, 0, 230]
      },
      "pixelSize": 5
    },
    "path": {
      "show": true,
      "width": 3,
      "material": {
        "polylineOutline": {
          "color": {
            "rgba": [255, 255, 255, 128]
          },
          "outlineColor": {
            "rgba": [255, 255, 255, 200]
          },
          "outlineWidth": 0
```

```
        }
      },
      "trailTime": 200,
      "leadTime": 1
    },
    "position": {
      "epoch": "2021-04-01T12:00:00Z",
      "cartesian": [
        0.0, 917841.53, -5530570.17, 3031350.82,
        0.2, 917841.55, -5530570.29, 3031350.89,
        0.4, 917841.57, -5530570.41, 3031350.95,
        ...
      ],
      "interpolationAlgorithm": "LAGRANGE",
      "interpolationDegree": 1,
      "referenceFrame": "FIXED"
    },
    "orientation": {
      "epoch": "2021-04-01T12:00:00Z",
      "unitQuaternion": [
        0.0, -0.15, -0.18, -0.61, 0.73,
        0.2, -0.16, -0.17, -0.62, 0.73,
        0.4, -0.15, -0.16, -0.63, 0.74
      ],
      "interpolationAlgorithm": "LAGRANGE",
      "interpolationDegree": 1
    },
    "properties": {
      "velocity": {
        "epoch": "2021-04-01T12:00:00Z",
        "cartesian": [
          0.0, 0, 0, 0,
          0.2, 1.41, 0.00, 0.01,
          0.3, 1.91, 0.00, 0.00
        ],
        "interpolationAlgorithm": "LAGRANGE",
        "interpolationDegree": 1
      },
      "q": {
        "epoch": "2021-04-01T12:00:00Z",
        "number": [
          0, 0,
          0.2, 1.22,
          0.4, 2.36
        ],
        "interpolationAlgorithm": "LAGRANGE",
        "interpolationDegree": 1
      }
    }
  }
]
```