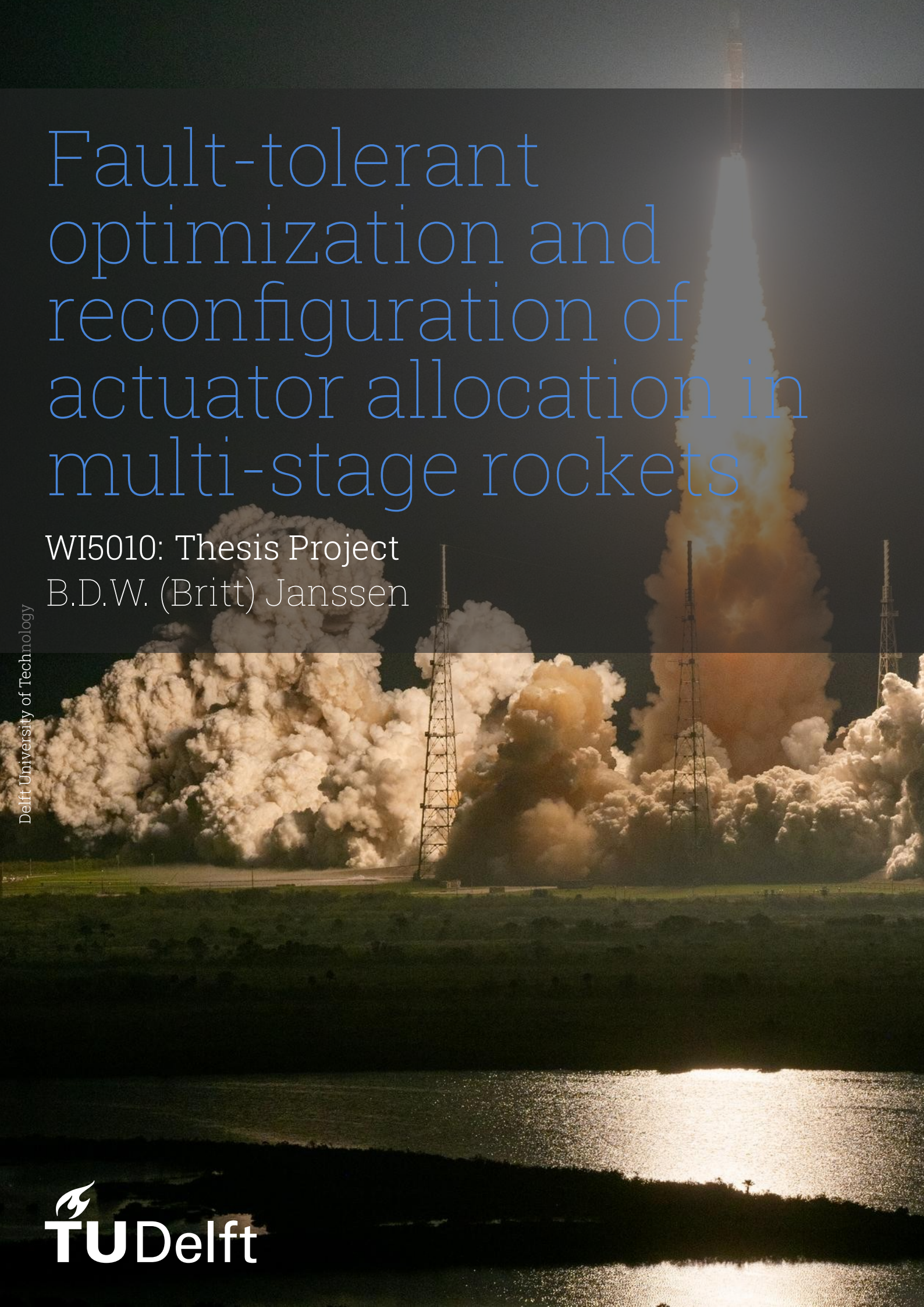
Fault-tolerant optimization and reconfiguration of actuator allocation in multi-stage rockets

WI5010: Thesis Project

B.D.W. (Britt) Janssen

Delft University of Technology



TU Delft

Fault-tolerant optimization and reconfiguration of actuator allocation in multi-stage rockets

by

B.D.W. (Britt) Janssen

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday July 3rd, 2025 at 15:00.

Student number:	5191343	
Group:	Discrete Mathematics and Optimization, Applied Mathematics	
Project duration:	November 18, 2024 – June 19, 2025	
Supervisors:	Dr. Y. Murakami,	TU Delft
	Dr.ir. J. T. van Essen,	TU Delft
	Dr. P. Lourenço,	GMV
Thesis Committee:	Dr. ir. G.F. Nane	TU Delft

Cover: Space Launch System (SLS) rocket by NASA/Joel Kowsky

TU Delft

Preface

This thesis marks the completion of my academic journey with a Master's degree in Applied Mathematics at Delft University of Technology. Conducted within the Discrete Mathematics and Optimization group and in collaboration with GMV. This project allowed me to explore a field at the intersection of aerospace technology, control systems, and optimization. While aerospace was entirely new to me at the start, I was quickly drawn in by its complexity and relevance. Optimization, on the other hand, was my specialization throughout the Master's program. It was nice to see real-world applications of optimization and how the two fields intersect. The problem challenged me technically, and thereby also deepened my appreciation of the application of mathematics in engineering problems.

I would like to express gratitude to my company supervisor, Pedro Lourenço. His expertise in fault-tolerant control and optimization gave sharp insights to the thesis. Also, I would like to thank his steady encouragement and constructive feedback throughout the process, his guidance played a crucial role in the development of this thesis. Furthermore, I would like to thank my other colleagues, Nuno Paulino, for his creative thinking and great knowledge on launchers, Hugo Pereira, for his expertise and enthusiasm in optimization and Matteo Pascucci for his well illustrated explanations of the control system architecture and simulation data. Besides my other colleagues who have inspired me by their enthusiasm in working in the aerospace field and being always open to explain and teach the technical context. I would like to sincerely thank Yuki Murakami, for his consistent support throughout this project. His thoughtful feedback and insightful questions helped me stay on track. Furthermore, I sincerely appreciate Theresia van Essen and Tina Nane, for taking the time to review my thesis and for being part of my thesis committee.

During my work, I really enjoyed the creative thinking involved in developing the optimization formulations, the steep but rewarding learning curve of getting into a interesting new field, and the new people I met who supported, challenged, and inspired me throughout this time.

Summary

Rockets are becoming increasingly complex, with multiple stages, clustered engines, and diverse actuator types. As a result, so do the demands on their control systems. These systems must not only steer the rocket precisely under nominal conditions but also adapt in the presence of actuator faults. Therefore, this makes effective and fault-tolerant control allocation of high importance. Control allocation determines how control commands, such as desired forces and torques, are allocated among actuators. This thesis presents a study of optimization-based control allocation methods for multi-stage rockets. It proposes three optimization formulations that are flexible and extensible, designed for fault tolerance and scalability. Specifically, it targets reusable vehicles equipped with thrust vector control (TVC) systems, aerodynamic fins, and reaction control systems (RCS).

The main objective in the work is the optimal distribution of force and torque commands from the guidance and control system to a set of actuators in both nominal and faulty conditions. The thesis evaluates several control allocation formulations, including Quadratic Programming (QP), Second-Order Cone Programming (SOCP), and Mixed-Integer SOCP (MISOCP) approaches. The models minimize error between commanded and achieved forces and torques by actuator output, while also minimizing actuator effort for efficient control.

A key contribution is the ability to integrate fault scenarios and multiple actuators without changing the structure of the optimization problems. Faults considered are thrust loss, TVC jamming, and loss of TVC power. These faults are included via constraint tightening, allowing the system to reconfigure the allocation strategy over the available non-faulty actuators.

The first model this thesis proposes is the Angle-Deflection (AD) formulation, which allocates the control commands over the deflections of the TVC. Furthermore, a bilinear problem formulation is considered that models both deflection angles and throttleable thrust of engines. The thesis proposes two convex problems for this bilinear problem, through a linearized method and a force-based SOCP reformulation.

To validate the proposed formulations, simulations were conducted using realistic control commands and fault cases. Results demonstrate the effectiveness of the optimization-based strategies in achieving low demand errors and reallocation strategies in case of faults. Furthermore, computational performance was analyzed using embedded solvers like ECOS.

In conclusion, the thesis provides scalable, fault-tolerant optimization frameworks for control allocation in multi-stage rockets. Contributing to development of efficient and resilient control allocation. Future work can focus on improving these methods by optimal (dynamic) parameter tuning, methods for convex formulations including discrete actuators, an adaptation for on-board implementation and research in reallocation solutions for multiple faults.

Contents

Preface i

Summary ii

Nomenclature v

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Application of control allocation in multi-stage rockets . . . 1
  - 1.2.1 Actuators . . . 3
  - 1.2.2 System architecture . . . 4
  - 1.2.3 Faults . . . 5
- 1.3 Goals . . . 5
- 1.4 Outline . . . 6

2 Control Allocation in Literature 7
- 2.1 Optimization background . . . 7
  - 2.1.1 Second-Order Cone Programming (SOCP) formulation . . . 8
  - 2.1.2 Interior Point method (IPM) . . . 8
- 2.2 Optimization-based control allocation . . . 9
  - 2.2.1 Error minimization . . . 9
  - 2.2.2 Effort minimization . . . 9
  - 2.2.3 Hierarchical dynamic control . . . 10
- 2.3 Fault models and types . . . 11

3 Control Allocation formulations 12
- 3.1 Force computation . . . 12
- 3.2 Rotation Matrix . . . 14
- 3.3 Torque computation . . . 15
- 3.4 Angle-Deflection problem . . . 15
  - 3.4.1 Angle-Deflection (AD) problem formulation . . . 15
  - 3.4.2 Maximum angle constraints (QP) . . . 17
  - 3.4.3 Maximum angle constraint (SOCP) . . . 17
  - 3.4.4 Deflection penalization . . . 18
  - 3.4.5 Normalization . . . 18
- 3.5 Bilinear Problem definition . . . 19
- 3.6 Linearized method . . . 19
  - 3.6.1 Linearized problem formulation . . . 20
- 3.7 SOCP - A Force approach . . . 23
  - 3.7.1 SOCP Problem formulation . . . 23
- 3.8 Rate constraints . . . 26
- 3.9 Faults . . . 27
  - 3.9.1 Loss of thrust . . . 27
  - 3.9.2 Jamming TVC . . . 27
  - 3.9.3 Loss of power of TVC . . . 28
- 3.10 Mixed Integer Second Order Cone Programming (MISOCP) formulation - including the Reaction Control System (RCS) and Aerodynamic fins . . . 28
- 3.11 Dynamic parameter . . . 28

4 Results 30
- 4.1 Simulation . . . 30

4.2 Embedded Conic Solver (ECOS) . . . 34
4.3 Solutions comparison of optimization formulations . . . 34
4.3.1 Error . . . 34
4.3.2 Actuation effort and behavior . . . 36
4.3.3 Runtime . . . 40
4.3.4 Faults . . . 41
4.3.5 Scalability . . . 46

5 Conclusion **48**

References **49**

A Pseudo-Inverse Allocation **52**

B Fault recovery graphs **54**
B.1 Loss of thrust - 80% Engine 5 . . . 54
B.2 Total Loss of Thrust Engine 5 . . . 56
B.3 Jamming TVC Engine 4 . . . 57
B.4 Loss of Power - maximum deflection Engine 4 . . . 59

C Mixed Integer Second Order Cone Programming (MISOCP) formulation - including the Reaction Control System (RCS) **61**

Nomenclature

Abbreviations

Abbreviation	Definition
AD	Angle-Deflection
CRE	Clustered Rocket Engines
ECOS	Embedded Conic Software
ESA	European Space Agency
FDIR	Fault Detection, Isolation and Reconfiguration
FTC-CRE	Fault-Tolerant Control of Clusters of Rocket Engines
GNC	Guidance, Navigation and Control
PI	Pseudo-Inverse
LP	Linear Programming
QP	Quadratic Programming
RCS	Reaction Control System
RCT	Reaction Control Thruster
SOCP	Second Order Cone Program
SQP	Sequential Quadratic Programming
TSTO	Two-stage-to-orbit
TVC	Thrust Vector Control

Symbols

Symbol	Definition	Unit
E	Set of engines	
$F_{i,cmd}$	Force command vector for actuator $i \in E$	[N]
$M_{i,cmd}$	Torque command vector for actuator $i \in E$	[Nm]
r_i	Location of the engine from the center of mass	[m]
R_i	Rotation system of engine $i \in E$ with reference to the axes of the thruster	
λ	Constant cost parameter values	
$\delta_{\{1,2\},i}$	Deflections of engine $i \in E$	[rad]
T_i	Total output thrust of engine $i \in E$	[N]
$T_{i,nom}$	The nominal thrust level of engine $i \in E$	[N]
ΔT_i	Differential thrust from nominal state ($T_i - T_{i,nom}$)	[N]
S	Set of faulty engines	

1 Introduction

In this section, we provide a comprehensive description of the types of problems this thesis aims to address. Section 1.1 discusses the motivation for applying optimization techniques to these problems, while Section 1.2 explains the context and application of the problem. In Section 1.3, the goals of the thesis are stated followed by a brief overview of the thesis structure in Section 1.4.

1.1. Motivation

A control system gives instructions to a system to regulate its behavior. So, the control commands direct the system to perform specific tasks or maintain desired states. The challenge of allocating these commands over mechanical devices is known as *control allocation*: how to assign control efforts to actuators in a way that meets the desired system behavior while minimizing cost functions such as energy use, actuator wear or deviation from the previous stage. Control systems have uses in multiple fields, like aerospace, automotive and marine applications. In modern control systems, the number of mechanical devices is increasing. This increasing number of devices in the systems causes redundancy, offering greater control authority and fault tolerance. However, it also introduces complexity in deciding how to distribute control demands efficiently among the available devices. As the complexity of the allocation schemes worsens as the number of actuators increases, there is an increasing need for systematic control allocation algorithms [20].

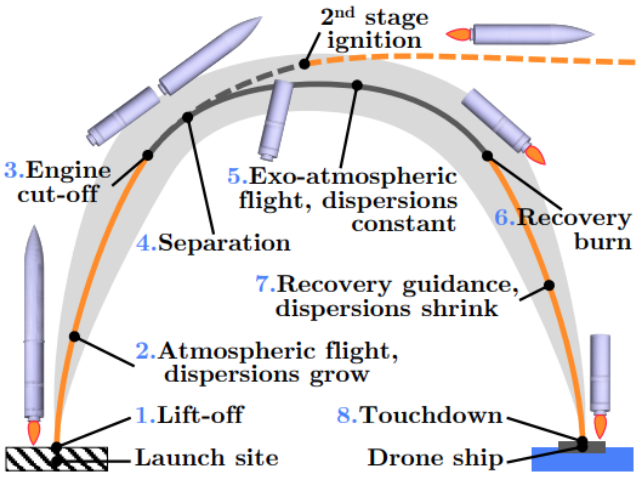
In addition, there is growing interest in control methods that are capable of reconfiguration in the event of actuator failures, enabling the system to adapt its control strategy without compromising mission objectives. As a result, there is a strong demand for control allocation algorithms capable of automatically distributing control commands across a large number of actuators, fully leveraging the available maneuvering authority, even under degraded conditions [4, 21, 28, 31].

Optimization in control allocation is essential for ensuring not only system performance but also operational efficiency, as it allows consideration of secondary objectives. As control surfaces and actuators have physical limits and dynamic constraints, an optimization approach to allocation allows systems to operate safely within bounds while adapting to real-time changes. With the growing interest in autonomous systems and sustainable operation, the role of optimization in control allocation offers a pathway to smarter, more adaptive control architectures in unfavorable conditions or faults.

1.2. Application of control allocation in multi-stage rockets

The application of the optimization model proposed in this thesis is on multi-stage rockets with clustered rocket engines (CRE). *Clustering* is when multiple engines are fired at the same time, thereby distributing the thrust and loads, instead of using a single engine. A *multi-stage* rocket is a launch vehicle with multiple stages stacked on top of each other, with every stage containing its own engines and propellant. When a stage runs out of propellant, it is able to separate so that the remaining rocket is smaller in size and mass, allowing easier acceleration of the rocket. More specifically it is applied

on a reusable two-stage-to-orbit (TSTO), where separation of two distinct stages is used in order to achieve orbital velocity and the first stage recovers back to earth. Illustration of the approach can be seen in Figure 1.1.
The ascent phase of the reusable launcher is the phase from lift-off to separation, illustrated by num-



(a) Downrange landing (DRL)

Figure 1.1: Stage separation of a recovery mission (Figure 5 of [33]).

bers 1-4 in Figure 1.1. This phase is controlled by adjusting the magnitude and direction of the thrust vector (TV) generated by the rocket engines [33], determined by the motion control system. As can be seen by number 6 in Figure 1.1, the engines are also used in the descent phase of the first stage to control magnitude and direction for a successful touchdown. Therefore, these phases are of particular interest to efficiently use the engines to create the desired magnitude and direction of forces. This thesis is therefore focused on these parts of the launch, where a simulation of the ascent phase is used for implementation.
To determine the direction of the launcher, it is important to consider the three principal rotations of the vehicle. These rotations occur around the three body-fixed axes and are generally referred to as *pitch, yaw,* and *roll*. An illustration of the rotations is given in Figure 1.2, where each rotation describes a specific change in orientation:

- *Roll* refers to rotation about the longitudinal z-axis (front-to-back of the vehicle), causing the launcher to spin around its own axis.
- *Yaw* refers to rotation about the vertical y-axis, turning the launcher left or right.
- *Pitch* refers to rotation about the lateral (side-to-side) x-axis, tilting the nose of the launcher up or down.

The mechanical devices that generate the mechanical forces and moments on the system are called *effectors*, like fins or gimbaled engines. *Actuators* are the electromechanical devices that control the magnitude and direction of the forces generated by such effectors by physically driving or moving the effectors [18]. For instance, in an engine, the engine itself is the effector that produces thrust to control the vehicle's motion, whereas a valve acts as the actuator by physically adjusting the flow of propellant. Therefore, the control commands are sent to the actuator, which in turn adjusts the state of the effectors. In this thesis, the combination of the effector and actuator is generally referred to solely as the actuator for simplicity. The decision on how the magnitude and direction commands by the control system are distributed over the available actuators, is where control allocation comes in place. In the following sections the available actuators in the TSTO and the system architecture are introduced. Furthermore, possible faults during the missions will be discussed.

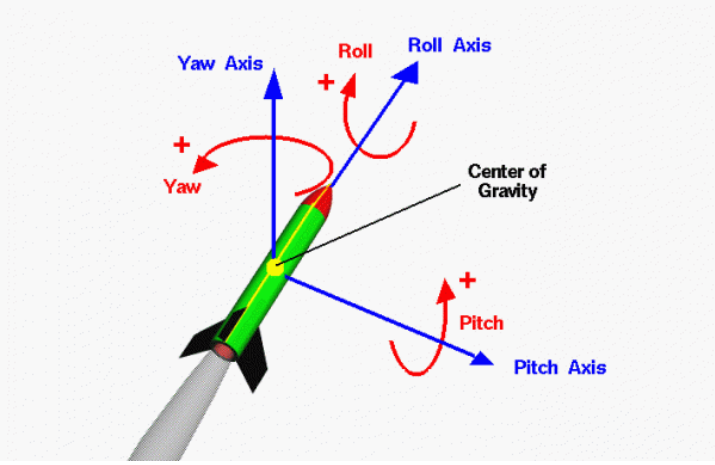


Figure 1.2: Pitch, Yaw and Roll rotations of a rocket with reference to the center of gravity (Figure of [8]).

1.2.1. Actuators

The multi-stage rocket is considered an *over-actuated* system because it is a system that has physical actuator redundancy [21]. Actuator redundancy means that the system has more actuators than the minimum required to control all of its degrees of freedom (DOFs). Here, DOFs represent the independent variables needed to define the system's state. In this thesis, the vehicle's motion is described by 6 DOF, existing of 3 translational and 3 rotational motions in 3D space. Forces on the rocket are primarily due to thrust from propulsion engines at the base of the rocket and moments by gimbaling these engines [3]. However, forces and torques can also be created by aerodynamic surfaces and the reaction control system (RCS). To generate time-varying mechanical forces and moments, the individual actuators control the magnitude and direction of these effectors. The location of the effectors are illustrated in Figure 1.3.

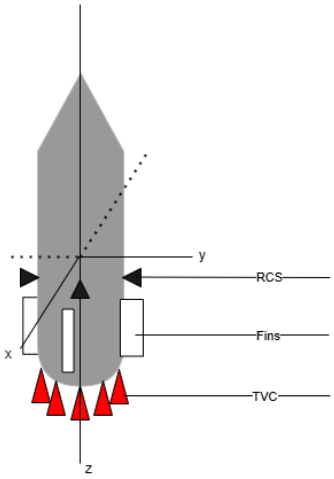


Figure 1.3: Illustration of the TVC, RCS and Grid Fins, which can all affect the direction and magnitude of the rocket.

Thrust Vector Control (TVC)

The Thrust Vector Control (TVC) scheme consists of liquid engines that are capable of rotating in two degrees of freedom (pitch and yaw) about a gimbal, as illustrated in Figure 1.4. An engine produces output thrust force, and the direction of this force is controlled by deflections. The deflections are created by a gimbal rotation mechanism. The gimbal rotation involves pivoting the entire engine and thrust

chamber, resulting in significant weight and mechanical complexity [3]. Nonetheless, TVC remains the standard actuation scheme for large rockets due to its effectiveness in controlling vehicle attitude's. Therefore, by the high effort and power required, minimizing the deflection of the thrust vector control (TVC) system is desirable.

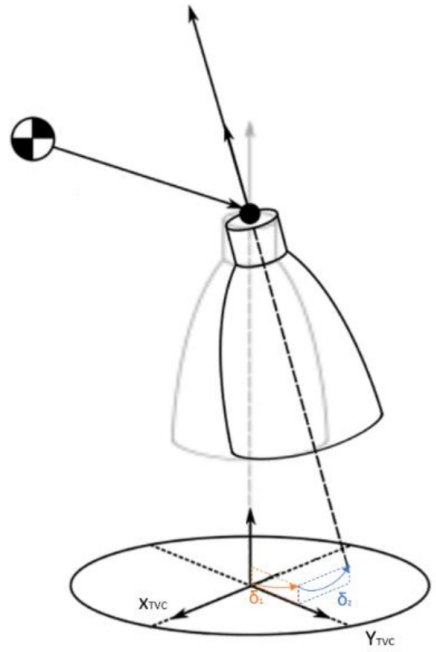


Figure 1.4: The motion of the angles along the axes of the nozzle, where δ_1 rotates around the x-axis (orange) and δ_2 rotates around the y- axis (blue). The gimbal mechanism is the point around which the nozzle is tilted, represented by the black circle at the base of the nozzle.

The engines among launcher manufacturers for a launch vehicle are commonly throttleable and re-ignitable [33]. This means it is capable of varying the amount of thrust created, the magnitude of the force vector, allowing extra freedom of actuator management. Torque is achieved due to the distance of the engine from the center of gravity where the force is applied.

Reaction Control System (RCS)

In contrary to the TVC, the Reaction Control Thrusters (RCT), which together form the RCS, are cold gas thrusters that do not have the ability to rotate around a gimbal. They are located on the sides of the launchers and have the ability to create force only in one direction. Furthermore, the amount of thrust is not changeable and the engine is either ON or OFF.

Aerodynamic Surfaces

Aerodynamic surfaces such as fins, rudders, or canards are used to steer the vehicle by exploiting airflow during atmospheric flight. Especially in the lower stages this can provide control authority with minimal propellant use. However, as the rocket ascends and atmospheric density decreases, its effect on control decreases. In this application in reusable launchers, the aerodynamic surfaces can also provide control authority during descent.

1.2.2. System architecture

To satisfy the specific trajectory and velocity to successfully deploy the payload of the rocket into the desired orbit, a certain force and torque is decided by the guidance and control system. Control allocation is the system which then distributes this command over the available actuators as illustrated in Figure 1.5. Therefore, the input of the control allocation is a command of force (F_{cmd}) and torque (M_{cmd}) and dynamic state variables like the center of mass (CoM), mass estimation and inertia.

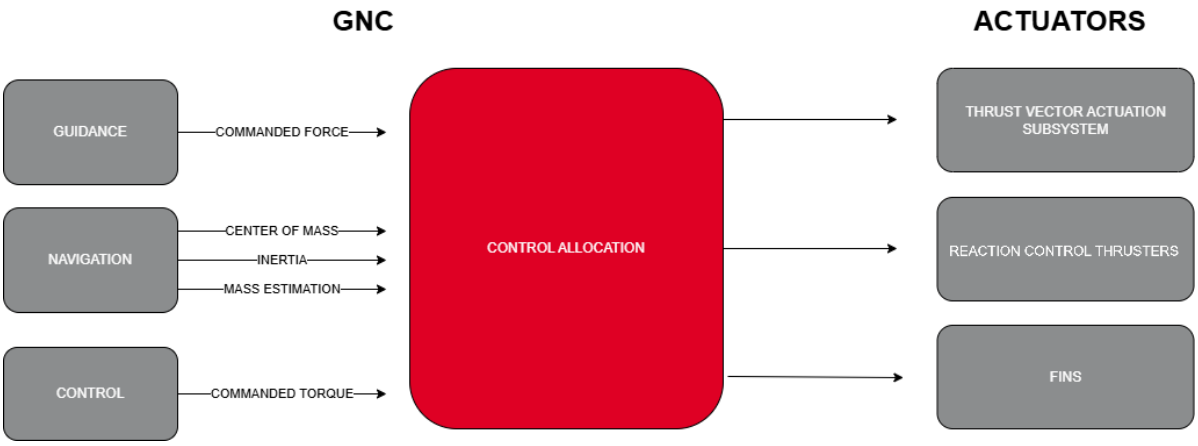


Figure 1.5: Control allocation in the Guidance, Navigation and Control (GNC) architecture. On the left we have the input for the control allocation, which is received from the GNC system. The control allocation is where the optimization formulation is implemented, such that the output are commands given to the actuators. The actuators, presented on the right, will then move to a state so that they create the desired commanded force and torque.

1.2.3. Faults

Various faults can compromise mission success and vehicle integrity in multi-stage launch vehicles. In the last 25 years about 50% of the faults were propulsion failures. Other issues were actuator failures or problems during the separation phase [24].
The objective of fault-tolerant control is to have the ability to reconfigure the actuator states in case of failures to continue satisfying the command using the non-faulty actuators. In this approach, we consider the following failures:

- **Thrust reduction/failure:** (Partial or full) loss of thrust acceleration or deviation of an engine or subset of engines;
- **Jamming TVC:** Failure in the TVC system by persistent off-nominal angles of deflection or a lock at a zero deflection for one rotation angle of an engine or a subset of engines;
- **Loss of power over TVC:** In case of loss of power of the TVC, an engine or subset of engines is free to move and generally has an uncontrolled deflection at its maximum range in one rotation direction. The thrust is still considered to be able to be controlled.

1.3. Goals

There tend to be an increasing number of actuators in modern control systems, allowing actuator redundancy. For instance modern launchers tend to use multiple smaller engines in clusters instead of using a few large engines, to improve thrust-to-weight ratios and as smaller engines are easier to manufacture. This also enables the system to be fault-tolerant and have more flexibility in mission demands. Therefore, the goal of this thesis is to formulate an optimization problem that can jointly compute the required actuator commands across multiple actuators. These actuators may differ in their contribution to the system. Furthermore, it is desired to be able to adapt to modern systems with an increasing amount of actuators. Hence, the goal is to construct a flexible and extensible optimization formulation to ensure scalability. The main objective of the formulation is to minimize the error between the actuator output and the control commands, ensuring that the commands are accurately followed and thereby maintaining reliable system performance. Furthermore, by the actuator redundancy of the over-actuated system, the goal is to include secondary objectives like minimizing effort and fuel consumption for a cost-effective access to space. Minimizing effort can be seen as reducing actuator activity to maintain stability while avoiding rapid gimbal or thrust movements that might strain the mechanical systems or cause actuator saturation.
Furthermore, the aim is to enable real-time reallocation of actuator commands in the event of faults, so that the control allocation algorithm can maintain reliable system performance despite failures. Minimizing the algorithm's computational cost is crucial to ensure it can function effectively in a real-time

environment.

1.4. Outline

This thesis starts with a review of control allocation strategies and fault-tolerant methods from the literature in Chapter 2, providing context for the approaches adopted in this work. Chapter 3 of the thesis focuses on three proposed control allocation formulations, starting with a Angle-Deflection (AD) problem formulation. After this, a more complex bilinear problem is introduced, introducing non-convexity. Both a linearized optimization formulation as well as a Second-Order Cone Programming (SOCP) formulation are proposed as convex formulations to handle this problem. The SOCP progresses to a mixed-integer second-order cone programming (MISOCP) model incorporating the RCS and fins. Furthermore, this chapter includes additional adaptions made to the model like maximum rate change of decision variables, adaption to faults and a dynamic parameter. Implementation details are discussed in Chapter 4, including the role of the embedded convex solver. Furthermore, this chapter contains the results of the use of the formulations, with emphasis on computational efficiency, error minimization and control effort. Furthermore, the results section presents the effectiveness in case of faults and large-scaled systems. Finally, the thesis concludes with a discussion of findings, practical considerations, and directions for future research in Chapter 5.

2

Control Allocation in Literature

Control allocation is an active area of research within the context of over-actuated systems [18]. In these systems, where the number of actuators exceeds the number of control effect dimensions, multiple actuator configurations can produce the same control effect. This redundancy expands the feasible set of actuator commands, instead of a single-point solution. Consequently, this expanded feasible set allows for the inclusion of secondary objectives in the control allocation process, such that the feasible set of solutions shrinks. As a result, control allocation is of interest in environments where actuator redundancy is common and there is a focus on improving fault tolerance, energy efficiency, and/or maneuverability. Hence, there are research applications in aerospace, autonomous vehicles, ships and underwater vehicles, or any area with a control command and a set of available actuators [32, 19, 14, 20]. All with the coinciding goal to assign control inputs to various actuators in order to achieve a desired performance or behavior.
The pseudoinverse and weighted pseudoinverse methods [26] are widely used as a traditional solution when solving the control allocation problem. However, formulating the control allocation problem as an optimization problem is also common in aerospace applications. The added benefit being that when sufficient control power exists, it allows for secondary objectives [20]. Furthermore, in contrast to pseudoinverse methods, optimization-based approaches have the ability to incorporate constraints. The optimization models give the ability to include saturation limits based on the physical or operational limitations of an actuator, such as maximum output thrust or deflection angles. Additionally, constraints can account for rate limits, ensuring that actuator states do not change too rapidly.

2.1. Optimization background

Mathematical optimization is the study of finding the best solution from a set of feasible options, either by minimizing or maximizing an objective function. Where the objective function is a mathematical formulation of the goal of the problem, which quantifies the quality or cost of different solutions [1]. An optimization problem typically consists of three main components: the objective function, a set of decision variables, and a set of constraints that define the feasible region. The constraints, which can be equalities or inequalities, restrict the values that the decision variables can take, based on physical limitations or system requirements. The standard form of an optimization problem is

$$\underset{x}{\text{minimize}} \quad f(x) \tag{2.1}$$

$$\text{subject to} \quad g_i(x) \leq 0, \qquad \text{for } i = 1, \cdots, m \tag{2.2}$$

$$h_j(x) = 0, \qquad \text{for } j = 1, \cdots, p \tag{2.3}$$

where $x \in \mathbb{R}^n$ is a vector of decision variables, $f(x)$ is the objective function and $g_i(x)$ and $h_j(x)$ are the constraints. Optimization problems can be broadly categorized based on the function types of the objective function and constraints. For instance, the problem is considered linear if $f(x), g_i(x), h_j(x)$ are all linear functions $\forall i$ and $\forall j$. The problem is defined as convex if $f(x), g_i(x)$ are convex $\forall i$ and

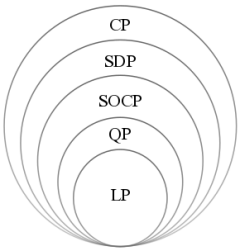


Figure 2.1: A hierarchy of convex optimization problems. (LP: linear program, QP: quadratic program, SOCP second-order cone program, SDP: semidefinite program, CP: cone program.)

$h_j(x)$ are affine functions $\forall j$. A function $f(x)$ is convex if $\forall x_1, x_2 \in \mathbb{R}^n$ and $\lambda \in [0,1]$, $(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$ holds. Affine functions satisfy the equality $(\lambda x_1 + (1-\lambda)x_2) = \lambda f(x_1) + (1-\lambda) f(x_2)$ for $\forall x_1, x_2 \in \mathbb{R}^n$ and $\lambda \in [0,1]$. Finally, an optimization problem is said to be continuous if the objective and constraint functions are continuous and the feasible region defined by the constraints is connected and does not have any gaps. The problem structure of a convex optimization problem guarantees that any local minimum is also a global minimum. As a result, convex problems benefit from robust algorithmic frameworks and can benefit from strong duality under certain conditions [5].
Building on this foundation, constrained optimization refers specifically to optimization problems that include constraints on the variables in the form of equalities and inequalities. Computationally efficient solution methods include interior-point methods [10], first-order methods [23], sequential quadratic programming (SQP), and barrier methods. These algorithms have been implemented in widely-used solvers, making constrained optimization practical across disciplines.
The need for real-time decision-making in embedded systems has led to the emergence of embedded optimization in control theory, a subfield of constrained optimization where numerical optimization algorithms are applied on embedded platforms for optimal decision making [15]. Because of the application on embedded platforms, these problems are generally constrained with limited memory and computational power. This has driven the development of lightweight solvers, implementing algorithms that are generally optimized for small problem sizes and fast solve times.

2.1.1. Second-Order Cone Programming (SOCP) formulation

The Second-order cone programming problem is a convex problem class that lies between linear or quadratic prgramming and semidefinite programming, as illustrated in Figure 2.1.

Definition 2.1.1 *The standard form of a Second-Order Cone Programming (SOCP) problem is*

$$\textit{minimize} \qquad f^T x \tag{2.4}$$

$$\textit{subject to} \qquad ||A_i x + b_i|| \leq c_i^T x + d_i, \qquad i = 1, \cdots, N, \tag{2.5}$$

where $x \in \mathbb{R}^n$ is the optimization variable to be determined, and the problem parameters are $f \in \mathbb{R}^n, A_i \in \mathbb{R}^{(n_i-1)\times n}, b_i \in \mathbb{R}^{n_i-1}, c_i \in \mathbb{R}^n$ and $d_i \in \mathbb{R}$. The norm is the Euclidean norm and the inequalities of line 2.5 are known as a second-order cone (SOC) constraints. The SOCP is a convex program [22].

2.1.2. Interior Point method (IPM)

The Interior Point Method is a common and powerful method when solving convex optimization problems, including Second-Order Cone Programs (SOCP). The method approaches an optimal solution by traversing the interior of a feasible set, following a path toward the optimal point instead of moving along the boundary as in active-set or simplex algorithms. Primal-dual interior point methods are particularly well-suited for conic optimization like SOCPs. Each iteration of the primal-dual IPM involves solving a system of linear equations derived from the perturbed Karush-Kuhn-Tucker (KKT) conditions. The self-dual and symmetric properties of the second-order cones enable efficient computation of the systems. Because the self-duality ensures the primal and dual constraints involve the same cone, allowing the same mathematical operations and properties to apply on both sides of the problem, leading to a converging duality gap. Furthermore, symmetry leads to simplified linear systems and well-conditioned

Newton steps, enabling cheap calculations for the KKT system. Polynomial-time convergence IPMs have been established with a high accuracy and a robust convergence and are known for handling large-scale problems [25]. Therefore, IPMs are widely applied in optimization solvers for SOCPs.

2.2. Optimization-based control allocation

Control allocation problems have been formulated as optimization problems in literature before. Specifically because an optimization problem formulation allows all available degrees of freedom to be utilized such that secondary objectives can be achieved [20]. These optimization problems have shown to be able to realistically solve the real-time control allocation in flight-control systems due to fast computation times [4]. There have been multiple different constrained optimization approaches in literature, such as direct allocation, error minimization, effort minimization and hierarchical dynamic control. The direct allocation formulation by Durham [12] optimizes the command satisfaction in a way that if the commands can not be fully satisfied, it keeps the allocated forces and moments in the same direction as the initial command. However, as this formulation does not take into account secondary objectives such as actuator usage cost or smoothness, this formulation is not used in the objectives of the proposed formulations in this thesis.

2.2.1. Error minimization

Another optimization formulation that is familiar in control allocation is a minimization of the error between the commanded output and the effected output. Let $u \in \mathbb{U}$ be the actuators such that $\mathbb{U}$ is a set bounded by actuator constraints, like minimum or maximum actuator states. Then the set of achievable control effects (forces/torques), $\mathbb{A}$, can be defined using the set of attainable actuator values so that

$$\mathbb{A} = \{ \ \tau \ \in \mathbb{R}^m : \exists u \in \mathbb{U}, Cu = \tau\},$$

where $C \in \mathbb{R}^{m \times p}$ is a matrix describing a linear relationship between the actuator values u and their output forces and torques. Then the error minimization problem is formulated in equation 2.6, where the difference between the commanded output τ_{cmd} and the output by the actuators from the actuator set is minimized.

$$\min\{||Cu - \tau_{cmd}|| \mid u \in \mathbb{U}\} \tag{2.6}$$

The norm in 2.6 has been set to l_1, l_2 and l_∞ norm in previous research, thereby creating both Linear Programming (LP) formulations as well as Quadratic Programming (QP) formulations [4, 29, 17]. These minimization problems make sure a feasible solution always exists, even when an unattainable command is asked. The LP forms have been solved by the simplex methods as well as interior point methods [7]. Although the simplex algorithm is a combinatorial approach which finds the optimal solution in a finite number of iterations, cycling often occurs when the optimization problem includes symmetric actuators in the control allocation problem [4]. Cycling in the Simplex algorithm is when the algorithm repeats sequences of the same feasible solutions and thereby failing to get to the optimal solution [7]. Although anti-cycling procedures exist, another drawback from using the LP is that the optimal solutions of the LP are always found at a vertex of the feasible set. Meaning that the solution will most often prefer to use a smaller amount of actuators than available. As this solution is generally less preferred in applications of actuator allocation in launchers, the proposed methods in this thesis are built upon a QP formulation of the error minimization. In this case, formulation 2.6 is applied with the Euclidean norm, adapted to a formulation including the physical and operational constraints in a launcher. In literature the QP has been efficiently solved using active set methods [16], interior-point methods [30] and an iterative fixed-point method [6]. Similarly as with the LP formulation, numerical challenges regarding cycling can arise.

2.2.2. Effort minimization

Other optimization-based formulations in literature are focused on minimizing the effort of the available actuators. For this formulation we define a most preferred state of actuation which needs least effort $u_0 \in \mathbb{U}$. Then the goal is to minimize the difference between the this state and the decision variables

$u \in \mathbb{U}$ [4], while the control effects satisfy the commands, such that

$$\min\{||u - u_0|| \mid u \in \mathbb{U}, Cu = \tau_{cmd}\}. \tag{2.7}$$

So in contrast to the error minimization approach, there is no feasible solution to the problem with an unattainable command where $\tau_{cmd} \notin \mathbb{A}$. Another way of incorporating minimal effort is to include it as a secondary optimization objective, after obtaining a minimum error solution. This is the approach of lexicographic goal programming (LGP) [11] [2], where conflicting goals are sorted in prioritized sequences of objectives. That is, one first optimizes the most important cost function, say c_1,

$$\min\{c_1 | c_1 = ||Cu - \tau_{cmd}||, u \in \mathbb{U}\}. \tag{2.8}$$

Then for the optimum c_1^*, this becomes a constraint in the second optimization problem,

$$\min\{c_2 | c_2 = ||u - u_0||, ||Cu - \tau_{cmd}|| \leq c_1^*, u \in \mathbb{U}\}, \tag{2.9}$$

Although this prioritizing problem has shown to reduce error [11] using an interior point method (IPM) solver, solving multiple optimization problems per instance is computatively expensive and thereby less attractive for real-time control.
Another application of effort minimization is a mixed optimization, combining error and effort minimization within the same objective function. To be able to prioritize within the mixed optimization, weighing parameters (λ_1, λ_2) can be introduced. Such that the objective becomes

$$\lambda_1 ||Cu - \tau_{cmd}|| + \lambda_2 ||u - u_0||. \tag{2.10}$$

This combined problem can be solved faster than the lexicographic problem and generally with better numerical properties [4]. Furthermore, the introduction of weighing parameters allows tuning of the objectives. This may be preferred in situations where the goals of the mission change between phases, such that the parametervalues are dependent on time ($\lambda_i(t)$).Therefore, the three formulations in this thesis include a multi-objective approach, where both effort and error are minimized while maintaining a solution for an unattainable command.
Also, Pascucci [27] considered control effort in a Second-Order Cone Problem (SOCP). However, his approach of penalizing control effort focuses on minimizing the different actuator efforts from the average actuator effort. So, enforcing evenly distributed efforts. Let N be the number of actuators, and λ_1, λ_2 weighing parameters, then the objective states

$$\min\{\lambda_1 ||u_i - \frac{1}{N} \sum_{j=1}^{N} u_i|| + \lambda_2 ||u_i - u_{0,i}||\}. \tag{2.11}$$

This cost function enforces an evenly distributed effort over the actuators. Although this may give a smoother solution and prevents actuator saturation, it also reduces freedom of non-uniform solutions. However, non-uniform solutions can be interesting for optimization in control allocation, especially in cases of reallocation after system failures. Therefore, this thesis proposes an SOCP formulation penalizing actuator effort without enforcing evenly distributed efforts.

2.2.3. Hierarchical dynamic control

When it is desired that the control commands are allocated over a subset of actuators (primary actuators), such that other actuators will only be used if there is an error between commands and the values produced by the primary actuators, it is said to have a hierarchy of control effectors [20]. Daisy chaining is an allocation approach that allocates the commands according to this hierarchy. It is a sequential control allocation where it first allocates the command over the primary control effectors which are constrained by saturation and rate limits, then if there is remaining command it passes to the next actuator in the chain. Therefore, the method is computationally light and easy to implement in real-time systems and naturally handles actuator limits. However, when the authority of control effectors are applied on multiple axes this method is likely to create conflicts. Furthermore, it only works for when a strict hierarchy is known as a priori and may struggle to re-allocate in case of faults. Instead, this thesis uses individual effort cost parameters in the multi-objective function. This makes it able to have hierarchical control only in times when it is needed, by increasing the cost parameters of a subset of actuators.

2.3. Fault models and types

At present, European launch vehicles rely primarily on a passive fault strategy based on hardware redundancy. However, active fault detection and isolation (FDI) techniques have been developed [13]. Methods and outcomes for guidance and control recovery strategies in launchers have been developed in 'Fault Tolerant Control of Clusters of Rocket Engines' (FTC-CRE) by the European Space Agency (ESA) and EuroGNC [28, 31]. This thesis proposes optimization formulations of which the structure of the optimization models do not need to change in case of faults. It uses existing constraints, such that faults are incorporated as input parameters that tighten these constraints. In case of the SOCP, the cone constraint is restricted to a line or point along a certain direction, enabling the integration of faults into the model.

3

Control Allocation formulations

In this chapter, we introduce three novel formulations for modelling control allocation. The control commands that are aimed to be satisfied by the control allocation model are the total force and total torque of the of the control system of the launcher. How the total force and total torque are formed by the state of the actuators is explained in Sections 3.1, 3.2 and 3.3. These sections are followed by the first formulation in Section 3.4, introduced as the angle-deflection (AD) problem. In the angle-deflection (AD) problem, we consider the situation where the nozzles are capable of deflection. In the other two models, the thrust created by the nozzles are also considered as throttleable. In the linearized problem (3.6) this decision on the thrust level is included by introducing the differential thrust as decision variable and including a linearized function to keep the formulation convex. The Second-Order Cone Programming (SOCP) problem (3.7) is a formulation which considers the individual forces of the actuators as decision variables. Then the actuation values for the deflection angles and thrust levels sent to the actuators are the unique solution of a pseudoinverse of the individual force vectors.
We compare the results from the models to results based on Pseudo-Inverse (PI) which is elaborated on in Appendix A. The PI method is particularly modelled for a 5 engine thruster. However, the proposed optimization formulations are generalized to any size of thruster engines. The comparisions of the formulations with the Pseudo-Inverse can be found in Chapter 4.

3.1. Force computation

To compute the forces in the three directions of the axes, the information needed is the amount of thrust created by the engine and the direction of the thrust. The direction of the thrust is caused by the angular deflections of the engine. As can be seen in Figure 3.1, a rotation created by the deflection of δ_1 (red in Figure 3.1) is applied in the yz-plane. However, it stays the same on the x-axis. So, it can be seen as a rotation around the x-axis of the nozzle. The δ_2 (blue in Figure 3.1) creates a rotation of the nozzle in the xz-plane about the y-axis. As the z-axis is defined as the longitudinal axis, a rotation around the z-axis is known as a *roll* rotation. The rotation created by δ_1 is also known as the *pitch* movement against the *xyz*-field, and the respective rotation matrix is defined as

$$R_{pitch} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\delta_1) & -\sin(\delta_1) \\ 0 & \sin(\delta_1) & \cos(\delta_1) \end{bmatrix}.$$

Similarly the rotation created by δ_2, the *yaw* rotation, is captured by the rotation matrix

$$R_{yaw} = \begin{bmatrix} \cos(\delta_2) & 0 & \sin(\delta_2) \\ 0 & 1 & 0 \\ \sin(\delta_2) & 0 & \cos(\delta_2) \end{bmatrix}.$$

Let $E \subset \mathbb{Z}$ be the number of engines in the TVC, so that $i \in E$ is an individual engine of the cluster. If the deflection angles δ_1 and δ_2 are known, the contributing force in the x, y and z direction can be

computed using the thrust value (T_i) of the engine. Let $F_{x,i}$,$F_{y,i}$, $F_{z,i}$ be the forces in x, y and z-direction generated by engine i. These forces can be calculated by

$$\begin{bmatrix} F_{x,i} \\ F_{y,i} \\ F_{z,i} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\delta_{1,i}) & -\sin(\delta_{1,i}) \\ 0 & \sin(\delta_{1,i}) & \cos(\delta_{1,i}) \end{bmatrix} \cdot \begin{bmatrix} \cos(\delta_{2,i}) & 0 & \sin(\delta_{2,i}) \\ 0 & 1 & 0 \\ \sin(\delta_{2,i}) & 0 & \cos(\delta_{2,i}) \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ T_i \end{bmatrix} \tag{3.1}$$

$$= \begin{bmatrix} \sin(\delta_{i,2}) \\ -\sin(\delta_{i,1})\cos(\delta_{i,2}) \\ \cos(\delta_{i,1})\cos(\delta_{i,2}) \end{bmatrix} \cdot T_i, \forall i \in E. \tag{3.2}$$

Now as the created forces by the nozzles are projected onto the *x,y,z*-axes, the total force of the vehicle

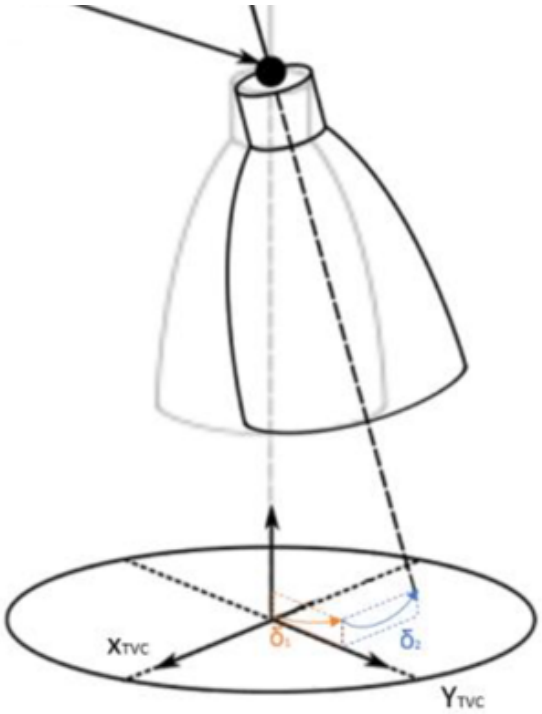


Figure 3.1: The motion of the angles along the axes of the nozzle.

can be calculated by the summation by the principle of superposition. Therefore, the total force can be calculated as

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \sum_{i \in E} \begin{bmatrix} F_{x,i} \\ F_{y,i} \\ F_{z,i} \end{bmatrix}, \tag{3.3}$$

The formulation of the forces is now a non-linear function since it contains trigonometric functions. However, as the deflection angles δ_1, δ_2 are expected to be small, we can use the first term of the Taylor expansion of the sine and cosine function as an approximation. In particular, we take the approximations $\cos(\delta) \approx 1, \sin(\delta) \approx \delta$. In case we assume the thrust level to be nominal and constant such that the forces are computed by

$$\begin{bmatrix} F_{x,i} \\ F_{y,i} \\ F_{z,i} \end{bmatrix} = \begin{bmatrix} sin(\delta_{i,2}) \\ -sin(\delta_{i,1})cos(\delta_{i,2}) \\ cos(\delta_{i,1})cos(\delta_{i,2}) \end{bmatrix} \cdot T_i \tag{3.4}$$

$$\overset{by\ Taylor\ exp.}{\approx} \begin{bmatrix} \delta_{i,2} \\ -\delta_{i,1} \\ 1 \end{bmatrix} \cdot T_i, \forall i \in E \tag{3.5}$$

3.2. Rotation Matrix

The axes of the nozzle along which the angles of rotation move, as in Figure 3.1, are not always aligned with the axes of the launcher, as illustrated in Figure 3.2. Therefore, a rotation matrix is introduced so that the total force is properly calculated by aligning all axes of the actuators with that of the launcher. Individual engines can have a different rotation axes relative to the axes of the vehicle as can be seen in Figure 3.2. As the goal is to allow scalability, the rotation matrix is generalized such that individual rotation matrices $R_i \in SO(3)$ can be supplied as input, where $SO(3)$ is the rotation group of a 3-dimensional space such that $R \in SO(3) = \left\{R \in \mathbb{R}^{3\times 3} \;\middle|\; R^\top R = I_3,\ \det(R) = 1\right\}$. Therefore, the general presentation of the rotation matrix in the optimization problem is as follows,

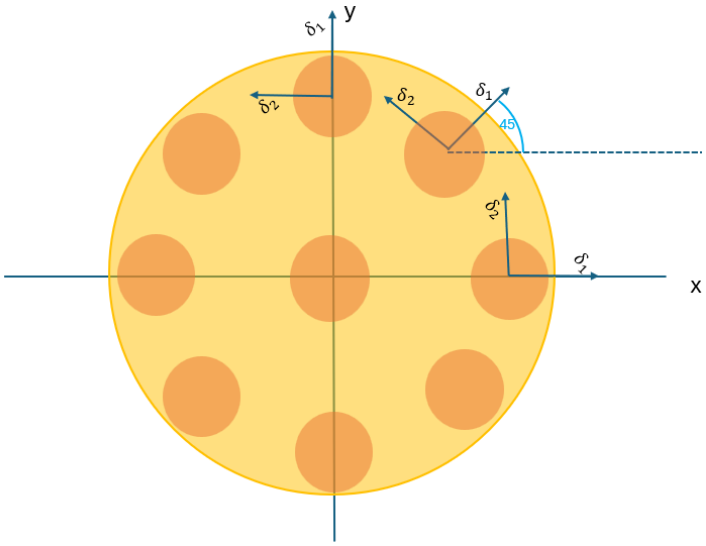


Figure 3.2: Axes of engines relative to the vehicles axes.

$$\overset{R_i = \text{rotation matrix of engine i}}{\begin{bmatrix} a_{1,i} & a_{2,i} & a_{3,i} \\ a_{4,i} & a_{5,i} & a_{6,i} \\ a_{7,i} & a_{8,i} & a_{9,i} \end{bmatrix}}.$$

Such that for a rotation of 45 degrees rotation in the xy field as in the top right nozzle of Figure 3.2, results in the following rotation matrix,

$$\overset{R_z\ rotation\ (45^\circ)}{\begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}}.$$

Then the force computation of an individual engine thrust level times the rotation, times the vector function of of the deflection of the engine:

$$\begin{bmatrix} F_{x,i} \\ F_{y,i} \\ F_{z,i} \end{bmatrix} = T_i \cdot \overset{R_i = \text{rotation matrix of engine i}}{\begin{bmatrix} a_{1,i} & a_{2,i} & a_{3,i} \\ a_{4,i} & a_{5,i} & a_{6,i} \\ a_{7,i} & a_{8,i} & a_{9,i} \end{bmatrix}} \cdot \begin{bmatrix} sin(\delta_{i,2}) \\ -sin(\delta_{i,1})cos(\delta_{i,2}) \\ cos(\delta_{i,1})cos(\delta_{i,2}) \end{bmatrix} \tag{3.6}$$

$$\approx T_i \cdot R_i \cdot \begin{bmatrix} \delta_{i,2} \\ -\delta_{i,1} \\ 1 \end{bmatrix}, \forall i \in E, \tag{3.7}$$

by the Taylor approximation as in 3.1.

3.3. Torque computation

The torque created by the actuators can be seen as a twist around an axis. This twist is created by force applied to a place on the object distanced from the center of mass. Therefore, the location of the actuators on the vehicle with reference to the center of mass is needed for the computation of the applied torque. We define $r_{x,i}$ the location of actuator i on the x-axis and r_{yi}, r_{zi} on the $y-$ and $z-$axis respectively. Then the torque produced by each engine is computed using the body-axis engine forces as in Equation 3.7 and its location on the rocket,

$$\begin{bmatrix} M_{x,i} \\ M_{y,i} \\ M_{z,i} \end{bmatrix} = \begin{bmatrix} 0 & -r_{z,i} & r_{y,i} \\ r_{z,i} & 0 & -r_{x,i} \\ -r_{y,i} & r_{x,i} & 0 \end{bmatrix} \cdot \begin{bmatrix} F_{x,i} \\ F_{y,i} \\ F_{z,i} \end{bmatrix} = \begin{bmatrix} r_{y,i}F_{z,i} - r_{z,i}F_{y,i} \\ r_{z,i}F_{x,i} - r_{x,i}F_{z,i} \\ r_{x,i}F_{y,i} - r_{y,i}F_{x,i} \end{bmatrix} \tag{3.8}$$

Later the location matrix $\begin{bmatrix} 0 & -r_{z,i} & r_{y,i} \\ r_{z,i} & 0 & -r_{x,i} \\ -r_{y,i} & r_{x,i} & 0 \end{bmatrix}$ is referred to as $r_i \in \mathbb{R}^{3\times 3}$ for simplicity.

The total output torque of the system including all actuators is defined by

$$\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = \sum_{i \in E} \begin{bmatrix} M_{x,i} \\ M_{y,i} \\ M_{z,i} \end{bmatrix} \tag{3.9}$$

by the superposition principle.

3.4. Angle-Deflection problem

The Angle-Deflection (AD) problem takes the freedom of changing the angles of the Thrust Vector Control (TVC, 1.2.1) to satisfy commanded torque and force. However, the thrust created by the individual engines is considered constant. Therefore, the thrust value of the engines are considered nominal. *Nominal thrust* is the most efficient thrust level, energy to force-wise, and is therefore the most desired thrust for an engine. The main objective is to control the direction and magnitude of the rocket by satisfying torque and force. However, to provide a solution although a command is unattainable, the formulation is based on the Error minimization approach as in Section 2.2.1, instead of using strict constraints. Furthermore, this formulation allows the introduction of weight parameters such that objectives in the cost function are adjustable over time. As all actuators are homogeneous, only considering the TVC, there is no hierarchy in actuators.

So let $M_x^{cmd} \in \mathbb{R}, M_y^{cmd} \in \mathbb{R}$ be the commanded torque around the x and y-axis. As it is desired to have no roll around the $z-$axis, M_z^{cmd} is set to 0. Furthermore, it is favored not to have any lateral forces, so we set $F_x^{cmd} = F_y^{cmd} = 0$, while for $F_z^{cmd} \in \mathbb{R}$ there is a specific command to reach orbit. Then the error minimization approach has, as objective function,

$$\min \quad \lambda_t^1 \left\| \begin{matrix} M_x^{cmd} - M_x \\ M_y^{cmd} - M_y \\ 0 - M_z \end{matrix} \right\|_2 + \lambda_t^2 \left\| \begin{matrix} 0 - F_x \\ 0 - F_y \\ F_z^{cmd} - F_z \end{matrix} \right\|_2, \tag{3.10}$$

for M_x, M_y, M_z the output torque in the three directions and F_x, F_y, F_z the output force. Where λ_t^1 and λ_t^2 are tunable time-dependent parameters. These parameters are used to be able to change the contribution of the differences from the command to the objective, thereby giving freedom to prioritize one command over another.

3.4.1. Angle-Deflection (AD) problem formulation

The following input parameters and decision variables are included in the AD formulation.

Input
Let

$E \subset \mathbb{Z} =$ set of available engines in the TVC
$T_{nom,i} \in \mathbb{R}^+ =$ the nominal thrust of engine i (in N)
$M^{cmd} \in \mathbb{R}^3 =$ a vector of torque commands by the control system in the x, y and $z-$direction (in Nm)
$F^{cmd} \in \mathbb{R}^3 =$ a vector of force commands by the control system in the x, y and $z-$direction (in N)
$R_i \in SO(3) =$ the rotation system of engine i in comparison with the axes of the thruster (cartesian coordinates)
$r_i \in \mathbb{R}^3 =$ the location of engine i with reference to the center of mass (in m)

Decision Variables
Let

$\delta_{1,i}$ (= deflection angle δ_1 for engine i (in radians)) $\in [\frac{-10\pi}{180}, \frac{10\pi}{180}]$

$\delta_{2,i}$ (= deflection angle δ_2 for engine i (in radians)) $\in [\frac{-10\pi}{180}, \frac{10\pi}{180}]$

Auxiliary Variables
$F_{x,i} =$ output force along the x-axis by engine i
$F_{y,i} =$ output force along the y-axis by engine i
$F_{z,i} =$ longitudinal output force along the z-axis by engine i
F_x = resultant force by all engines along the x-axis
F_y = resultant force by all engines along the y-axis
F_z = resultant longitudinal force by all engines along the z-axis
$M_{x,i} =$ output torque around the x-axis by engine i
$M_{y,i} =$ output torque around the y-axis by engine i
$M_{z,i} =$ output roll torque around the z-axis by engine i
M_x = total torque by all engines along the x-axis
M_y = total torque by all engines along the y-axis
M_z = total roll torque by all engines along the z-axis

Objective
$$\min \quad \lambda_t^1 \left\| \begin{matrix} M_x^{cmd} - M_x \\ M_y^{cmd} - M_y \\ 0 - M_z \end{matrix} \right\|_2 + \lambda_t^2 \left\| \begin{matrix} 0 - F_x \\ 0 - F_y \\ F_z^{cmd} - F_z \end{matrix} \right\|_2$$

Constraints
s.t.

$$\begin{bmatrix} F_{x,i} \\ F_{y,i} \\ F_{z,i} \end{bmatrix} = T_i \cdot R_i \cdot \begin{bmatrix} \delta_{2,i} \\ -\delta_{1,i} \\ 1 \end{bmatrix}, \forall i \in E, \tag{3.11}$$

$$\begin{bmatrix} M_{x,i} \\ M_{y,i} \\ M_{z,i} \end{bmatrix} = r_i \cdot \begin{bmatrix} F_{x,i} \\ F_{y,i} \\ F_{z,i} \end{bmatrix}, \forall i \in E, \tag{3.12}$$

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \sum_{i \in E} \begin{bmatrix} F_{x,i} \\ F_{y,i} \\ F_{z,i} \end{bmatrix} \tag{3.13}$$

$$\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = \sum_{i \in E} \begin{bmatrix} M_{x,i} \\ M_{y,i} \\ M_{z,i} \end{bmatrix} \tag{3.14}$$

3.4.2. Maximum angle constraints (QP)

As the nozzles have the ability to deflect to a maximum degree of 10 degrees, constraints have to be incorporated in the optimization problem such that the solutions are feasible regarding these physical constraints. Furthermore, in the launcher system to which the formulation is applied, the middle engine (engine 1) is not deflectable. Therefore, as constraint, these are set to 0 deflection.
As the the maximum angular deflection is 10 degrees of the TVC in the system to which the formulation is applied, the constraints are as follows

$$\delta_{1,1} = 0 \tag{3.15}$$

$$\delta_{2,1} = 0 \tag{3.16}$$

$$-10\frac{\pi}{180} \leq \delta_{j,i} \leq 10\frac{\pi}{180}, \text{for } j \in \{1,2\} \text{ and } i \in E \backslash 1 \tag{3.17}$$

3.4.3. Maximum angle constraint (SOCP)

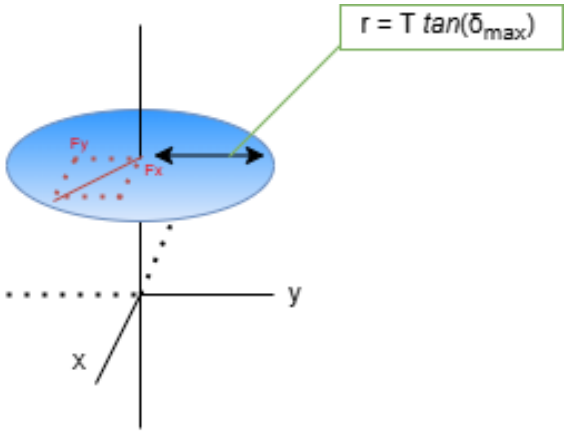


Figure 3.3: The circle or attainable values of $F_{x,i}, F_{y,i}$, bounded by the radius formed by the computation of output thrust and maximum deflection.

The deflection of the nozzle as pictured in Figure 3.1 is created by a pushing arm. Therefore, the maximum angle of deflection is the same in every direction of angulation, which is 10 degrees of rotation. Hence, in cases where the maximum deflection in both directions of the nozzles are equal, we can create a constraint on the maximum angulation by constructing a circle of attainable deflections by using the formula of lateral forces in case of deflections. This is illustrated in Figure 3.3. It can be seen that the norm of the forces acting on the x and y-axis are smaller than the boundary of the force and maximum possible angle. This circle that is created by using the maximum angle can be seen as the

attainable set of forces created by a feasible set of angles.

$$\sqrt{|F_{x,i}|^2 + |F_{y,i}|^2} \leq T_{i,nom} \cdot \tan(\frac{10\pi}{180}) \tag{3.18}$$

Including this constraint in the optimization problem instead of the original linear constraint of the maximum angle creates a second order cone constraint. Thereby making it a Second-Order Cone Problem (SOCP).

3.4.4. Deflection penalization

Deflection of the nozzles has a negative effect on the wear and tear of the actuator, as it consists of an electromechanical device that converts electrical energy into mechanical motion to displace the nozzle. Furthermore, any deflection of the nozzle results in a lateral force that does not contribute to the longitudinal motion of the rocket. These are necessary to steer the rocket and counteract on external disturbances. However, they must be minimized as a secondary objective given that mutual cancellation of thrusters often result in energy losses. So for a sustainable and cost-effective launch, the secondary objective is to minimize the deflections made. This can be included in the objective function by taking the norm of all deflections of all nozzles in both directions, such that the objective function becomes

$$\min \quad \lambda_t^1 \left\| \begin{matrix} M_x^{cmd} - M_x \\ M_y^{cmd} - M_y \\ 0 - M_z \end{matrix} \right\|_2 + \lambda_t^2 \left\| \begin{matrix} 0 - F_x \\ 0 - F_y \\ F_z^{cmd} - F_z \end{matrix} \right\|_2 + \lambda_t^3 \left\| \begin{matrix} \delta_{1,1} \\ \delta_{2,1} \\ \vdots \\ \delta_{1,i} \\ \delta_{2,i} \end{matrix} \right\|_2 \text{, for } i \in E$$

3.4.5. Normalization

When incorporating the deflections into the objective function, the different individual costs contribute to the full objective in different units. Therefore, the individual costs must be normalized, so the individual costs contribute proportionally allowing a balanced aggregated objective function despite the different units and scales. Therefore, the individual costs are divided by its maximum states.
For the torque, the normalization term is defined by the commanded torque $||M^{cmd}||$. To make sure the problem still works if there is no commanded torque such that $||M^{cmd}|| = 0$, a relatively small ϵ is added to the denominator of the fraction. The normalization of the lateral forces is defined by the lateral forces that exist when there is a maximum deflection, $\left\| \begin{matrix} \sum_{i \in E} F_{x,i}(T_{i,nom}, \delta_{1,i}^{max}, \delta_{2,i}^{max}) \\ \sum_{i \in E} F_{y,i}(T_{i,nom}, \delta_{1,i}^{max}, \delta_{2,i}^{max}) \end{matrix} \right\|_2$, such that this number is small. The longitudinal force is penalized by the nominal state $||T_{nom}||_2$, where deflections are normalized by the maximum deflections, $\left\| \begin{matrix} \delta_{1,1}^{max} \\ \delta_{2,1}^{max} \\ \vdots \\ \delta_{1,i}^{max} \\ \delta_{2,i}^{max} \end{matrix} \right\|_2$, for a balanced contribution to the full objective function. Furthermore, the penalization parameters $\lambda_t^1, \lambda_t^2, \lambda_t^3$ can be utilized to tune the contribution of the costs to the total objective function.

3.5. Bilinear Problem definition

In the AD problem, the only actuation variables considered are the deflection angles of the engines of the TVC. However, generally the engines in a multi-stage rocket are throttleable, meaning it has the ability to vary the amount of thrust. Besides influencing the upward force, a varied thrust between individual engines can also create a torque when the change is non-symmetric with reference to the center of mass. This varied differential thrust between engines can be interesting when there is a higher command of torque. Therefore, it is of interest to take into account the difference of thrust from the nominal state as a decision variable in the optimization model as well. Including the new variable differential thrust, noted as $\Delta T_i \in \mathbb{R}$ (in N), the force of an individual engine i is computed as

$$\begin{bmatrix} F_{x,i} \\ F_{y,i} \\ F_{z,i} \end{bmatrix} = T_i \cdot R_i \cdot \begin{bmatrix} sin(\delta_{2,i}) \\ -sin(\delta_{1,i})cos(\delta_{2,i}) \\ cos(\delta_{1,i})cos(\delta_{2,i}) \end{bmatrix}, \text{where } T_i = T_{nom,i} + \Delta T_i, \tag{3.19}$$

where R_i is the rotation matrix, $T_{nom,i}$ the nominal trust and T_i the output thrust of engine i. $\delta_{1,i}$ and $\delta_{2,i}$ are the deflection angles as in previous formulation.
By the multiplication of the thrust and deflections, this now becomes a bilinear function and thereby causes non-convexity. To be able to use an existing on-board SOCP solver, two methods are proposed in this thesis to reformulate the non-convex problem to a convex formulation. The first suggestion is quite straightforward, where the bilinear function is linearized around the nominal state of the engines. Furthermore, another optimization approach is suggested, which uses the individual engine forces as decision variables instead of the previously defined decision variables of thrust and deflection angles. Then using the optimal individual engine forces from the optimization problem, the optimal ΔT_i and $\delta_{i,1}, \delta_{i,2}$ can be calculated with an inverse function. This inverse of the individual engine forces have a unique solution for the thrust and deflections. In this way the solution of torque is not needed to be approximated, enforcing accuracy and feasibility. However, solving SOCPs are generally slower than QPs, making it possibly less attractive for real-time applications like is desired in control allocation. These methods will be further explained and formulated in the following sections.

3.6. Linearized method

In case of the linearized method we consider the same optimization problem as the AD version. However, now we assume the thrust of individual engines to be able to throttle between 30% and 110% of its nominal thrust. Then, if function 3.19 (noted as f for simplicity) is linearized around the nominal state, which is considered as having T_{nom} thrust output and no deflections (s.t. $\cos(0) = 1, \sin(0) = 0$),a linear equation can be derived

$$\begin{bmatrix} F_{x,i} \\ F_{y,i} \\ F_{z,i} \end{bmatrix} = f(T_{nom} + \Delta T, \delta_{2,i}, \delta_{1,i}) \tag{3.20}$$

$$\overset{by\ Taylor\ exp.}{\approx} f(T_{nom}, 0, 0) + \frac{df(T_{nom}, 0, 0)}{dT_i}(\Delta T) + \frac{df(T_{nom}, 0, 0)}{d\delta_{i,1}}(\delta_{i,1}) + \frac{df(T_{nom}, 0, 0)}{d\delta_{i,2}}(\delta_{i,2}) \tag{3.21}$$

$$= T_{nom} \cdot R_i \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \Delta T \cdot R_i \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + T_{nom} \cdot R_i \cdot \begin{bmatrix} 0 \\ -\delta_{1,i} \\ 0 \end{bmatrix} + T_{nom} \cdot R_i \cdot \begin{bmatrix} \delta_{2,i} \\ 0 \\ 1 \end{bmatrix} \tag{3.22}$$

$$= T_{nom} \cdot R_i \cdot \begin{bmatrix} \delta_{2,i} \\ -\delta_{1,i} \\ 1 \end{bmatrix} + \Delta T_i \cdot R_i \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.23}$$

such that this can be used as linear constraint for the computation of forces in the optimization problem. To validate the approximation of the linearized model, we study error difference of the linearized forces compared to the nonlinear forces for all physically available deflection rates. We visualize this in Figure 3.4 for a change of thrust from -100% and 100% of nominal thrust. In the figure possible values for the deflection angles are on the horizontal axes, and differential thrust is presented by $\frac{\Delta T}{T_{nom}}$ on the

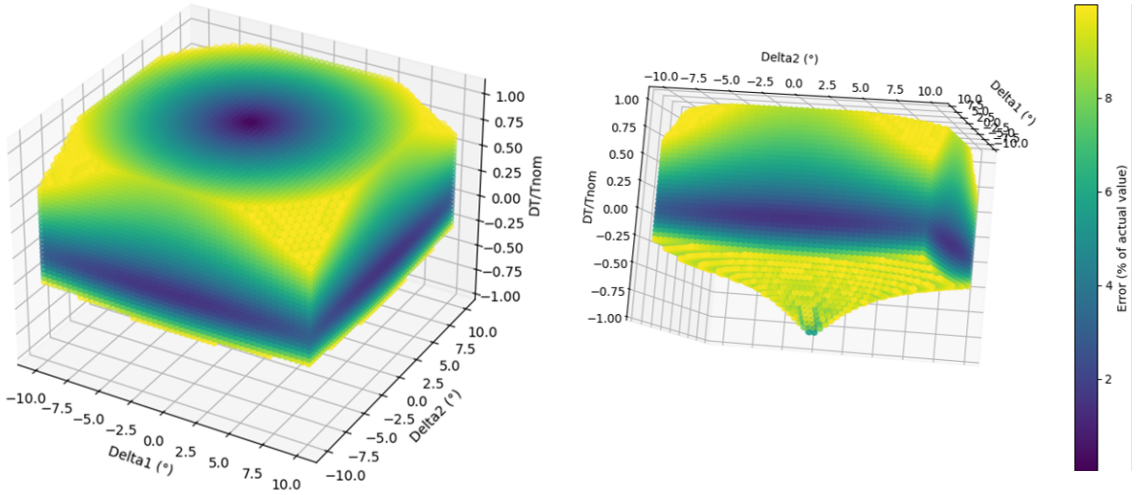


Figure 3.4: For $F \in \mathbb{R}^3$ calculated by the non-linear function (3.19) for the $\delta_1, \delta_2, \frac{\Delta T}{T_{nom}}$ on the axis and $F_{linear} \in \mathbb{R}^3$ (3.6) a linearized function for the same variables, this figure shows the error by $\frac{||F - F_{linear}||}{||F||}$ up to 10% of the non-linear computed vector F.

longitudinal axis. The error is shown by the different colours, up to 10% error. So let F be the solution vector of the non-linear computation of equation 3.19 and F_{linear} the solution vector computed by equation 3.23. Then the error is computed as $\frac{||F - F_{linear}||}{||F||}$ for the variables on the axes. So the error is the difference in computed force, divided by the actual force computed by the non-linear function. From the figure it can be seen that the error is generally smaller for small deflections and differential thrust. For any state of deflection angle, a force approximation is expected to be within the 10% error if ΔT is between a negative 25% of thrust and a positive 75% of nominal thrust. Therefore, in the optimization problem, the physical constraints to the differential thrust can be narrowed to stricter constraints by $-\Delta T \leq 0.25 \cdot T_{nom}$ for the model to predict the forces with a more accurate solution. The upper bound of ΔT, which is $0.1 \cdot T_{nom}$ is already within this range. Another way for a more accurate linearized formulation is to refer to the previous force reading, instead of the input nominal state. In that case a linearized function is expected to be closer to the actual value, as the next iteration may be close to the previous and therefore within the 10% error field. However, high differences in torque or force commands in small timesteps during certain phases or missions can contradict that assumption.

3.6.1. Linearized problem formulation

Including the ΔT as decision variable in the problem formulation, we have the following input and variable definitions:

Input
Let

$E \subset \mathbb{Z} =$ set of available engines in the TVC
$T_{nom,i} \in \mathbb{R}^+ =$ the nominal thrust of engine i (in N)
$M^{cmd} \in \mathbb{R}^3 =$ a vector of torque commands by the control system in the x, y and $z-$direction (in Nm)
$F^{cmd} \in \mathbb{R}^3 =$ a vector of force commands by the control system in the x, y and $z-$direction (in N)
$R_i \in SO(3) =$ the rotation system of engine i in comparison with the axes of the thruster (cartesian coordinates)
$r_i \in \mathbb{R}^3 =$ the location of engine i with reference to the center of mass (in m)

Decision Variables
Let

ΔT_i (= the change of thrust of engine i from its nominal thrust (in N)) $\in [-0.7 \cdot T_{nom,i}, 0.1 \cdot T_{nom,i}]$.

$\delta_{1,i}$ (= deflection angle δ_1 for engine i (in radian)) $\in [\frac{-10\pi}{180}, \frac{10\pi}{180}]$.

$\delta_{2,i}$ (= deflection angle δ_2 for engine i (in radian)) $\in [\frac{-10\pi}{180}, \frac{10\pi}{180}]$.

Auxiliary Variables
$F_{x,i} =$ output force along the x-axis by engine i
$F_{y,i} =$ output force along the y-axis by engine i
$F_{z,i} =$ longitudinal output force along the z-axis by engine i
F_x = resultant force by all engines along the x-axis
F_y = resultant force by all engines along the y-axis
F_z = resultant longitudinal force by all engines along the z-axis
$M_{x,i} =$ output torque around the x-axis by engine i
$M_{y,i} =$ output torque around the y-axis by engine i
$M_{z,i} =$ output roll torque around the z-axis by engine i
M_x = total torque by all engines along the x-axis
M_y = total torque by all engines along the y-axis
M_z = total roll torque by all engines along the z-axis

Objective

$$\min \quad \lambda_t^1 \frac{\left\| \begin{matrix} M_x^{cmd} - M_x \\ M_y^{cmd} - M_y \\ 0 - M_z \end{matrix} \right\|_2}{\left\| M^{cmd} \right\|_2} + \lambda_t^2 \frac{\left\| \begin{matrix} 0 - F_x \\ 0 - F_y \end{matrix} \right\|_2}{\left\| \begin{matrix} F_x^{max} \\ F_y^{max} \end{matrix} \right\|_2} + \lambda_t^{22} \frac{\left\| N \cdot T_{nom} - F_z \right\|_2}{\left\| N \cdot T_{nom} \right\|_2} + \lambda_t^3 \frac{\left\| \begin{matrix} \delta_{1,1} \\ \delta_{1,2} \\ \vdots \\ \delta_{n,2} \end{matrix} \right\|_2}{\left\| \begin{matrix} \delta_{1,1}^{max} \\ \delta_{1,2}^{max} \\ \vdots \\ \delta_{n,2}^{max} \end{matrix} \right\|_2} + \lambda_t^4 \sum_{i \in E} \frac{||\Delta T_i||_2}{||T_{i,nom}||_2} \tag{3.24}$$

Constraints

$$\begin{bmatrix} F_{x,i} \\ F_{y,i} \\ F_{z,i} \end{bmatrix} \overset{by\ Taylor\ exp.}{\approx} T_{nom} \cdot R_i \cdot \begin{bmatrix} \delta_{2,i} \\ -\delta_{1,i} \\ 1 \end{bmatrix} + \Delta T_i \cdot R_i \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \forall i \in E \tag{3.25}$$

$$\begin{bmatrix} M_{x,i} \\ M_{y,i} \\ M_{z,i} \end{bmatrix} = \begin{bmatrix} 0 & -r_{z,i} & r_{y,i} \\ r_{z,i} & 0 & -r_{x,i} \\ -r_{y,i} & r_{x,i} & 0 \end{bmatrix} \cdot \begin{bmatrix} F_{x,i} \\ F_{y,i} \\ F_{z,i} \end{bmatrix}, \forall i \in E \tag{3.26}$$

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \sum_{i \in E} \begin{bmatrix} F_{x,i} \\ F_{y,i} \\ F_{z,i} \end{bmatrix} \tag{3.27}$$

$$\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = \sum_{i \in E} \begin{bmatrix} M_{x,i} \\ M_{y,i} \\ M_{z,i} \end{bmatrix} \tag{3.28}$$

$$-10\frac{\pi}{180} \leq \delta_{j,i} \leq 10\frac{\pi}{180}, \text{for } j \in \{1,2\} \text{ and } i \in E \tag{3.29}$$

$$\Delta T_i \leq 0.1 \cdot T_{nom,i} \text{ for } i \in E \tag{3.30}$$

$$-\Delta T_i \leq 0.7 \cdot T_{nom,i} \text{ for } i \in E \tag{3.31}$$

Where 3.25 is the linearized constraint derived as equation 3.6. Constraint 3.26, 3.27, 3.28, 3.29 are as in the AD model. The first term of the objective function (3.6.1) corresponds to the normalized difference in torque, the second term is the normalized difference in lateral forces and the third term is the normalized longitudinal force as in the AD model. The fourth term corresponds to the normalized angle deflections and the last term the normalized differential thrust. The maximum amount of differential thrust is defined by constraint 3.30 as the throttleable engines can go up to 110% of its nominal state. The lowerbound of the engine is defined by 3.31 as 30% of the nominal thrust is the minimum required thrust if the engine is on. However, constraint 3.31 can be replaced by

$$-\Delta T_i \leq 0.25 \cdot T_{nom,i} \text{ for } i \in E \tag{3.32}$$

to assure a maximum of 10% error in the force computation with the non-linear value as shown in Figure 3.4.

3.7. SOCP - A Force approach

Another approach is not to decide on the $\Delta T, \delta_1, \delta_2$ as decision variables, but to optimize the distribution of needed forces and torque over the individual output forces of actuators. Then using an inverse function, this distribution of force can be distributed per actuator over either deflection or thrust. The thrust and deflection angles will be the unique solution to the inverse.
However, we need to make sure that the objective of fulfilling the right torque command as well as secondary objectives like minimizing deflection and thrust effort are still targeted. We are still able to do this by translating these targets into different formulations. For instance, minimizing deflections can be seen as minimizing lateral forces, as these lateral forces can only be created by deflections. Therefore, we can penalize $|F_{x,i}|^2 + |F_{y,i}|^2$ for every individual engines to target actuator effort by deflection.
Furthermore, the constraints on the maximum thrust per engine can still be incorporated by constraining the norm of the force vector. This creates a feasible set as an interior of a cone, where the radius is defined by the maximum angle and thrust value. However the minimum thrust constraint would make the problem non-convex, as it defines the exterior of the cone. Therefore, for the sake of convexity, this can be translated into a lowerbound of the longitudinal force F_z, assuming the lateral forces are small in proportion to the longitudinal force. However, for a more accurate approximation, we can also use the linearized technique as introduced in previous problem. Then an approximation of the norm of the force vector is computed, for each engine i, as

$$||F||_{i,approx} \approx ||F_0|| + \frac{1}{||F_0||} F_0^T (F - F_0) = \frac{F_0^T F}{||F_0||},$$

where $F_0 = \begin{bmatrix} 0 \\ 0 \\ T_{nom} \end{bmatrix}$. So it is linearized around the nominal state similarly to the previous formulation. However, this vector F_0 can also be adapted to the previous force vector solution for further refinement.

3.7.1. SOCP Problem formulation

Input
Let

$E \subset \mathbb{Z} =$ set of available engines in the TVC
$T_{nom,i} \in \mathbb{R}^+ =$ the nominal thrust of engine i (inN)
$M^{cmd} \in \mathbb{R}^3 =$ a vector of torque commands by the control system in the x, y and $z-$direction (in Nm)
$F^{cmd} \in \mathbb{R}^3 =$ a vector of force commands by the control system in the x, y and $z-$direction (in N)
$R_i \in SO(3) =$ the rotation system of engine i in comparison with the axes of the thruster (cartesian coordinates)
$r_i \in \mathbb{R}^3 =$ the location of engine i with reference to the center of mass (in m)

Decision Variables
$F_{v,i} \in \mathbb{R}, \forall i \in E, \forall v \in \{x, y, z\}$

Auxiliary Variables
$||F||_{i,approx} =$ an approximation of the total force magnitude (in N).

Objective

$$\min \quad \lambda_t^1 \left\| \begin{matrix} \sum_{i \in E} [r_i \cdot F_i]_x - M_{x,cmd} \\ \sum_{i \in E} [r_i \cdot F_i]_y - M_{y,cmd} \\ \sum_{i \in E} [r_i \cdot F_i]_z - M_{z,cmd} \end{matrix} \right\|_2 + \lambda_t^2 \left\| \begin{matrix} \sum_{i \in E} F_{x,i} - F_{x,cmd} \\ \sum_{i \in E} F_{y,i} - F_{y,cmd} \end{matrix} \right\|_2 + \lambda_t^{22} \left\| \sum_{i \in E} F_{z,i} - F_{z,cmd} \right\|_2$$
$$+ \lambda_t^3 \sum_{i \in E} \left\| \begin{matrix} F_{x,i} \\ F_{y,i} \end{matrix} \right\|_2 + \lambda_t^4 \sum_{i \in E} (||F||_{i,approx} - T_{nom,i})^2,$$

where $M_{z,cmd}$ and $F_{x,cmd}, F_{y,cmd}$ is set to 0 in our application to avoid roll torque and lateral forces, respectively.

Normalized Objective

$$\min \quad \lambda_t^1 \frac{\left\| \begin{matrix} \sum_{i\in E}[r_i \cdot F_i]_x - M_{x,cmd} \\ \sum_{i\in E}[r_i \cdot F_i]_y - M_{y,cmd} \\ \sum_{i\in E}[r_i \cdot F_i]_z - M_{z,cmd} \end{matrix} \right\|_2}{\left\| M^{cmd} + \epsilon \right\|_2} + \lambda_t^2 \frac{\left\| \begin{matrix} \sum_{i\in E} F_{x,i} \\ \sum_{i\in E} F_{y,i} \end{matrix} \right\|_2}{\left\| \begin{matrix} \sum_{i\in E} F_{x,i}^{max} \\ \sum_{i\in E} F_{y,i}^{max} \end{matrix} \right\|_2} + \lambda_t^{22} \frac{\left\| \sum_{i\in E} F_{z,i} - F_{z,cmd} \right\|_2}{\left\| F_{z,cmd} \right\|_2} + \lambda_t^3 \frac{\sum_{i\in E} \left\| \begin{matrix} F_{x,i} \\ F_{y,i} \end{matrix} \right\|_2}{|E| \cdot \left\| \begin{matrix} F_{x,i}^{max} \\ F_{y,i}^{max} \end{matrix} \right\|_2} +$$

$$\lambda_t^4 \frac{\sum_{i\in E}(||F||_{i,approx} - T_{nom,i})^2}{|E| \cdot T_{nom}},$$

The tunable parameters (λ_t) can adjust the priority of the following objectives:

$\lambda_t^1 =$ cost on the error between commanded and created torque over time (t)
$\lambda_t^2 =$ cost on the total lateral force in the x and y direction from the commanded lateral force over time (t)
$\lambda_t^{22} =$ cost on the difference between the needed and total output force in upward (z) direction over time (t)
$\lambda_t^3 =$ cost on individual lateral forces, so indirectly the deflection of every individual engine over time (t)
$\lambda_t^4 =$ cost on the differential thrust over time (t)

Constraints

$$\sqrt{|F_{x,i}|^2 + |F_{y,i}|^2} \leq F_{z,i} \cdot \tan(\frac{10\pi}{180}), \forall i \in E \tag{3.33}$$

$$||F_i|| \leq T_{max}, \forall i \in E \tag{3.34}$$

$$T_{min} \leq ||F||_{i,approx}, \forall i \in E \tag{3.35}$$

Where constraint 3.33 makes sure the deflections do not go beyond saturation limits, by creating an attainable conic set as illustrated in Figure 3.5. The lateral forces $F_{x,i}, F_{y,i}$ reflect the deflections, as these forces by the TVC can only be created by deflections. Furthermore, constraint (3.34) makes sure the magnitude of every engine is bounded by its maximum thrust. As bounding the magnitude by the minimum thrust would make the problem non-convex, we approximate the magnitude of the individual engines. This can be done by assuming the F_z value valid for small nozzle deflections. However, the linearized approximation of $||F||_{i,approx}$ can be used for a convex formulation as well, which is used in constraint (3.35).

Output

The solution of the SOCP optimization problem gives the optimal distribution of individual force vectors of the different actuators $F_{v,i}, \forall i \in E, \forall v \in \{x, y, z\}$. However, as presented by the system architecture, it is desired to deliver deflection and thrust actuator commands as output. Therefore, these values are calculated by the inverse of the following force function

$$\begin{bmatrix} F_{x,i} \\ F_{y,i} \\ F_{z,i} \end{bmatrix} = (T_{nom,i} + \Delta T_i) \cdot R_i \cdot \begin{bmatrix} sin(\delta_{2,i}) \\ -sin(\delta_{1,i})cos(\delta_{2,i}) \\ cos(\delta_{1,i})cos(\delta_{2,i}) \end{bmatrix}, \forall i \in E,$$

such that $(\Delta T_i, \delta_{1,i}, \delta_{2,i}) = f^{-1}(F_{v,i})$

ΔT_i = the change of Thrust of engine i with reference to the nominal thrust.

$\delta_{1,i}$ = deflection angle δ_1 for engine i

$\delta_{2,i}$ = deflection angle δ_2 for engine i

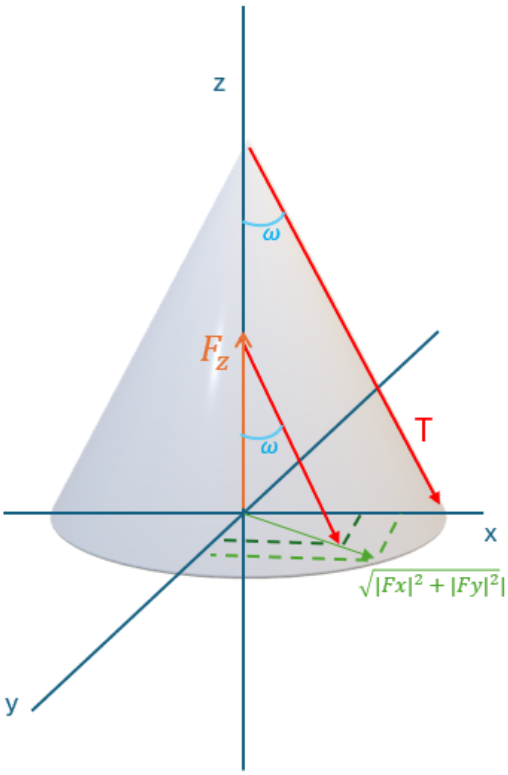


Figure 3.5: Illustration of attainable conic set of deflections and differential thrust of one engine.

3.8. Rate constraints

Another constraint that is added to the mentioned optimization problems are the rate constraints. The actuators have a maximum speed and acceleration in changing the nozzle deflection. Therefore, the following constraint can be added to the AD and linearized problem to enforce feasibility of the solutions of the problem

$$||\delta^{prev} - \delta|| \leq \Delta t \cdot \dot{\delta}, \tag{3.36}$$

where δ^{prev} is the deflection solution of the previous timestep in the online optimization problem. δ is the current decision variable on the deflection angle and Δt is the differential time between the timesteps. Then the difference between the deflections of consecutive timesteps is constrained by the rate change $\dot{\delta}$ in $[m/s]$. Furthermore, the differential in thrust also has a maximum rate of change. This rate is dependent on $\alpha \in [1, 10]$, a time constant of thruster dynamics, referring to the characteristic time it takes for the thruster output to respond to the change in input by actuator level. Generally α takes a value between 1 and 10 rad/s, where 1 means the valve opens more slowly and 10 means the valve opens more rapidly. A valve that can move more rapidly allows bigger difference in thrust over the same timespan. Discretized, this becomes $||T_i(k-1) - T_i(k)|| \leq \alpha \cdot \Delta t \cdot T_{nom}$ for timestep Δt and k being the current timestep. In our simulation in order to incorporate worse case solutions, α is set to 1.

However, as the deflection angles and differential thrust in the SOCP are not direct decision variables, we need to rewrite this constraint to include it in the problem. Instead of thrust, in this case we bound the difference in longitudinal force, such that $||F_{z,i}(k-1) - F_{z,i}|| \leq \alpha \cdot \Delta t \cdot T_{nom}$. However, in further research this may be done by using the magnitude approximation $||F_i||_{approx}$ introduced in the SOCP formulation. To bound the rate of deflection the same approach is used as by Pascucci [27], where we use the small angle assumption again, so applying the Taylor expansion we get $\tan(\delta) \approx \delta$, for the following approximation

$$||F_{xy,i}|| = F_{z,i} \cdot \tan(\frac{\delta \cdot \pi}{180}) \tag{3.37}$$

$$\approx F_{z,i} \cdot \frac{\delta \cdot \pi}{180}. \tag{3.38}$$

Then differentiating the equation, we get

$$||\dot{F_{xy,i}}|| = \dot{\delta} \cdot F_{z,i} + \delta \cdot \dot{F_{z,i}}, \tag{3.39}$$

by rearranging, the rate of change is defined by

$$\dot{\delta} = \frac{||\dot{F_{xy,i}}|| - \delta \cdot \dot{F_{z,i}}}{F_{z,i}} \tag{3.40}$$

which we know is bounded by $\dot{\delta_{max}}$. We assume that $F_{z,i} > 0$, such that we get

$$F_{z,i} \cdot \dot{\delta_{max}} \geq |||\dot{F_{xy,i}}|| - \delta \cdot \dot{F_{z,i}}| \tag{3.41}$$

Then bounding this by the maximum deflection and triangle inequality we get

$$||\dot{F_{xy,i}}|| + |\delta_{max} \cdot \dot{F_{z,i}}| \geq | \, ||\dot{F_{xy,i}}|| - \delta \cdot \dot{F_{z,i}}| \tag{3.42}$$

now for a discrete context $||\dot{F_{xy,i}}||$ can be approximated as $||\dot{F_{xy,i}}|| \approx \frac{||F_{xy,i}^{prev} - F_{xy,i}||}{\Delta t}$, such that applying the bound of the absolute values by Equation 3.42 to Equation 3.41, the discrete deflection rate can be bounded as follows.

$$||F_{xy,i}^{prev} - F_{xy,i}|| \leq (F_{z,i} \cdot \dot{\delta_{max}} - |\delta_{max} \cdot \dot{F_{z,i}}|) \cdot \Delta t, \tag{3.43}$$

can be added as a constraint to the SOCP formulation for maximum deflection rate.

3.9. Faults

As described in subsection 1.2.3, there are three types of faults considered: loss of thrust, TVC jamming and loss of power of the TVC. To ensure robustness, it is of great value to be able to use the same optimization model in case of faults, such that the problem automatically adapts and recovers in this situation. Instead of using separate models in case of failure, therefore the optimization formulation must be modelled in a way such that the problem is not changed, but the input parameters in case of faults can tighten the constraints.

3.9.1. Loss of thrust

For the situation where an engine experiences loss of thrust, let $S_1 \subset E$ be the faulty engines of this kind. In case of total loss of thrust, this means $T_{s_1} = 0$ for $s_1 \in S_1$. This means the engine is not able to create any force. In case the thrust gets stuck at a certain level, for instance at 50% of its nominal state, $T_{s_1} = 0.5 \cdot T_{f,nom}$.
As previously mentioned, it is desired to implement these bounds within the original problem instead of creating a new problem with these constraints. This loss of thrust can be implemented in the AD and linearized formulation as two inequality constraints, where the input parameters are normally the minimum and maximum thrust, such that

$$T_i \leq T_{i,max} \tag{3.44}$$
$$T_{i,min} \leq T_i \tag{3.45}$$

Then in case of fault, the inequalities change to the faulty state such that

$$T_s \leq T_{s,max} + \epsilon \tag{3.46}$$
$$T_{s,min} - \epsilon \leq T_s, \tag{3.47}$$

where $T_{s,max} = T_{s,min} = T_{s_1}$ and ϵ both equally small.
By constraining the ΔT as in constraints 3.31 and 3.32 in the linearized problem, the faults bound the decision variable in the following way

$$\Delta T_s \leq T_{s_1} - T_{s,nom} + \epsilon \tag{3.48}$$
$$-\Delta T_s \leq T_{s,nom} - T_{s_1} - \epsilon, \tag{3.49}$$

In case of the SOCP formulation, it is not possible to implement these constraints directly on the thrust value as this is not a decision variable of the problem. Therefore, the bound is implemented by adjusting constraints 3.35 and 3.36 to the faulty constraints, such that

$$||F_s|| \leq T_{s,max} \tag{3.50}$$
$$T_{s,min} \leq ||F_s||_{approx}, \tag{3.51}$$

where $T_{s,max} = T_{s,min} = T_{s_1}$.

3.9.2. Jamming TVC

Another possibility is that the nozzle is stuck to a certain deflection, such that $\delta_{f,1} = 0$ or $\delta_{f,2} = 0$ for zero deflection or $\delta_{f,1} = b_1$ or $\delta_{f,2} = b_2$ for nozzles stuck at a certain deflection $b_1, b_2 \in [-10\frac{\pi}{180}, 10\frac{\pi}{180}]$.
As can be seen in the SOCP formulation in section 3.7, there is a cone constraint to bound the maximum deflections. The general cone constraint in the problem is as follows

$$\sqrt{|F_{x,i}|^2 + |F_{y,i}|^2} \leq F_{z,i} \cdot \tan(\frac{10\pi}{180}), \forall i \in E \Leftrightarrow ||(I - dd^T)F_i|| \leq d^T F_i \cdot \tan(\alpha\frac{\pi}{180}), \forall i \in E, \tag{3.52}$$

where $d = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$, $\alpha = 10$ is the maximum angle of the cone and $F_i = \begin{bmatrix} F_{x,i} \\ F_{y,i} \\ F_{z,i} \end{bmatrix}$.
Now in case of a stuck deflection at b_1 and b_2, the d vector changes into the direction of the deflection

such that $d = \begin{bmatrix} \sin(b_2 \frac{\pi}{180}) \\ -\sin(b_1 \frac{\pi}{180}) \cos(b_2 \frac{\pi}{180}) \\ \cos(b_1 \frac{\pi}{180}) cos(b_2 \frac{\pi}{180}) \end{bmatrix}$ as in equation 3.2. So if the stuck angles b_1 and b_2 are given as input variables, the formulation can create d as a vector of constant values, which then determines the direction of the cone. As the nozzle is stuck, it is not able to deviate in deflections, therefore the cone becomes a line into one direction. In such cases, we set $\alpha = 0$, such that the constraint becomes a line of different magnitudes in the same direction.

3.9.3. Loss of power of TVC

Another type of failure is the total loss of power of the TVC. By thrust eccentricity and quasi-static acceleration the fault results in a maximum deflection of the faulty engine. The nozzle is seen as free in deflection, however, it generally aligns with the external force to a position corresponding to the maximum deflection. Therefore, as an example we construct inclusion of the fault in the same as the jamming TVC, but now having one of the deflections stuck at the maximum 10 degrees such that $b_1 = 10\frac{\pi}{180}, b_2 = 0$, implying $d = \begin{bmatrix} \sin(\frac{10\pi}{180}) \\ -\sin(\frac{10\pi}{180}) \\ \cos(\frac{10\pi}{180}) \end{bmatrix}$ and $\alpha = 0$. This could be done for b_2 containing the maximum deflection as well.

3.10. Mixed Integer Second Order Cone Programming (MISOCP) formulation - including the Reaction Control System (RCS) and Aerodynamic fins

The formulation of Equation (3.52) also allows to incorporate actuators that cannot deflect and are fixed at a certain rotation, for instance for engines in the RCS. Besides that, the constraint adaptation technique used for handling loss of thrust in subsection 3.9.1 can also be used in cases when the output force of an actuator is fixed. For example, for aerodynamic fins, whose output force is dependent of a function of external factors. Therefore, these constructions show that the formulations are able to adapt to different actuators. However, the discrete nature of the Reaction Control System requires constraints that are binary. Combined with existing constraints, this makes the problem Mixed-Integer, which is further discussed in Appendix C. Because the convex conic solver in the control system cannot handle mixed-integer problems, implementing this formulation is beyond the scope of this thesis. Nonetheless, the general approach of the formulation allows incorporation of the actuators without changing the optimization problem structure.

3.11. Dynamic parameter

Now that the linearized and SOCP formulation include the differential thrust as an actuator freedom, it also gives the freedom to create torque without deflection. As mentioned, generally the TVC deflections are preferred over the use of differential thrust. Both for the purpose of minimizing energy consumption and flexibility, as the deflections can generally change faster than the differential thrust. However, in some cases the opposite can be preferred, using differential thrust instead of deflections. For instance, when there is a long term command of constant torque. As it is desired to keep the freedom of deflections for smaller differences in commands and to minimize the wear and tear of the nozzles, a differential torque is preferred over nozzle deflections. Therefore, a dynamic parameter can be introduced, which penalizes the costs of deflections more if it is constantly used over a longer period.
To penalize this behaviour, we want λ_t^3 to be dependent on the previous deflections $\delta_{\{1,2\},i}^{prev}$. To allow information of history, without having to save all the previous solutions in the system, the following recursive algorithm is constructed:

$$\lambda_{t_{j,i}}^3 = \omega \cdot \lambda_{t-1_{j,i}}^3 + \delta_{j,i}(t-1) \tag{3.53}$$

where $0 < \omega < 1$, such that the influence of the deflection solutions of previous time steps decreases exponentially with time. Therefore, deflections closer to the current time step have more influence on the penalization term than time steps further away from the current timestep. A higher ω value provides

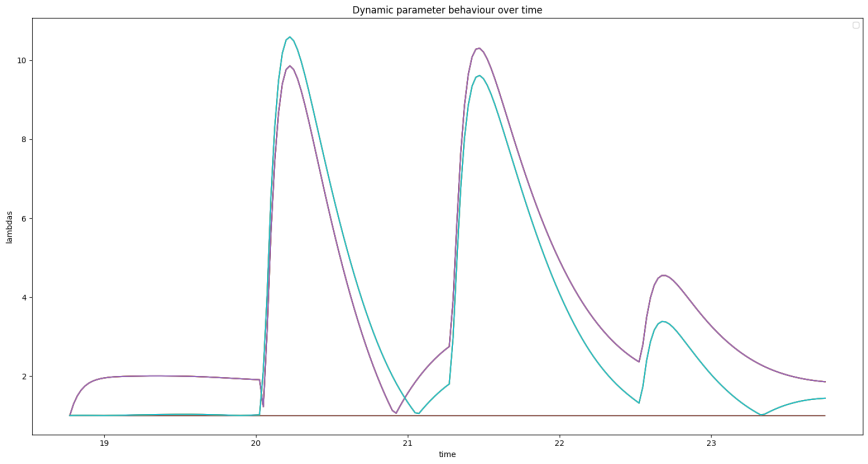


Figure 3.6: The dynamic parameter values $|\lambda^3_{j,i}|$ over time. The brown line at zero defines the values for Engine 1, which are 0 because this engine is considered non-deflectable in this simulation. As all other engines deflect the same way, the penalization for the other engines all behave the same. It can be seen that during periods when there is more deflection (20-21) the parameter increases, while it decreases again when there is less deflection.

a higher influence of previous λ, thereby previous deflections, where lower ω values provides higher influence of the last deflection position. The preference of ω value can be up to the user, however for the purpose of penalizing long-term constant deflections, a higher ω is advised. The only values required to be stored for the computation are the single constant values of λ_{k-1} and the deflection solutions of previous time step $\delta_{\{1,2\},i}(k-1)$. As the deflections can be both positive as negative, the cost penalization is defined by the absolute value, as we do want to keep the sign for computations of further timesteps $k+1$. This is as we do want to penalize constant deflections in the same direction, while avoiding penalizing deflections with different directions over time. This penalization discourages persistent deflections. So

$$\begin{bmatrix} |\lambda^3_{1,1}| \\ |\lambda^3_{2,1}| \\ \vdots \\ |\lambda^3_{1,|E|}| \\ |\lambda^3_{2,|E|}| \end{bmatrix}^T \cdot \begin{bmatrix} \delta_{1,1}(t) \\ \delta_{2,1}(t) \\ \vdots \\ \delta_{1,|E|}(t) \\ \delta_{2,|E|}(t) \end{bmatrix}, \tag{3.54}$$

where $\lambda_{j,i} = \lambda_0 + c(|\lambda_{t_{j,i}}|)$, such that λ_0 is a constant penalization and $c(x)$ a class κ function (i.e., c is a continuous function that is strictly increasing where $c(0)+0$).
Figure 3.6 illustrates the behavior of the dynamic parameter values, showing higher penalization for larger deflection. However, further research is needed to tune parameter ω so that the penalization effectively targets long-term, sustained deflections. To evaluate whether this approach successfully redistributes the commands in cases of long-term constant deflections, it should be applied to simulation data with long-term torque commands.

4

Results

In this chapter, the results of solving the different optimization formulations will be discussed. Section 4.1 first elaborates on the simulation setup and the data used to verify the results. It is followed by an explanation on the conic solver ECOS in Section 4.2, which is used to solve the optimization formulations. In Section 4.3 the solutions of the different optimization problems are compared regarding to output error, actuation behavior and runtime. Furthermore, solutions in case of failures and large-scaled actuator sets are presented in this section.

4.1. Simulation

Our evaluation platform was a system equipped with a quad-core Intel Core i7-10510U processor running at a base frequency of 1.8GHz with a maximum turbo frequency of 4.9GHz. The system had 16GB of DDR4 RAM clocked at 2666MHz in a single-channel configuration. The code of the optimization problems were implemented in Python using ECOS as elaborated in Section 4.2.
To evaluate the solutions of the various models, the simulation of an ascent of a two-stage launcher is used. A two-stage launcher is a rocket composed of two separate stages, with a separation mechanism between them as introduced in Section 1.2. This simulation is based on the first stage of the vehicle, which includes the ascent phase of the full vehicle up to separation.

In Figure 4.1, the overall architecture of the system and available data from the simulation is illustrated. The *Allocation Unit* (red), is the unit where the constructed optimization problems can be implemented. In the simulation, the *pseudo-inverse allocation method* (PI) (Appendix A) was used to allocate the commanded force and torque over the available actuators as allocation strategy. This method uses the same input, commanded force and torque, and output, actuator commands, as the optimization problems. From the simulation, we can use the following data (of which the placement in the architecture can be seen in Figure 4.1):

- *Commanded Force* ($\vec{F}^{cmd}$) = this is the force that is commanded by the guidance for the specific velocity (in N)
- *Center of Mass* (CoM) = the center of the distribution of mass of the vehicle
- *Commanded torque* ($\vec{M}^{cmd}$) = the commanded rotational force by the control system so the direction of the vehicle aligns with the trajectory (in Nm)
- *Demanded Deflection Angles* = the solution of the PI-method for the individual deflection of the actuators (in radians)
- *Demanded Thrust* = the solution of the PI-method for the individual thrust of the actuators (in radians)
- *Actual Deflection Angles* = actual deflection output from the actuator (which may differ from the demanded due to actuator errors)

- *Actual Angular Acceleration* = actual angular acceleration output from the actuator (which may differ from the demanded due to actuator errors)
- *Actual Thrust* = actual individual thrust by the engines (which may differ the demanded due to actuator errors)
- *(Output) Forces* = the actual force output on the vehicle after actuation (in N)
- *(Output) Torque* = the actual rotational force output after actuation (in Nm)
- *Time* = the time in seconds for each timestep

The available information from the simulation can, in addition to using it as parameter input for the optimization models, be used for comparison with previous solutions of the allocation unit.

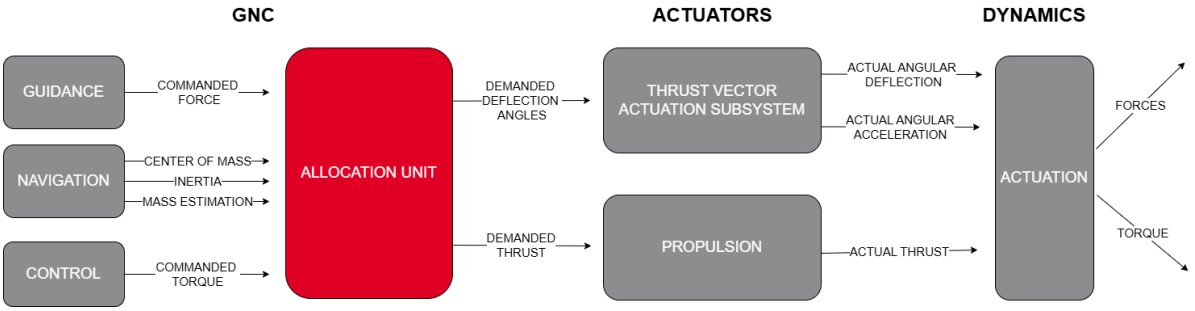


Figure 4.1: Guidance, Navigation, Control (GNC) architecture and the available data of the simulation (the text in arrows), where the allocation unit (red block) is the unit where the control allocation optimization problem can be implemented.

The two-stage launcher considered in the simulation has 5 engines of which the middle engine is not deflectable. The other 4 engines can deflect and are located on the vehicle as illustrated in Figure 4.2. This system is used for the testing of the optimization problems for the nominal states as well as the faults. After which, the optimization problems are also applied to a larger problem illustrated in next Subsection 4.1.

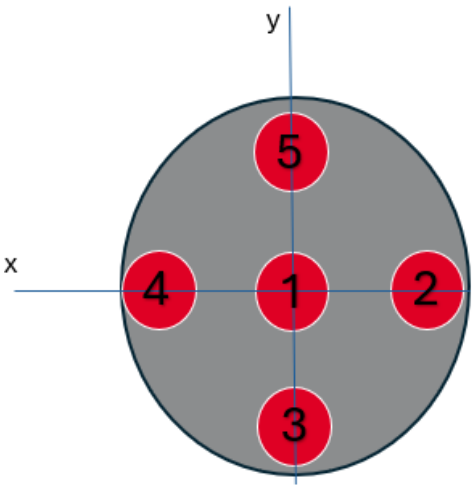


Figure 4.2: The position of the engines of the two-stage 5 engine rocket.

Scalability

To see if the problem formulation adapts properly to a rocket with more engines, we run the same optimization problems for a thruster with 27 engines. Due to the lack of simulation data for such a construction, the same torque and force commands are used for the scalability validation as in the previous simulation. To determine the amount of available thrust per engine, the amount of total possible thrust of previous simulation is divided over 27 engines instead of 5. Furthermore, the engines are located

as in Figure 4.3. It can be seen as three different rings around the center of mass, such that there are 15 engines in the outer ring, 9 in the middle ring and there are 3 engines in the center. Similarly structured as the SpaceX new Starship with 33 engines [34]. This is a multi-stage rocket with reusable first stage, the type of application we aim to target in this thesis. Let l_1 be the radius length for the first three engines, l_2 the radius of the middle ring and l_3 the radius of the outer ring. Let k_3 be the engines in the outer ring, k_2 the engines in the middle ring and k_1 the center engines. Engine $k_{3,i}$ refers to the ith engine in the outer ring in counter-clockwise direction, where the (x, y)-coordinates are given by

$$x_{k_{3,i}} = l_3 \cdot \cos(\frac{2\pi i}{15}), y_{k_{3,i}} = l_3 \cdot \sin(\frac{2\pi i}{15}) \text{ for } k_3 \in \{1, \cdots, 15\} \tag{4.1}$$

$$x_{k_{2,i}} = l_2 \cdot \cos(\frac{2\pi i}{9}), y_{k_{2,i}} = l_2 \cdot \sin(\frac{2\pi i}{9}) \text{ for } k_2 \in \{1, \cdots, 9\} \tag{4.2}$$

$$x_{k_{1,i}} = l_1 \cdot \cos(\frac{2\pi i}{3}), y_{k_{2,i}} = l_1 \cdot \sin(\frac{2\pi i}{3}) \text{ for } k_1 \in \{1, \cdots, 3\} \tag{4.3}$$

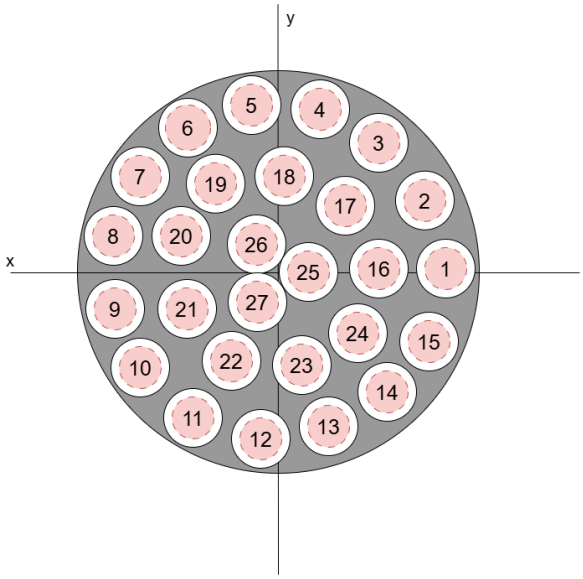


Figure 4.3: Thruster with 27 deflectable and throttleable TVC engines.

Torque command

The simulation data contains 40 timesteps per second in the ascent phase of a total of 1546 seconds. In Figure 4.4, the blue diagram represents the commanded torque over time. The green diagram represents the output torque after actuator activation by the solutions of the PI-solver. The overlap indicates that the solutions are generally able to match the commanded torque. After 20 seconds in the ascent phase, there is a high pitch command to tilt the launcher, such that there is some atypical high command of the torque compared to the rest of the flight. Therefore, this is an interesting point to analyze the solutions of the different problems, as in such cases it is useful to use both the freedom in deflection and differential thrust to fulfill the higher demanded torque.

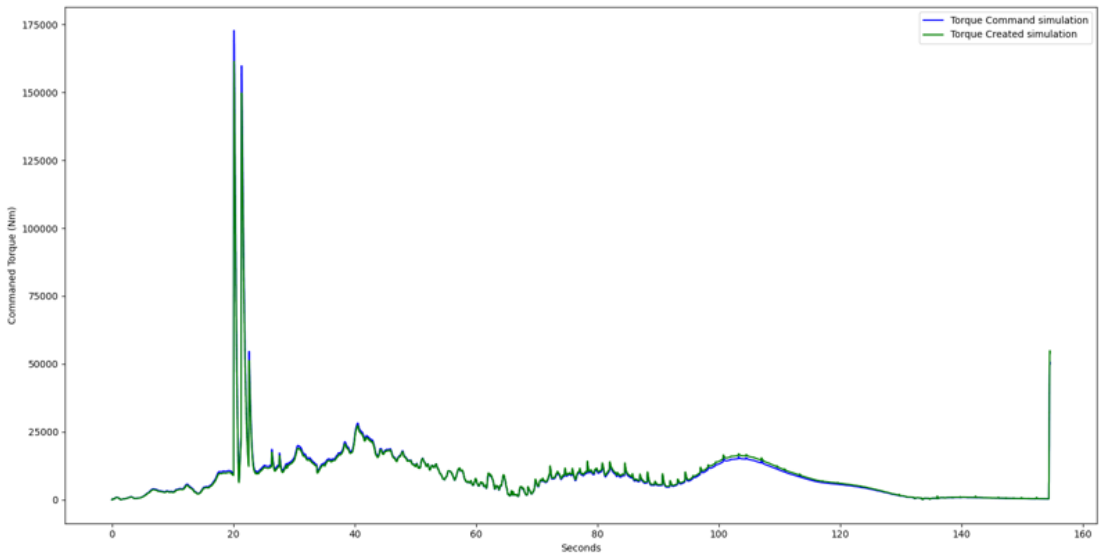


Figure 4.4: The commanded torque by the guidance (blue) and the created (output) torque after actuation (green) of Pseudo-Inverse solution over time.

Force command

As can be seen in Figure 4.5, the commanded force (blue) by the control system is constant over time. However, the measured force resulting from actuation of the actuators, determined by the Pseudo-Inverse solution (green), has a non-constant behavior. This is due to the fact that the actual commands are adjusted to external factors during flight. These adaptations are not included in the optimization problem as they are estimated by the control system and incorporated into the commands before they are used as input to the optimization problem. Therefore, the force computation in the optimization formulations will be based on the constant command. However, this makes it difficult to directly compare the computed forces with the solution of the pseudo-inverse. Therefore force solutions will only be compared as the error between the force resulting from actuator solutions from the optimization problems and the command.

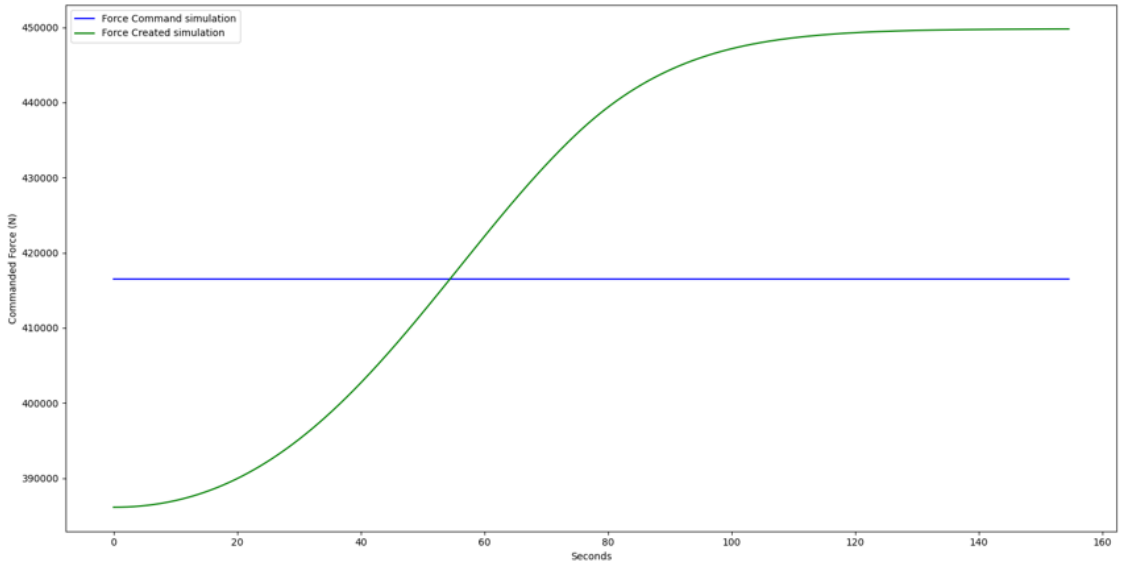


Figure 4.5: The commanded force by the control (blue) and the created (output) force after actuation (green) of Pseudo-Inverse solution over time.

4.2. Embedded Conic Solver (ECOS)

The optimization problems presented in this study were formulated and solved using CVXPY [9], a Python-based modeling language for convex optimization. As CVXPY allows to define problems in a readable, declarative manner and automatically converts them into standard forms suitable for numerical solvers. For solving the convex problems proposed in this thesis, the ECOS (Embedded Conic Solver) [10] was used. ECOS is an efficient open-source solver designed to handle SOCPs. The ECOS solver employs an interior-point method, with self-dual embedding. This algorithm iteratively approaches the optimal point while allowing for the detection of infeasibility and unboundedness. Furthermore, its integration with CVXPY ensures a direct workflow from problem definition to numerical solution.

4.3. Solutions comparison of optimization formulations

To compare the solutions, there are multiple points of interest. Since the problem is intended to be solved in real-time, the computational runtime is of high interest. Furthermore, it is essential to evaluate the error between the commanded torque and force and the resulting output torque and force after actuator activation. Another measure of interest is the actuator effort and behavior. The effort an behavior is analysed by the state of the deflection angles and differential thrust over time.
As the goal is to develop a fault-tolerant control allocation, these error and actuator effort solutions are also of high interest in scenarios involving actuator failures. To assess scalability, solutions are compared across different numbers of engines.

4.3.1. Error

In Figure 4.6 the output torque and force are presented after decision on the control allocation during the 21-24 seconds of the ascent phase, together with the commanded force and torque. The solutions to the optimization problems are generally equivalent with respect to the Pseudo-Inverse, which gives different solutions. This is the consequence of the difference described in the force command in Figure 4.5. In Figure 4.7 the optimization solutions are compared to the commands, by taking the difference between the output and command torque or force for the time between 19-24 seconds. The AD solutions give a larger error difference for M_x and M_y when there is a high command of torque. The linearized solution is closer to the command, whereas the error of the SOCP is almost zero. This

shows that the SOCP is generally better in satisfying the larger rotations of torque commands. The error of the F_z command is largest for the AD and linearized method, showing that the longitudinal force is not always satisfied. The longitudinal force is highly important during the launch to reach orbit by the right trajectory, including stage separation timing. If the longitudinal force is not satisfied, for example during the peak difference at 20 seconds, this might cause an unstable path causing oscillations or an off-nominal reentry angle.
The solutions for F_x, the lateral forces, are the same for any method. From these graphs it can be concluded that the error of SOCP is generally smallest, which confirms the assumption that this method is indeed more accurate as it includes less approximations than the linearized method, but has more actuator freedom than the AD method.

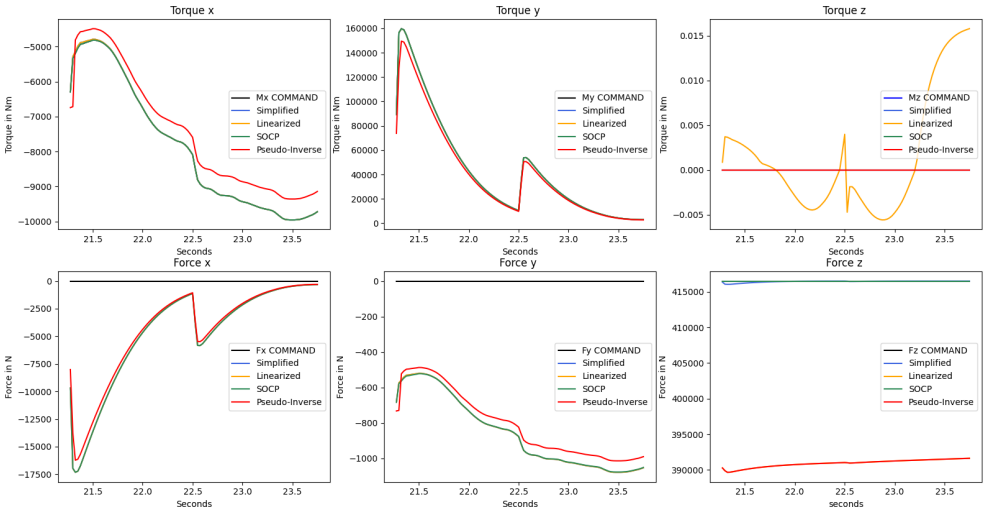


Figure 4.6: Results of torque and force outputs after actuator activation decided by optimization problems in 21-24 seconds ascent phase.

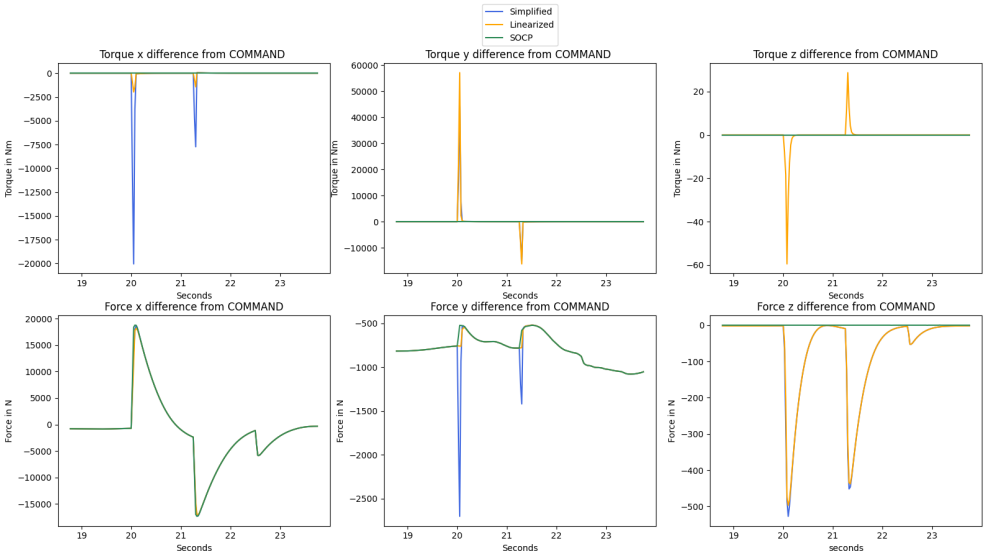
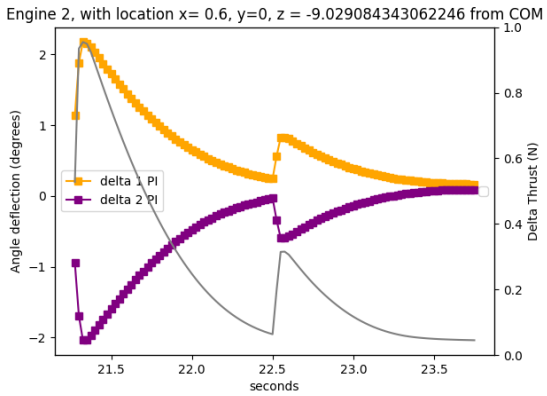


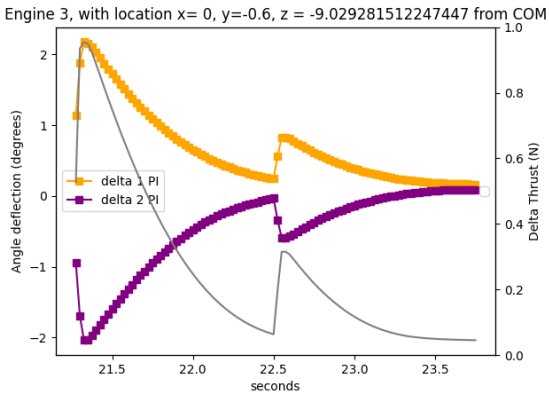
Figure 4.7: The difference between commanded torque and force and the output torque and force after actuator activation decided by the optimization problems in 19-24.

4.3.2. Actuation effort and behavior

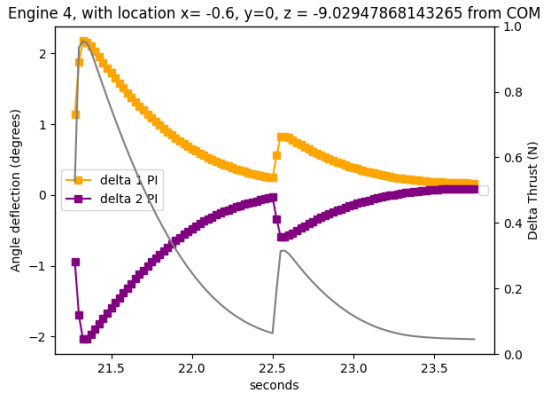
We also compare the actuation effort and behavior by looking at the states of the deflection angles and differential thrust. In Figure 4.8, one can see the actuator commands of the original PI method, which shows larger deflection angles for a higher torque command. Comparing it with Figure 4.9, the solutions of the PI problem and the AD optimization problem have a similar behavior. This is in line with the expectations as the AD problem has the same freedom as the PI problem, both not having the ability to differentiate thrust. Furthermore, the problem has the same linear approximations of the trigonometric functions such that the calculation of the force and torque are exactly the same. So the AD optimization problem gives the same expected solutions as the original approach, but is more flexible to extend to multiple engines, actuators and include faults.



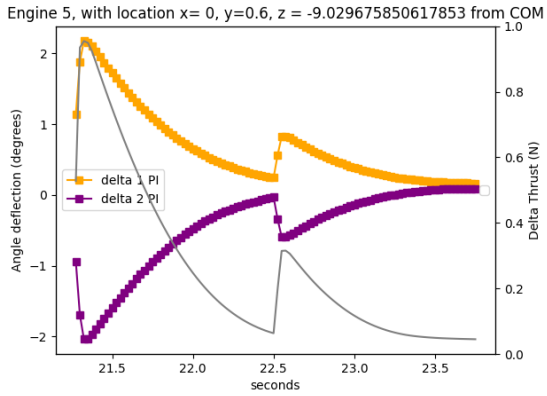
(a) Engine 2.



(b) Engine 3.

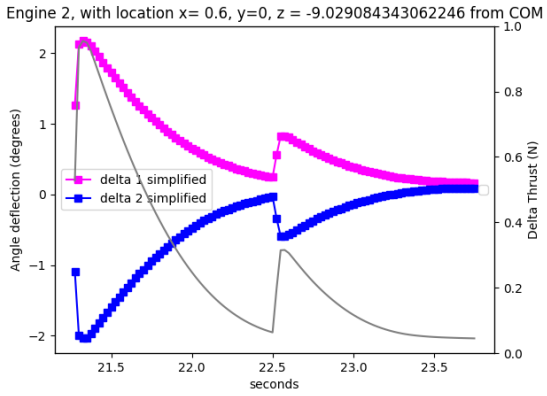


(c) Engine 4.

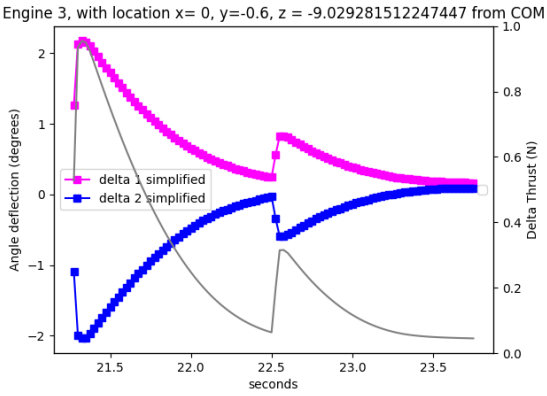


(d) Engine 5.

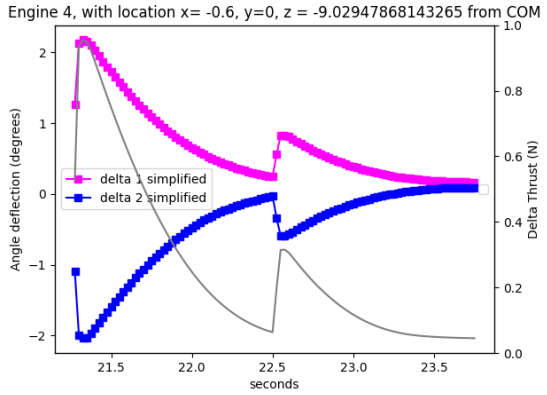
Figure 4.8: Angular deflection solutions from simulation (PI method) per engine. The left y-axis shows the angle deflection in degrees, for angle δ_1 in yellow and δ_2 in purple. The right y-axis shows differential thrust, which is none as the thrust is static in the PI method. The torque command is presented by the gray line, without axis. This line is there to illustrate the torque commands when the engines are deflected.



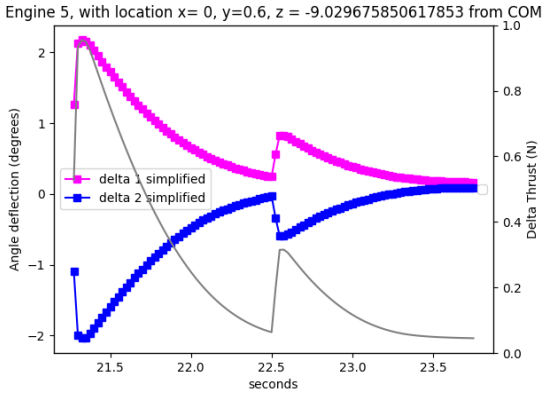
(a) Engine 2.



(b) Engine 3.



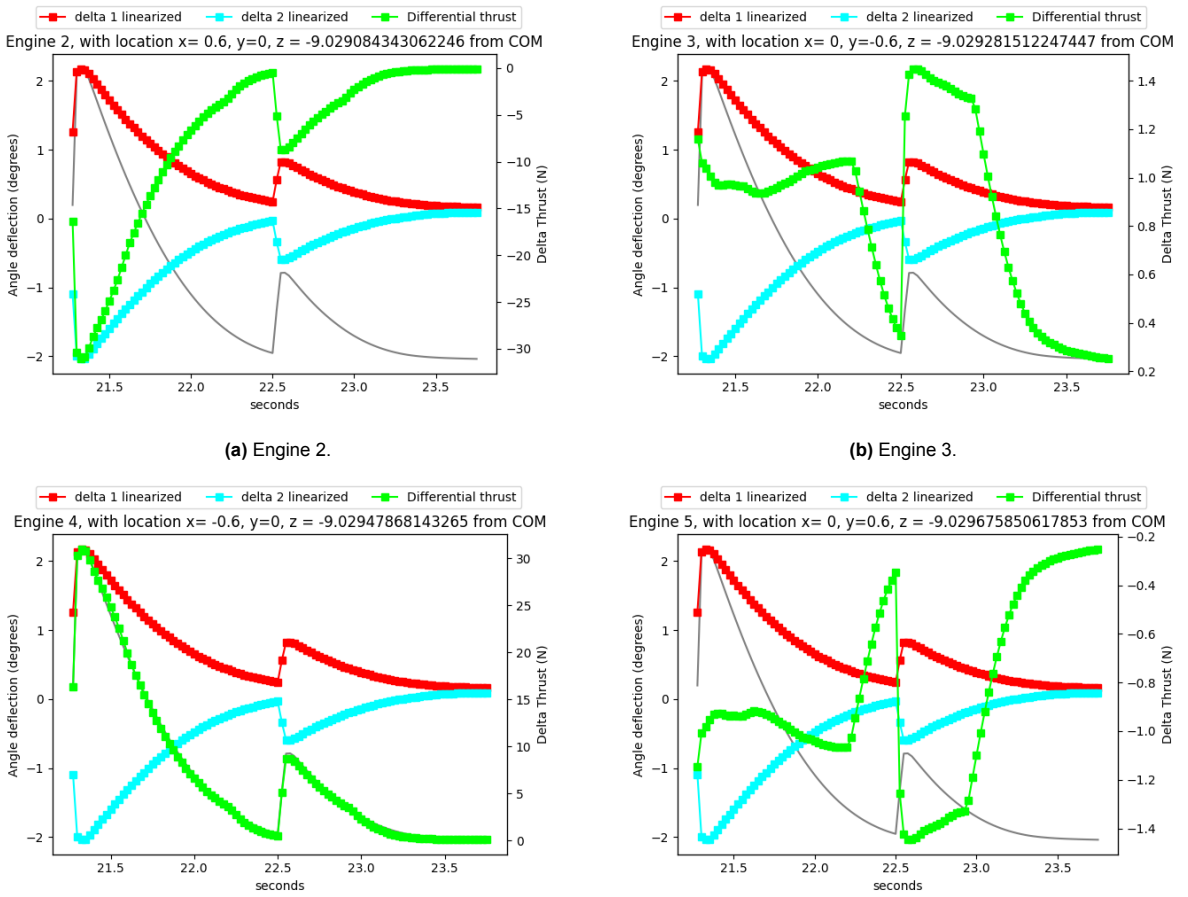
(c) Engine 4.



(d) Engine 5.

Figure 4.9: Angular deflection solutions from AD optimization problem using the actuator solutions from the ECOS solver per engine. The left y-axis shows the angle deflection in degrees, for angle δ_1 in pink and δ_2 in blue. The right y-axis shows differential thrust, which is none as the thrust is static in the AD method. The torque command is presented by the gray line, without axis. This line is there to illustrate the torque commands when the engines are deflected.

In contrast, the linearized problem also has the ability to differentiate in thrust. Figure 4.10 shows that the differential thrust changes between engines. From Figure 4.2, this difference can be explained by their position. The time when Engine 2 has a negative differential thrust, the opposite engine, Engine 4, has a positive differential thrust. The same conclusion is obtained from the other mirroring engines, Engine 3 and 5, although in smaller significance. This solution creates the insight that the freedom of differential thrust in the optimization problem can indeed contribute to extra torque when a larger change in direction is needed.



(a) Engine 2.

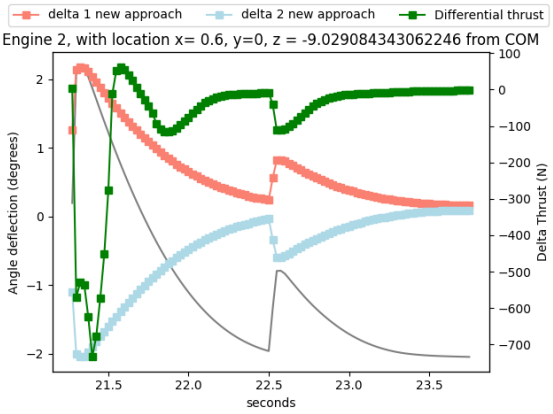
(b) Engine 3.

(c) Engine 4.

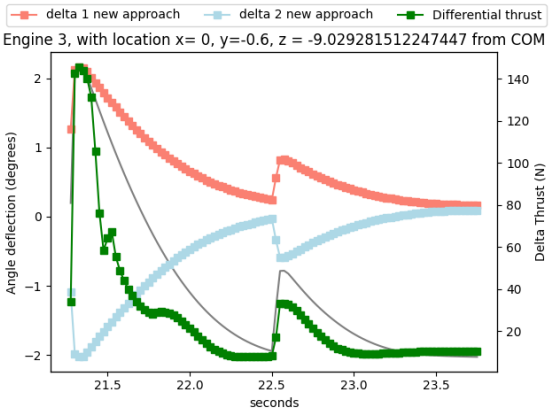
(d) Engine 5.

Figure 4.10: Angular deflection solutions from the linearized optimization problem using the actuator solutions from the ECOS solver per engine. The left y-axis shows the angle deflection in degrees, for angle δ_1 in red and δ_2 in turquoise. The right y-axis shows differential thrust, which is presented by the green line. The torque command is presented by the gray line, without axis. This line is there to illustrate the torque commands when the engines are deflected.

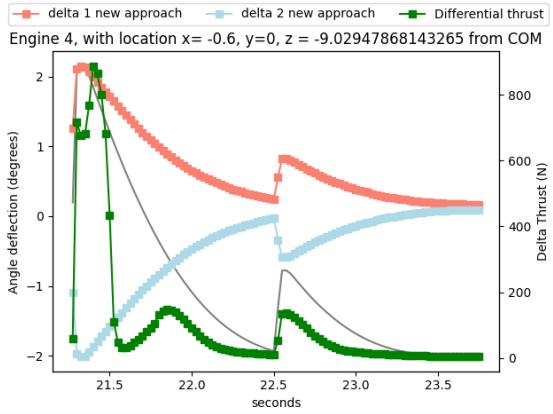
Figure 4.11 shows the solutions of the SOCP, where the decision variables are the individual force vectors. In this solution the differential thrust is also used to create torque, however the overall positive differential thrust is larger than the negative. As Engines 3,4,5 all have a higher thrust than the nominal state of the engines. Hence, the total differential thrust is higher, which is penalized in the objective function. However, it minimizes the difference between the output and commanded longitudinal force F_z, as obtained in Figure 4.7. Figure 4.12 show how the differential thrust is distributed among the different engines in the 5 engine thruster. This distribution shows that the uneven distribution of differential thrust adds a rotation in the same direction as the deflection angles.



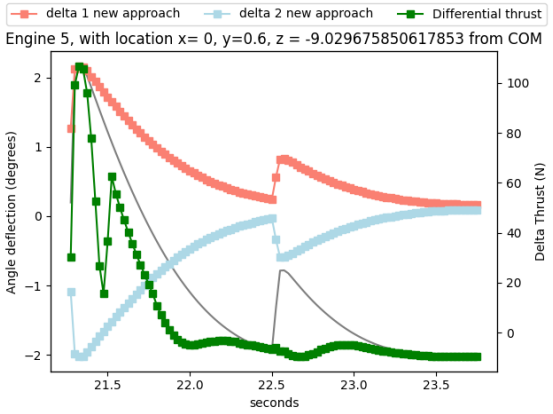
(a) Engine 2.



(b) Engine 3.



(c) Engine 4.



(d) Engine 5.

Figure 4.11: Angular deflection solutions from SOCP optimization problem using the actuator solutions from the ECOS solver per engine. The left y-axis shows the angle deflection in degrees, for angle δ_1 in light pink and δ_2 in light blue. The right y-axis shows differential thrust, which presented by the green line. The torque command is presented by the gray line, without axis. This line is there to illustrate the torque commands when the engines are deflected.

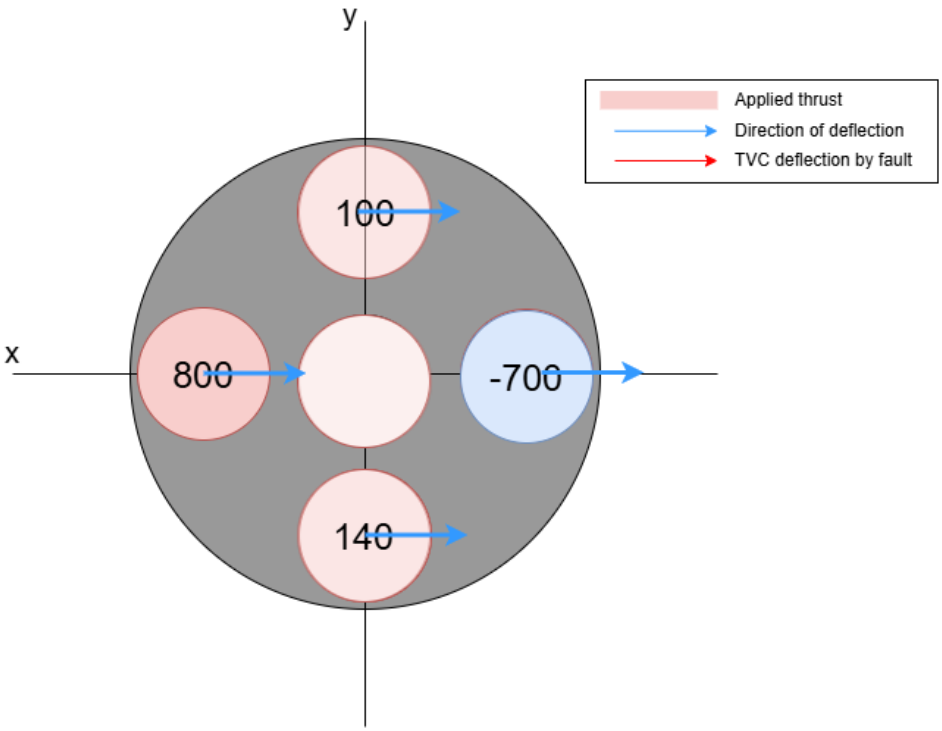


Figure 4.12: In this illustration the behavior of differential thrust and deflection angle solutions of the SOCP is presented for different engines in the 5 engine thruster in case of high torque command at 21 seconds during ascent phase. The Blue shows negative differential thrust in N, where the red shows positive differential thrust in N. The arrows show the direction of deflection.

4.3.3. Runtime

To compare the computational load of the optimization problems, the average runtime of the different formulations are illustrated in Figure 4.13. The runtime shows the amount of time needed to solve the optimization problem for one timestep, which has been done for 100 different timesteps with different input values. The AD problem generally needs less computation time, followed by the linearized problem and the SOCP has the largest computational load. It should be noted that solving the optimization based control allocation generally takes longer on an embedded control system in launchers. This can be up to 100 times slower, so we can expect each instance of the SOCP formulation to be solved in 1 second. In the simulation, the Pseudo-Inverse allocated commands 40 times per second. However, future work could explore optimizing the code for embedded implementation with lower computational load, for instance by precomputed values, code-level optimization or hardware optimization.

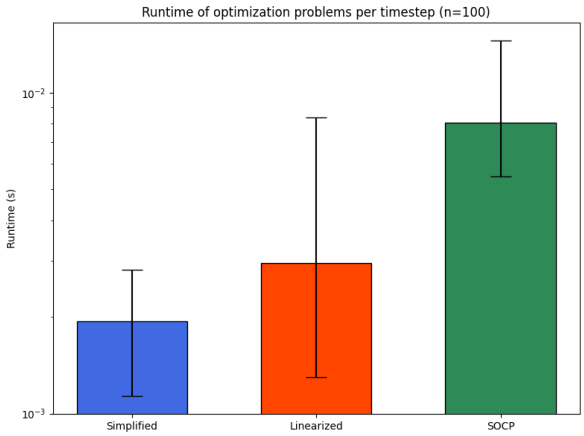


Figure 4.13: Runtime (s) of the optimization problem of one timestep in the control allocation problem for the AD, linearized and SOCP formulation solved with the ECOS solver. Ran for an instance with 5 engines with computer specifics as in Section 4.1.

4.3.4. Faults

The faults are implemented in the system as described in Section 3.9.

Loss of thrust

To see the reallocation of actuator commands in occurrence of loss of thrust, an example is taken when an engine can only produce 80% of its nominal thrust. So, $T_{s_1} = 0.8 \cdot T_{nom}$ for a faulty engine $s_1 \in S_1 \subset E$ as in Section 3.9.1. In the following example, the recovery methods of the different methods can be obtained for a 80% thrust level for faulty Engine 5.

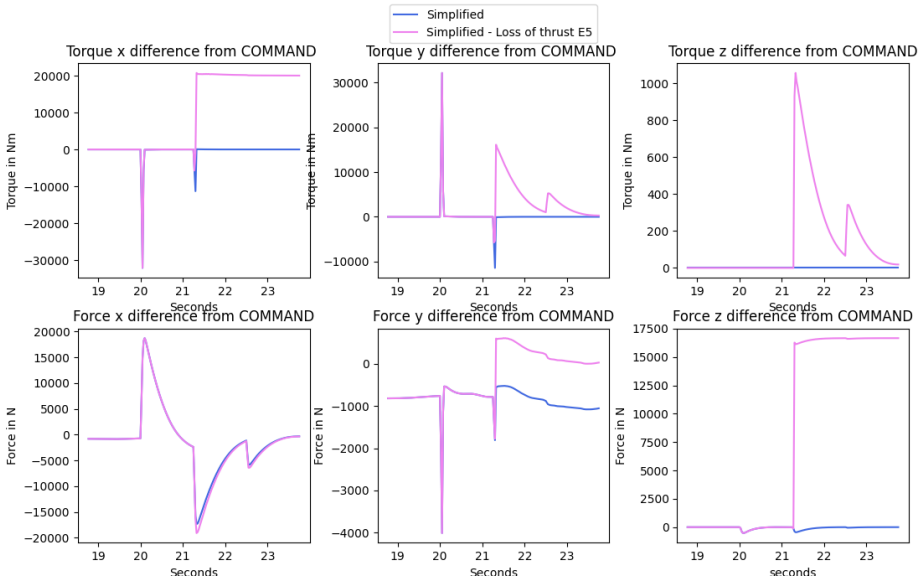


Figure 4.14: Difference between command and AD problem output for occurrence of fault at 21.4 seconds. The fault is a Loss of Thrust case of Engine 5, where it constantly uses only 80% of nominal thrust.

Figure 4.14 shows the difference between the force and torque command and the output force and torque using the actuator values after solving the AD formulation that includes the faulty engine. If one of the engines has a loss of 20%, the AD formulation is not capable of lowering the longitudinal error (F_z difference) because the AD formulation cannot adjust the thrust of the other engines. This error can lead to instability in trajectory tracking, as the control system tries to correct for the mismatch, which can cause oscillations or divergence from the desired path. Furthermore, it is inefficient as the path will deviate from the initially planned optimal trajectory. Lastly, if the longitudinal force error becomes

too large, the rocket may fail to achieve the necessary velocity or attitude for orbit insertion, leading to mission failure.

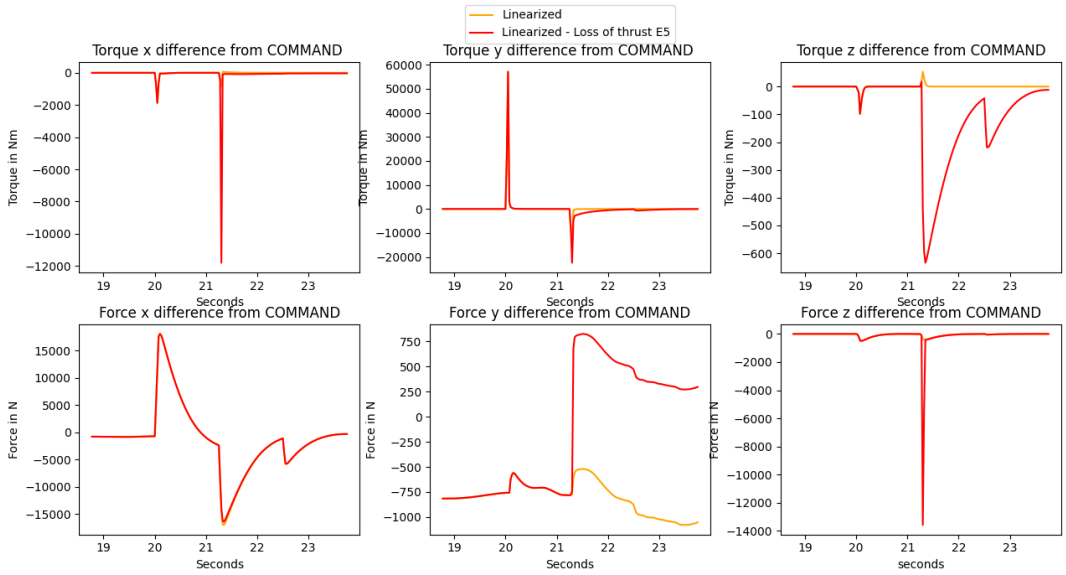


Figure 4.15: Difference between command and linearized problem output for occurrence of fault at 21.4 seconds. The fault is a loss of thrust case of engine 5, where it constantly uses only 80% of nominal thrust.

In Figure 4.15, where the linearized method is applied, we can see that the longitudinal error (F_z difference) is quickly decreased. The decrease in longitudinal error follows from the increased thrust of the non-faulty engines. The differential thrust of engines 1-4 increase from the nominal state, as can be obtained from Figure B.2 in Appendix B. The same recovery strategy can be seen in Figure 4.16 with corresponding individual actuation solutions in Figure B.3. In conclusion, the SOCP and linearized formulation are able to rapidly decrease the longitudinal force error in case of loss of thrust of 20%. Showing efficient reallocation of actuator effort over non-faulty actuators to minimize the error from control commands and thereby supports satisfaction of mission objectives.

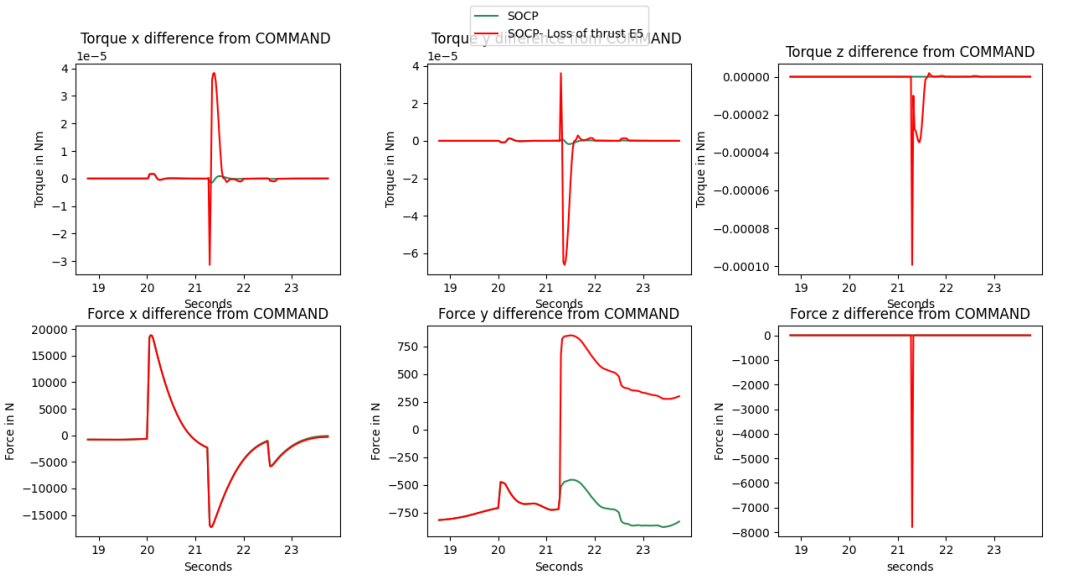


Figure 4.16: Difference between command and SOCP output for occurrence of fault at 21.4 seconds. The fault is a Loss of Thrust case of Engine 5, where it constantly uses only 80% of nominal thrust.

Furthermore, it is interesting to see how the optimization formulation recovers if one of the engines is fully shut. In that case $T_{s_1} = 0$. For the solution of the AD formulation (Figure 4.17), the same issue arises as in the previous loss of thrust, where it is not capable of counteracting on the F_z error.

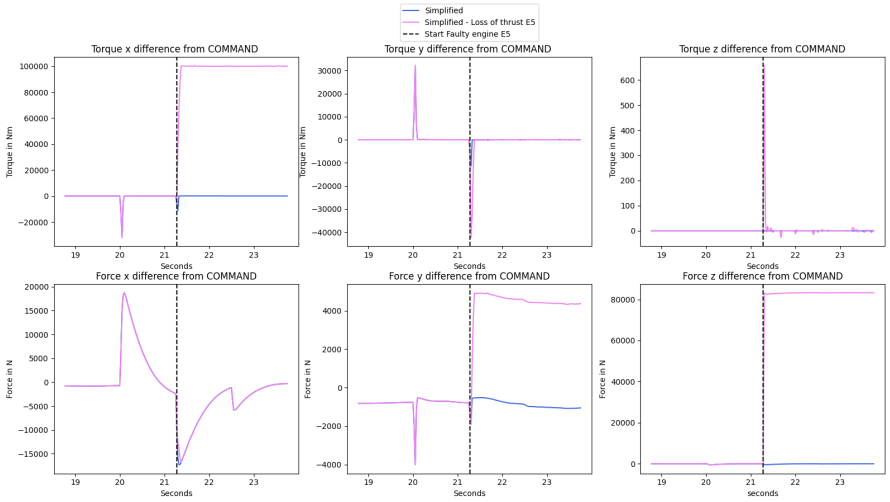


Figure 4.17: Difference between command and AD formulation output during fault at the dotted black line. The fault is a Loss of Thrust case of Engine 5 such that the engine is not capable of producing any thrust anymore.

The solution of the linearized formulation for total loss of thrust of Engine 5 (Figure 4.18) shows a solution where the other engines increase thrust to outbalance lost longitudinal force. However, it is not capable of full recovery for the loss of thrust, as the other engines are bounded by a maximum increase of thrust of 10%.

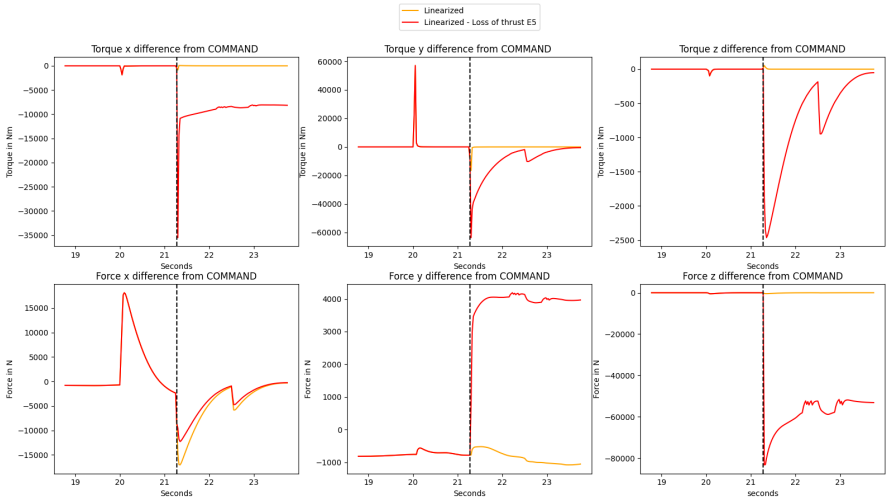


Figure 4.18: Difference between command and linearized formulation output during fault at the dotted black line. The fault is a Loss of Thrust case of Engine 5 such that the engine is not capable of producing any thrust anymore.

For the SOCP, without adapting the cost parameter values, the thrust of the other engines would converge to a minimum as well. However, as longitudinal force is a highly important factor during the launch, the parameter value of differential thrust and longitudinal error is increased to move to a solution where the other actuators counteract on the longitudinal force F_z loss, similarly to the linearized solution. So for efficient use of the formulations, changing cost parameters in case of faults can be

considered in further research. In that case, the system must be able to detect a fault to actively tune the cost parameters. Figure 4.19 shows a similar behavior of the SOCP compared to the linearized formulation. Though, the SOCP seems to constantly provide maximum thrust for the non-faulty engines, where the linearized does not always move to the maximum. This could be due to the lower accuracy of the linearized method in off-nominal states as illustrated by the 10% error graph in the formulation Chapter. In cases of faults, the differential thrust and deflections move outside the 10% accuracy set.

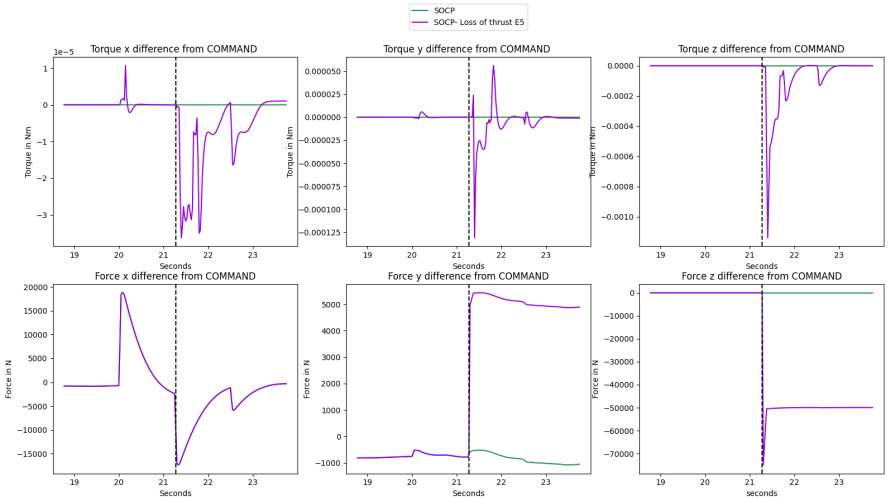


Figure 4.19: Difference between command and SOCP formulation output during fault at the dotted black line. The fault is a Loss of Thrust case of Engine 5 such that the engine is not capable of producing any thrust anymore.

Jamming TVC

In case of a jamming TVC, the nozzle is stuck to a certain deflection. For instance, a nozzle can be stuck to 0, so that it is not able to deflect from the nominal state. In that case, $\delta_{i,1} = \delta_{i,2} = 0$ for faulty engine i. This is implemented in the formulations according the models described in Subsection 3.9.2. In Figure 4.20, Engine 4 is stuck to having no freedom of deflection. The recovery strategy by the solutions of the optimization formulations is to enlarge the deflections of the other available engines. In Figure 4.20, this is shown for Engine 2 in the AD formulation, where δ_1 and δ_2 deflect to a 3° and -3° angle, in times of high torque command. While in the non-faulty case, these deflections were 2° and -2°. Engine 3 and Engine 5 take show the same deflection results as Engine 2, however Engine 1 is non-deflectable. The linearized as well as the SOCP formulations take the same recovery strategy, as can be seen in the Figures of Appendix B.3. However, the overall minimal error from the commands is obtained by the SOCP formulation, as it also applies differential thrust to the recovery strategy, as can be seen in Figure 4.21.

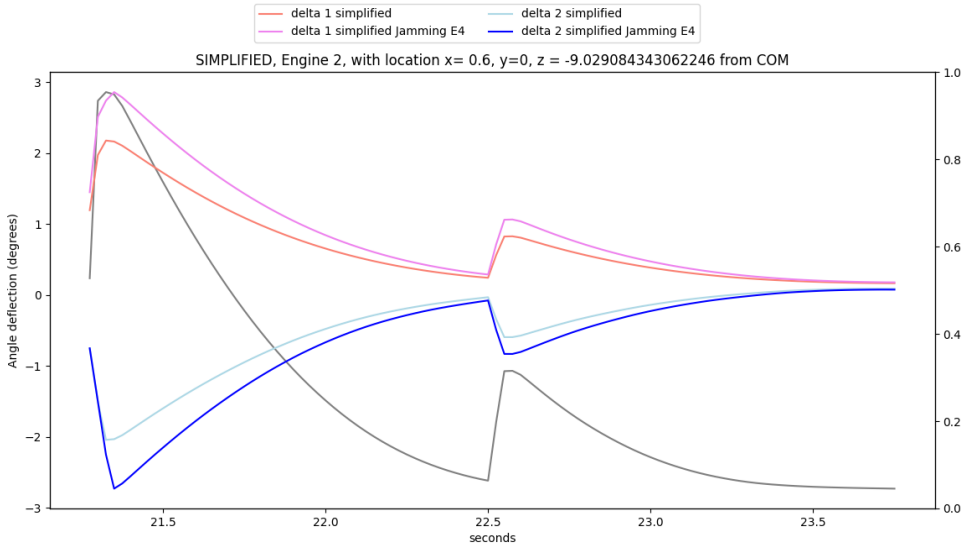


Figure 4.20: AD formulation output for Engine 2 when Engine 4 is not capable of deflecting. The left y-axis show the deflection in degrees for δ_1 in pink and δ_2 in blue. The gray line present the torque command over time.

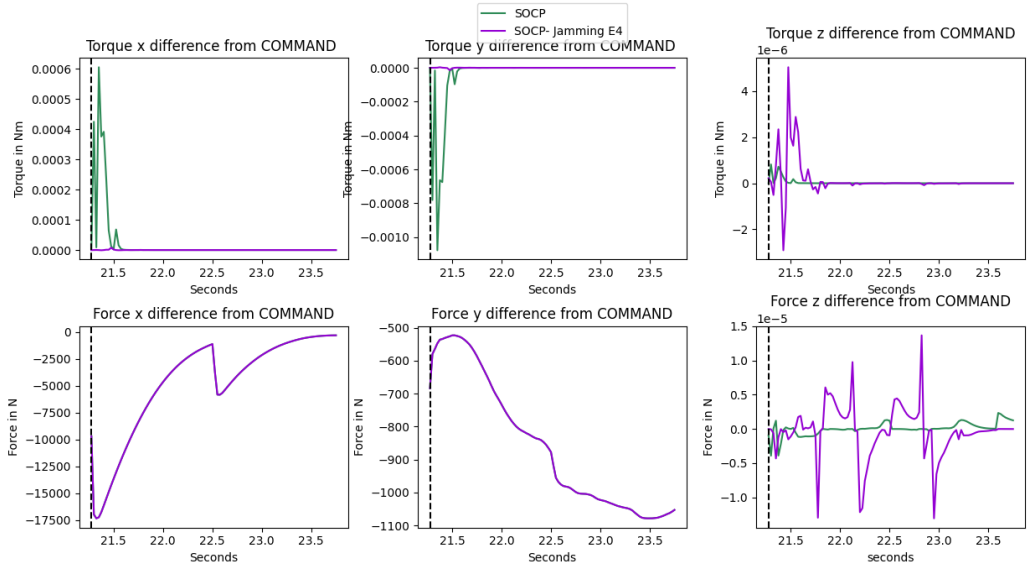


Figure 4.21: Difference of SOCP output from command for Jamming Engine 4 compared to a non-faulty case.

Loss of power of TVC

As described in Subsection 3.9.3, when there is a loss of power of the TVC, it is assumed to have maximum deflection into one direction of the nozzle. Therefore, to construct the fault, we set $\delta_{1,f} = 10 \cdot \frac{\pi}{180}$ and $\delta_{2,f} = 0$ for the faulty engine. In this example, we consider Engine 4 as the faulty engine. In Figure 4.22, the recovery behavior of the AD formulation can be seen. The graphs of the behavior of the individual engines can be found in Appendix B.4. As a recovery, the other engines start to deflect in the opposite direction to minimize the torque error. This also causes the lateral forces to increase. Therefore, as the thrust value of the AD formulation is static, the longitudinal force error increases compared to a non-faulty solution as can be seen in Appendix B.4. The recovery by the SOCP however, is able to use the differential thrust as well. The recovery strategy including the differential thrust can be found in Figure 4.23. The faulty engine produces less total thrust than the nominal state, such that the faulty deflection has less impact on the produced torque. The other engines increase thrust to make

up for the lost longitudinal force, which is a consequence of both the decrease of the faulty engine as well as the increase of lost force by lateral forces of the heavily deflected non-faulty engines. This can be seen by the effort behavior in Appendix B.4. Furthermore, the deflections behave the same as the AD solution. Although the linearized formulation is able to differentiate in thrust as well, the solution of the problem output a 0 differential thrust state, resulting in the same solution as the AD problem. This could be a consequence of poor tuning of the λ's or because of the high error possibilities for far off-nominal cases.

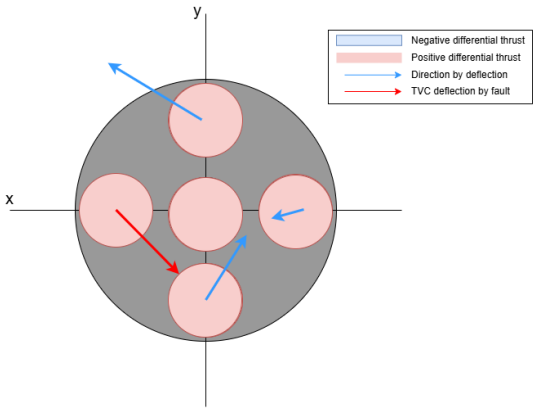


Figure 4.22: AD recovery method when there is loss of power over Engine 4. The red arrow describes the faulty deflection, the blue arrows present the recovery deflections.

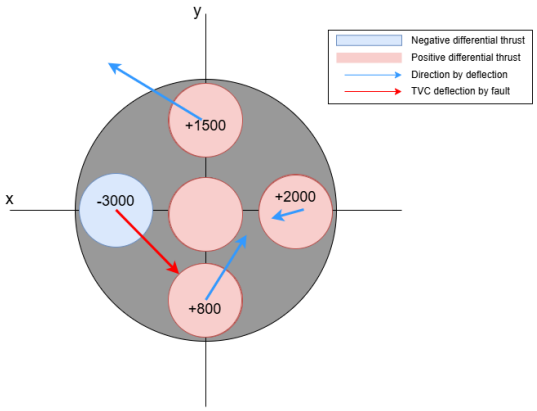


Figure 4.23: SOCP recovery method when there is loss of power over Engine 4. The red arrow describes the faulty deflection, the blue arrows present the recovery deflections. The blue engines describe negative differential thrust, where the red describe positive differential thrust.

4.3.5. Scalability

When applying the SOCP method on a problem with more thrusters, as constructed in Section 4.1, the solution provides asymmetric differential thrust solutions in case of high torque commands. In Figure 4.24, it is shown how the outer engines produce larger absolute differential thrust compared to inner engines. This is as expected, as differential thrust further away from the center of mass can add up more to the commanded torque using less thrust, by the larger radius from the center of mass. That said, it is also expected that the deflections of the engines differ per engine by position. But the actuation commands of the SOCP formulation have equivalent deflection values for deflections. However, the engines need less deflection per engine to satisfy the torque command, namely 1.5° and -1.5° instead of the 2° and -2° solutions for the 5 engine simulation data. To show the behavior, the deflection angles and differential thrust of Engine 10 is presented in Figure 4.25.

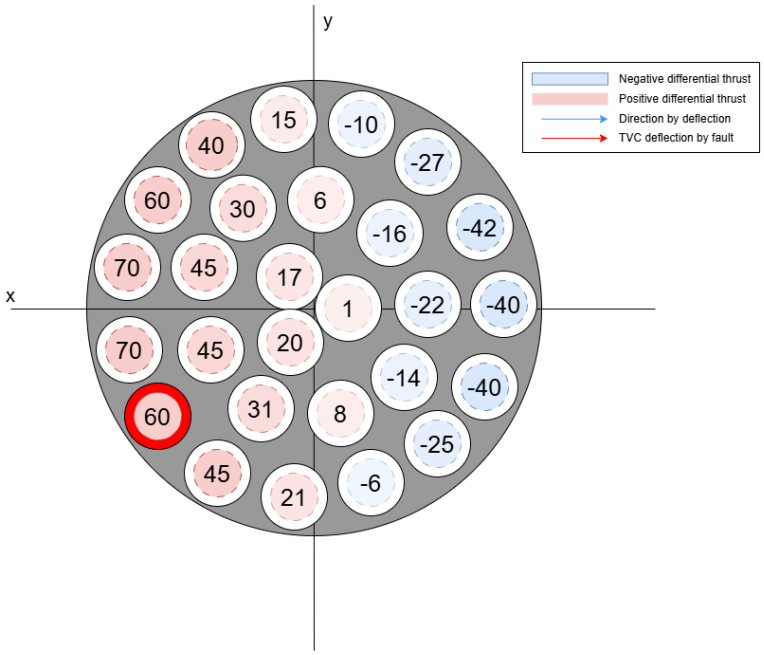


Figure 4.24: Difference in thrust from nominal state for every engine, where the numbers are in Newton. Engine 10 from Figure 4.25 is marked red.

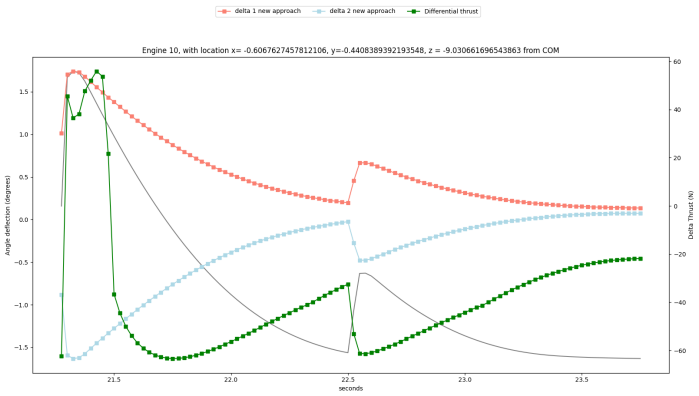


Figure 4.25: Behavior of Engine 10 in solution of thruster with 27 engines. The angle deflections are presented by the pink and blue line to the left y-axis, in degrees. The differential thrust (in N) is presented by the green line, corresponding to the right y-axis. The gray line presents the behavior of the commanded torque, without axis.

5

Conclusion

In conclusion, this thesis demonstrates three different actuator allocation optimization approaches. The angle-deflection (AD) problem is an approach for applications in control systems with actuators capable of deflection and a static output force. The linearized formulation shows a direct approach of including differential thrust of actuators while maintaining a convex problem by linear approximations. To avoid these approximations of control output by the actuator values, a SOCP formulation is proposed. This formulation tackles the allocation problem by optimizing the force output vector of individual actuators as decision variables. Then an inverse function allocates this optimal individual force vector over deflections and thrusts as actuator commands. It is shown in this thesis that an SOCP can be effectively constructed, while still providing the ability to penalize individual actuator efforts and satisfying actuator constraints of the TVC.

Furthermore, the formulations are designed so that in case of faults the optimization problem can reconfigure the output solutions without changing the structure of the formulation. In this thesis we considered fault types including loss of thrust, jamming TVC and loss of power. The formulation also gives the ability to include other actuators such as the RCS and aerodynamic fins, although the discrete nature of the RCS transforms the problem into a mixed-integer optimization problem. As this is not desired in the control system relying on convex conic solvers, further research has to be done to investigate relaxation techniques or other methods to efficiently solve the problem including these actuators.

To give the freedom of defining priority in satisfying multiple objectives in the control allocation problem, all proposed optimization objectives are tunable by cost parameters. Furthermore, this thesis proposes a dynamic weighting parameter to enable reallocation of actuator commands in case of constant long-term actuation of specific actuators. However, further research has to be done to show effectiveness of reallocation in the control allocation solutions as result of this dynamic parameter.

The formulations were implemented and solved for control commands during the ascend phase of a rocket. The AD method is the least computationally demanding, but restricting the decision of actuator control solely to deflections. In times of high torque command, the AD problem did not satisfy the control commands. The linearized and SOCP formulations have a better ability to satisfy the control commands in cases of high torque command. As these formulations use differential torque for additional rotation. However, the SOCP is computatively the most demanding, as expected.

Although the SOCP formulation entails a higher computational cost, it provides better accuracy in fault handling, with flexibility to adjust performance based on system requirements. The linearized formulation has higher errors for approximations of large off-nominal states and thereby is less accurate when handling faults. However, further research can be done in a sequential approach by linear approximations around the previous actuator states instead of the nominal.

Furthermore, the formulations show an effective adaptability to larger scaled problems. Nevertheless, observed discrepancies in deflection distribution across different engines suggests future research to optimize tuning strategies.

Suggestions for further work also include an adaptation for on-board implementation in flight software. Overall, this work proposes applicable optimization approaches in control allocation with potential for further refinement.

References

[1] K. Aardal, L. van Iersel, and R. Janssen. “Optimization”. In: *Lecture Notes AM2020, TU Delft*. September 28, 2023.

[2] Behçet Açıkmeşe. “Application of Lexicographic Goal Programming with Convex Optimization in Control Systems”. In: Aug. 2013. ISBN: 978-1-62410-224-0. DOI: 10.2514/6.2013-5103.

[3] T.M. Barrows and J.S. Orr. *Dynamics and Simulation of Flexible Rockets*. Academic Press, 2021. ISBN: 978-0-12-819994-7. URL: https://doi.org/10.1016/C2019-0-00384-6.

[4] Marc Bodson. “Evaluation of Optimization Methods for Control Allocation”. In: *Journal of Guidance Control and Dynamics - J GUID CONTROL DYNAM* 25 (July 2002). DOI: 10.2514/2.4937.

[5] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, Mar. 2004. ISBN: 0521833787.

[6] John Burken et al. “Two Reconfigurable Flight-Control Design Methods: Robust Servomechanism and Control Allocation”. In: *Journal of Guidance, Control, and Dynamics* 24 (June 2001). DOI: 10.2514/2.4769.

[7] G. B. Dantzig. *Linear programming*. Springer, 1997.

[8] M. Dasilva. “Rocket Rotations - Beginners guide to aeronautics”. In: (2023). URL: https://www1.grc.nasa.gov/beginners-guide-to-aeronautics/rocket-rotations/.

[9] Steven Diamond and Stephen Boyd. *CVXPY: A Python-Embedded Modeling Language for Convex Optimization*. 2016. arXiv: 1603.00943 [math.OC]. URL: https://arxiv.org/abs/1603.00943.

[10] Alexander Domahidi, Eric Chu, and Stephen Boyd. “ECOS: An SOCP solver for embedded systems”. In: *2013 European Control Conference (ECC)*. 2013, pp. 3071–3076. DOI: 10.23919/ECC.2013.6669541.

[11] Daniel Dueri, Frederick Leve, and Behçet Açıkmeşe. “Minimum Error Dissipative Power Reduction Control Allocation via Lexicographic Convex Optimization for Momentum Control Systems”. In: *IEEE Transactions on Control Systems Technology* 24.2 (2016), pp. 678–686. DOI: 10.1109/TCST.2015.2442838.

[12] Wayne C. Durham. “Constrained control allocation - Three-moment problem”. In: *Journal of Guidance, Control, and Dynamics* 17 (1994), pp. 330–336. DOI: 10.2514/3.21201.

[13] Stefano Farì et al. “Robust Fault Detection and Isolation Algorithms for TVC Systems: An Experimental Test”. In: *Proceedings of the 75th International Astronautical Congress (IAC)*. Milano, Italy: IAF, Oct. 2024. DOI: 10.52202/078373-0054.

[14] Thor I. Fossen and Tor A. Johansen. “A Survey of Control Allocation Methods for Ships and Underwater Vehicles”. In: *2006 14th Mediterranean Conference on Control and Automation*. 2006, pp. 1–6. DOI: 10.1109/MED.2006.328749.

[15] Gianluca Frison et al. “BLASFEO: Basic linear algebra subroutines for embedded optimization”. In: *CoRR* abs/1704.02457 (2017). arXiv: 1704.02457. URL: http://arxiv.org/abs/1704.02457.

[16] O. Harkegard. “Efficient active set algorithms for solving constrained least squares problems in aircraft control allocation”. In: *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.* Vol. 2. 2002, 1295–1300 vol.2. DOI: 10.1109/CDC.2002.1184694.

[17] Ola Härkegard. “Dynamic Control Allocation Using Constrained Quadratic Programming”. In: *Journal of Guidance Control and Dynamics - J GUID CONTROL DYNAM* 27 (Nov. 2004). DOI: 10.2514/1.11607.

[18] Tor A. Johansen and Thor I. Fossen. “Control allocation—A survey”. In: *Automatica* 49.5 (2013), pp. 1087–1103. ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2013.01.035. URL: https://www.sciencedirect.com/science/article/pii/S0005109813000368.

[19] Faïza Khelladi et al. “An Emergency Hierarchical Guidance Control Strategy for Autonomous Vehicles”. In: *IEEE Transactions on Intelligent Transportation Systems* 23.5 (2022), pp. 4319–4330. DOI: 10.1109/TITS.2020.3043485.

[20] W.S. Levine. *The Control Handbook: Control System Applications, Second Edition*. ISSN. CRC Press, 2018. ISBN: 9781420073614. URL: https://books.google.pt/books?id=mTX4NRtGNhIC.

[21] Chunsheng Liu et al. “Fault-tolerant control allocation for over-actuated discrete-time systems”. In: *Journal of the Franklin Institute* 352.6 (2015), pp. 2297–2313. ISSN: 0016-0032. DOI: https://doi.org/10.1016/j.jfranklin.2015.02.026. URL: https://www.sciencedirect.com/science/article/pii/S0016003215000988.

[22] Miguel Sousa Lobo et al. “Applications of second-order cone programming”. In: *Linear Algebra and its Applications* 284.1 (1998). International Linear Algebra Society (ILAS) Symposium on Fast Algorithms for Control, Signals and Image Processing, pp. 193–228. ISSN: 0024-3795. DOI: https://doi.org/10.1016/S0024-3795(98)10032-0. URL: https://www.sciencedirect.com/science/article/pii/S0024379598100320.

[23] Pedro Lourenço et al. “Verification & Validation of Optimisation-Based Control Systems: Methods and Outcomes of VV4RTOS”. In: *Proceedings of the 12th International ESA Conference on Guidance, Navigation & Control Systems*. Sopot, Poland: ESA, June 2023. DOI: 10.5270/esa-gnc-icatt-2023-155.

[24] Diego Navarro-Tapia, Pedro Simplicio, and Andrés Marcos. “Fault-Tolerant Dynamic Allocation Strategies for Launcher Systems”. In: *Aerospace* 12 (Apr. 2025), p. 393. DOI: 10.3390/aerospace12050393.

[25] Y. Nesterov and A. Nemirovskii. In: *Interior-Point Polynomial Algorithms in Convex Programming*, pp. 1–9. DOI: 10.1137/1.9781611970791.ch1. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611970791.ch1.

[26] Michael W. Oppenheimer, David B. Doman, and Michael A. Bolender. “Control Allocation for Over-actuated Systems”. In: *2006 14th Mediterranean Conference on Control and Automation*. 2006, pp. 1–6. DOI: 10.1109/MED.2006.328750.

[27] Carlo Alberto Pascucci, Michael Szmuk, and Behçet Açikmeşe. “Optimal control allocation for a multi-engine overactuated spacecraft”. In: *2017 IEEE Aerospace Conference*. 2017, pp. 1–6. DOI: 10.1109/AERO.2017.7943690.

[28] Nuno Paulino et al. “Fault Tolerant Control for a Cluster of Rocket Engines – Methods and Outcomes for Guidance and Control Recovery Strategies in Launchers”. In: *Proceedings of the 12th International ESA Conference on Guidance, Navigation & Control Systems*. Sopot, Poland: ESA, June 2023. DOI: 10.5270/esa-gnc-icatt-2023-085.

[29] J.A.M. Petersen and M. Bodson. “Constrained quadratic programming techniques for control allocation”. In: *IEEE Transactions on Control Systems Technology* 14.1 (2006), pp. 91–98. DOI: 10.1109/TCST.2005.860516.

[30] John Petersen and Marc Bodson. “Interior-Point Algorithms for Control Allocation”. In: *Journal of Guidance Control and Dynamics - J GUID CONTROL DYNAM* 28 (May 2005), pp. 471–480. DOI: 10.2514/1.5937.

[31] Cristina Roche Arroyos et al. “Fault-Tolerant Control for a Cluster of Rocket Engines - Results for Launch and Landing of a Re-Usable Launcher”. In: *Proceedings of the 2024 CEAS EuroGNC Conference*. Bristol, UK: CEAS, June 2024.

[32] Edouard Sadien et al. “A simple and efficient control allocation scheme for on-ground aircraft runway centerline tracking”. In: *Control Engineering Practice* 95 (2020), p. 104228. ISSN: 0967-0661. DOI: https://doi.org/10.1016/j.conengprac.2019.104228. URL: https://www.sciencedirect.com/science/article/pii/S0967066119301959.

[33] Pedro V. M. Simplicio, Andres Marcos, and Samir Bennani. “Reusable Launchers: Development of a Coupled Flight Mechanics, Guidance and Control Benchmark”. In: *Journal of Spacecraft and Rockets* (2019). ISSN: 0022-4650. DOI: 10.2514/1.A34429.

[34] SpaceX. “Starship”. In: 2025. URL: https://www.spacex.com/vehicles/starship/.

[35] L. A. Wolsey. *Integer programming (2nd ed)*. Hoboken, NJ:John Wiley Sons, Inc., 2021.

A

Pseudo-Inverse Allocation

Similarly to the Angle-Deflection (AD) problem, the *pseudo-inverse allocation problem* (PI) does not account for the engines' throttle capability. Instead, the method uses a constant nominal thrust value in the computation of forces and torques. The nominal level refers to the typical or standard thrust level, which is not necessarily the maximum. More often it is a thrust level that is the most efficient, meaning it generates the most thrust per unit of fuel consumed. In the PI method, the trigonometric functions of the delection angles are linearized according to the Taylor expansion as in the AD problem, such that $\vec{\delta} = \begin{bmatrix} \delta_{i,2} \\ -\delta_{i,1} \\ 1 \end{bmatrix}$. As there is no freedom in the amount of thrust per engine, the individual force and torque computation of engine i are computed as $\vec{F}_i = T_i^{nom} \cdot R_i \cdot \vec{\delta}$ and $\vec{M}_i = r_i \times \vec{F}_i$.

As the PI method used is specifically structured for a thruster where all the engines are similarly +45 degrees rotated from the axes of the vehicle, the rotation matrix is $R_i = \begin{bmatrix} \frac{1}{2}\sqrt{2} & \frac{1}{2}\sqrt{2} & 0 \\ -\frac{1}{2}\sqrt{2} & \frac{1}{2}\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}, \forall i$. Furthermore, for simplification of the computation the PI method assumes the engines are all symmetrically located from the center of mass, as in Figure 4.2. It is assumed Engine 1 is not capable of deflecting and is therefore excluded from the vector of deflection angles. Therefore, the computation of torque and force, which is the sum of individual force and torque of the engines, are calculated according to equation (1). We define

$$A = \frac{\sqrt{2}}{2} \begin{bmatrix} T_2^{nom} r_{1z} & T_2^{nom} r_{1z} & T_3^{nom} r_{1z} & T_3^{nom} r_{1z} & T_4^{nom} r_{1z} & T_4^{nom} r_{1z} & T_5^{nom} r_{1z} & T_5^{nom} r_{1z} \\ -T_2^{nom} r_{1z} & T_2^{nom} r_{1z} & -T_3^{nom} r_{1z} & T_3^{nom} r_{1z} & -T_4^{nom} r_{1z} & T_4^{nom} r_{1z} & -T_5^{nom} r_{1z} & T_5^{nom} r_{1z} \\ -T_2^{nom} r_{2x} & -T_2^{nom} r_{2x} & -T_3^{nom} r_{3y} & -T_3^{nom} r_{3y} & T_4^{nom} r_{2x} & T_4^{nom} r_{2x} & -T_5^{nom} r_{3y} & T_5^{nom} r_{3y} \\ -T_2 & T_2 & -T_3 & T_3 & -T_4 & T_4 & -T_5 & T_5 \\ -T_2 & -T_2 & -T_3 & -T_3 & -T_4 & -T_4 & -T_5 & -T_5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\Omega = \begin{bmatrix} \delta_{2,1} \\ \delta_{2,2} \\ \delta_{3,1} \\ \delta_{3,2} \\ \delta_{4,1} \\ \delta_{4,2} \\ \delta_{5,1} \\ \delta_{5,2} \end{bmatrix}, B = \begin{bmatrix} r_{3y}(T_3^{nom} - T_5^{nom}) \\ r_{2x}(T_4^{nom} - T_2^{nom}) \\ 0 \\ 0 \\ 0 \\ T_1^{nom} + T_2^{nom} + T_3^{nom} + T_4^{nom} + T_5^{nom} \end{bmatrix}, \text{such that}$$

$$\begin{bmatrix} M_x \\ M_y \\ M_z \\ F_x \\ F_y \\ F_z \end{bmatrix} = A \cdot \Omega + B. \tag{A.1}$$

Therefore, for a constant thrust and known commanded torque and force vectors, the angular deflections can be calculated according to the pseudo-inverse such that the solution of the allocation unit is

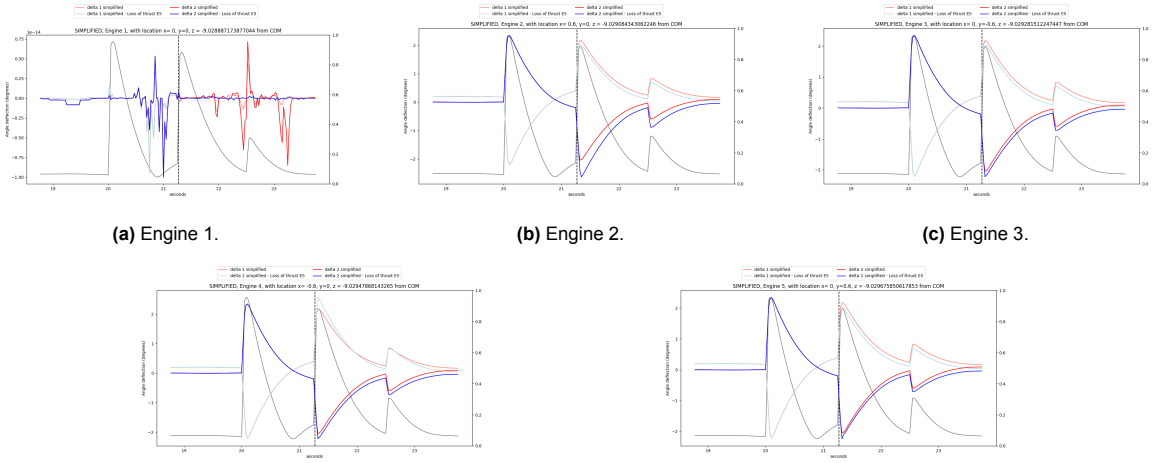
$$\Omega = [A]^{+} \left(\begin{bmatrix} M_x \\ M_y \\ M_z \\ F_x \\ F_y \\ F_z \end{bmatrix} - B \right), \text{where } [\,]^{+} \text{ is the Moore-Penrose pseudo-inverse.} \tag{A.2}$$

B

Fault recovery graphs

B.1. Loss of thrust - 80% Engine 5

AD formulation



(a) Engine 1.

(b) Engine 2.

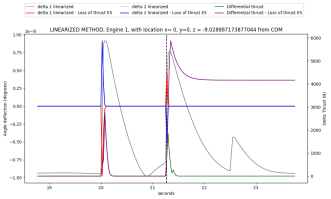
(c) Engine 3.

(d) Engine 4.

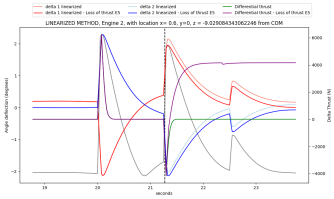
(e) Engine 5.

Figure B.1: Recovery of the AD problem formulation in case of a 20% loss of thrust of an outer engine (Engine 5)

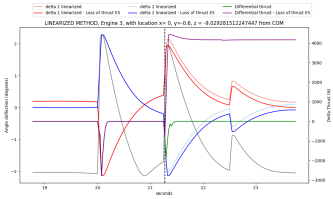
Linearized formulation



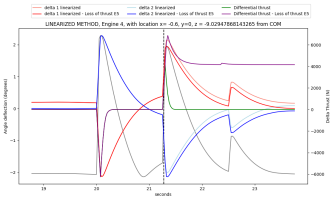
(a) Engine 1.



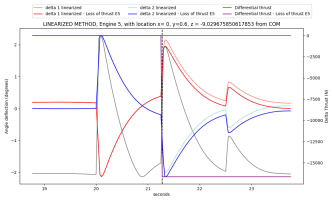
(b) Engine 2.



(c) Engine 3.



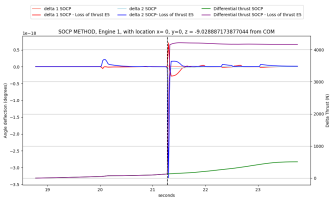
(d) Engine 4.



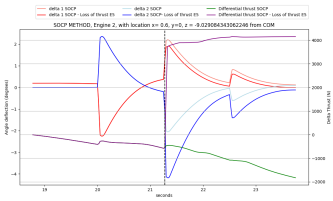
(e) Engine 5.

Figure B.2: Recovery of the linearized problem formulation in case of a 20% loss of thrust of an outer engine (Engine 5)

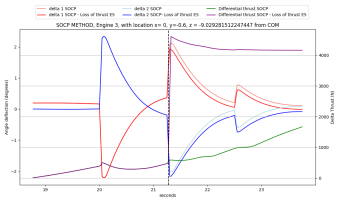
SOCP formulation



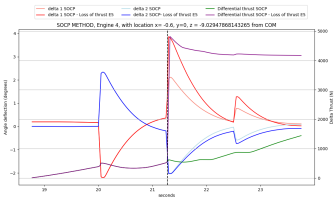
(a) Engine 1.



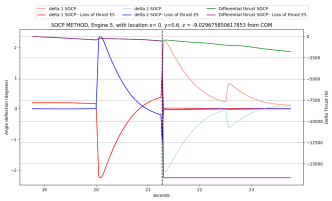
(b) Engine 2.



(c) Engine 3.



(d) Engine 4.

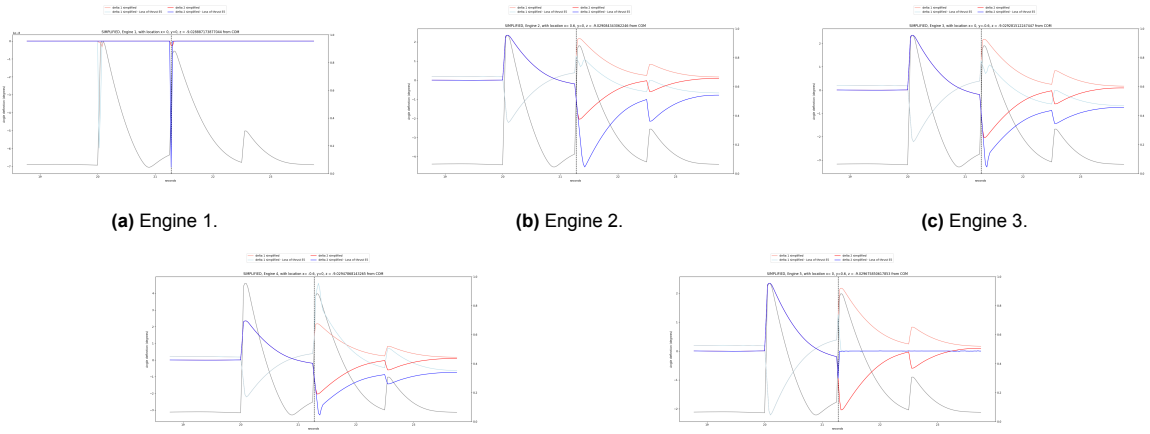


(e) Engine 5.

Figure B.3: Recovery of the SOCP formulation in case of a 20% loss of thrust of an outer engine (Engine 5)

B.2. Total Loss of Thrust Engine 5

AD formulation

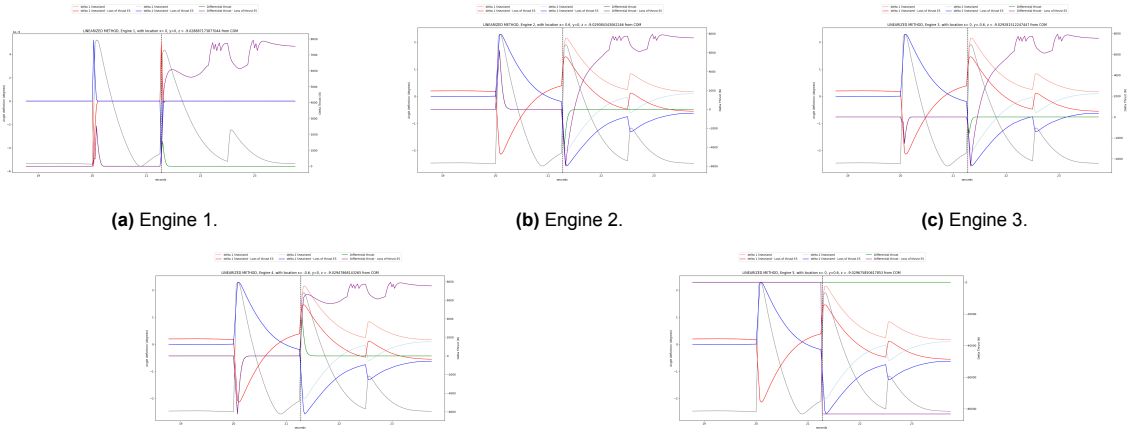


(a) Engine 1. **(b)** Engine 2. **(c)** Engine 3.

(d) Engine 4. **(e)** Engine 5.

Figure B.4: Recovery of the AD problem formulation in case of a total loss of thrust of an outer engine (Engine 5)

Linearized formulation

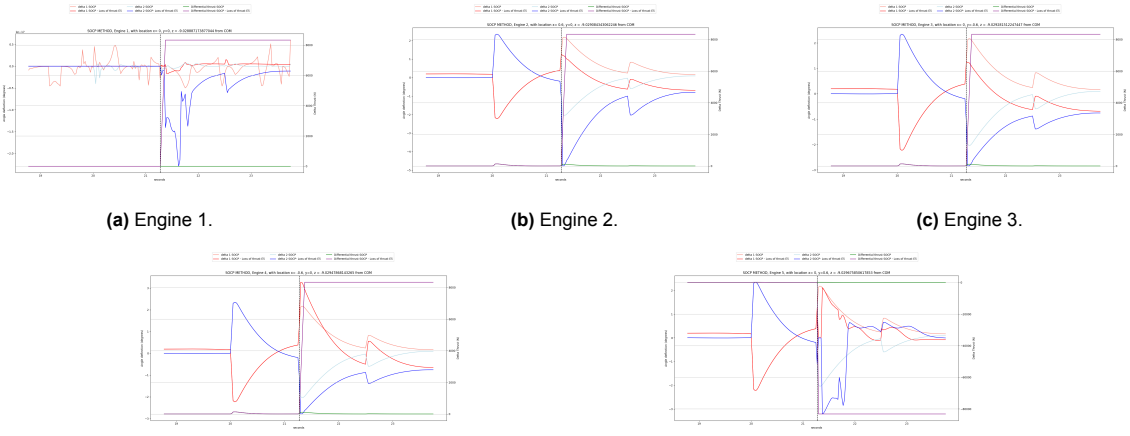


(a) Engine 1. **(b)** Engine 2. **(c)** Engine 3.

(d) Engine 4. **(e)** Engine 5.

Figure B.5: Recovery of the linearized problem formulation in case of a total loss of thrust of an outer engine (Engine 5)

SOCP formulation



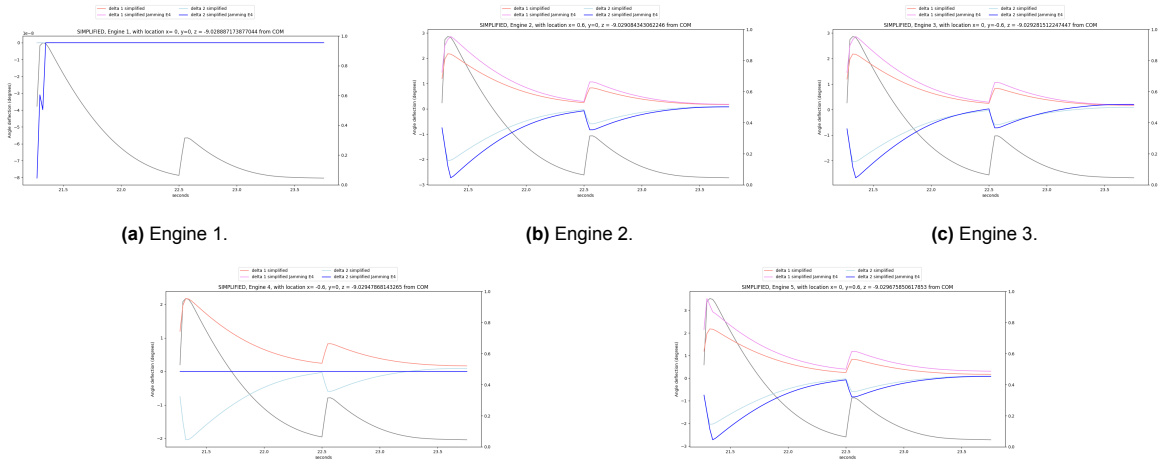
(a) Engine 1. **(b)** Engine 2. **(c)** Engine 3.

(d) Engine 4. **(e)** Engine 5.

Figure B.6: Recovery of the SOCP formulation in case of a total loss of thrust of an outer engine (Engine 5)

B.3. Jamming TVC Engine 4

AD formulation



(a) Engine 1. **(b)** Engine 2. **(c)** Engine 3.

(d) Engine 4. **(e)** Engine 5.

Figure B.7: Recovery of the AD formulation in case of a Jamming TVC such that Engine 4 has no deflection

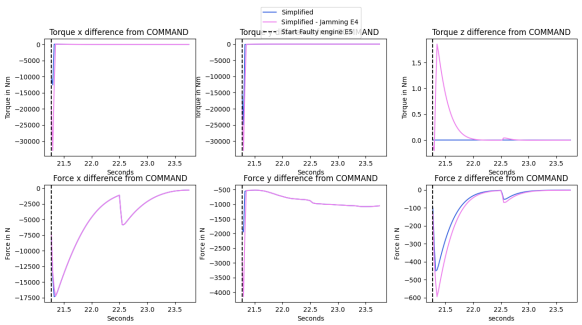
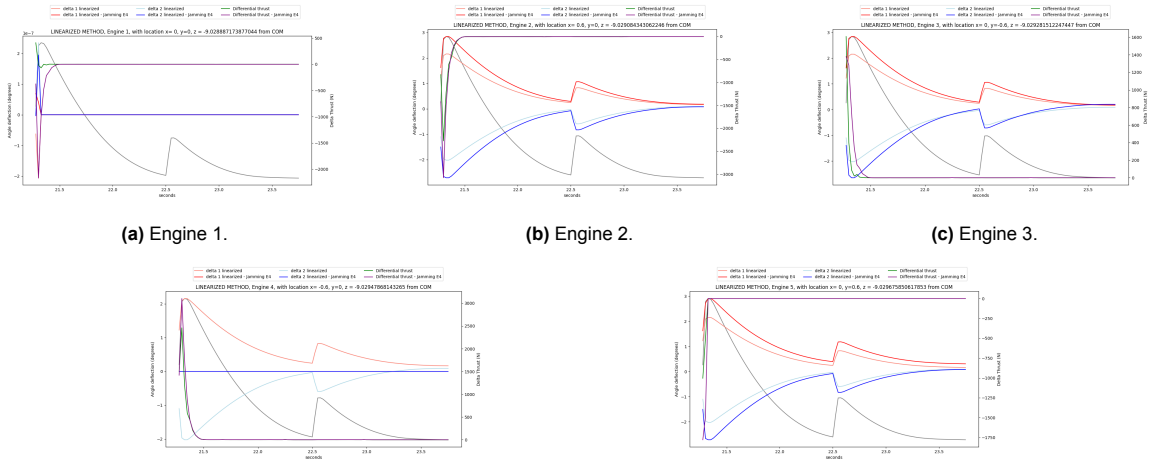


Figure B.8: Difference of AD output from command for Jamming Engine 4 compared to a non-faulty case.

Linearized formulation



(a) Engine 1. **(b)** Engine 2. **(c)** Engine 3.

(d) Engine 4. **(e)** Engine 5.

Figure B.9: Recovery of the Linearized formulation in case of a Jamming TVC such that Engine 4 has no deflection

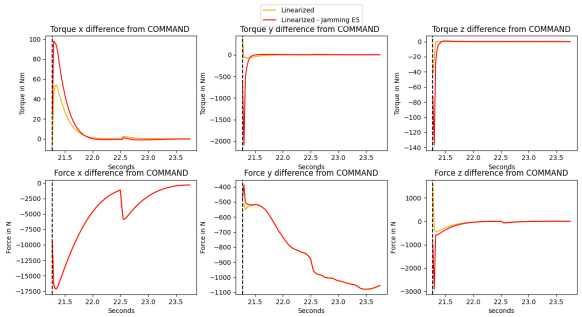
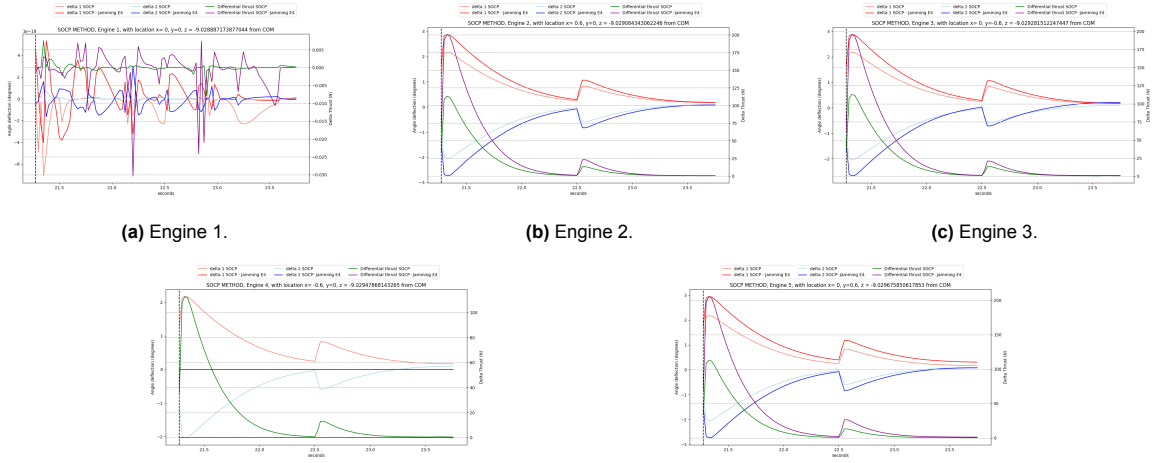


Figure B.10: Difference of Linearized output from command for Jamming Engine 4 compared to a non-faulty case.

SOCP formulation



(a) Engine 1. **(b)** Engine 2. **(c)** Engine 3.

(d) Engine 4. **(e)** Engine 5.

Figure B.11: Recovery of the SOCP formulation in case of a Jamming TVC such that Engine 4 has no deflection

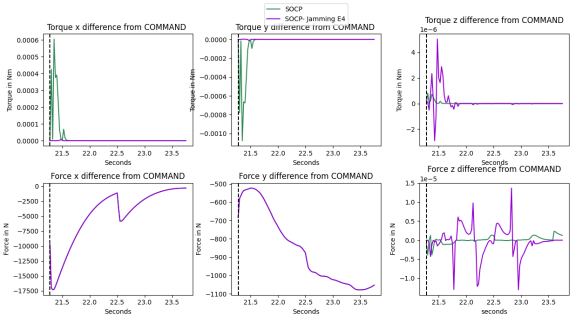
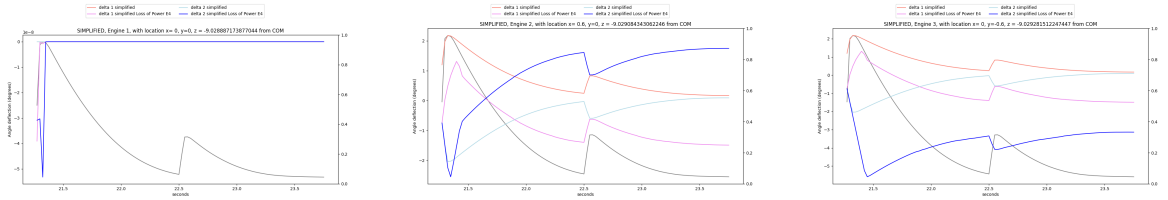


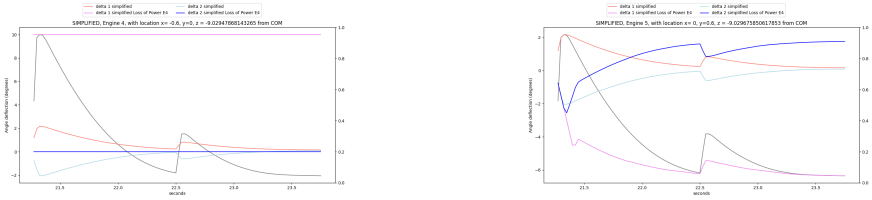
Figure B.12: Difference of SOCP output from command for Jamming Engine 4 compared to a non-faulty case.

B.4. Loss of Power - maximum deflection Engine 4

AD formulation



(a) Engine 1. **(b)** Engine 2. **(c)** Engine 3.



(d) Engine 4. **(e)** Engine 5.

Figure B.13: Recovery of the AD formulation in case of a loss of power of engine 4.

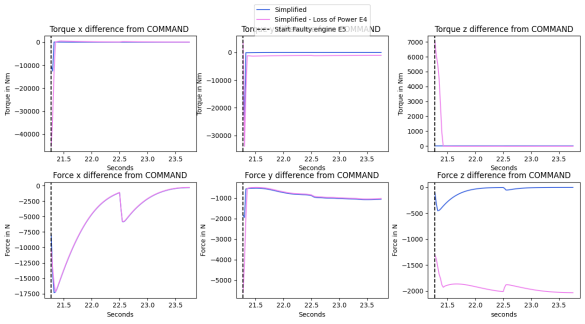
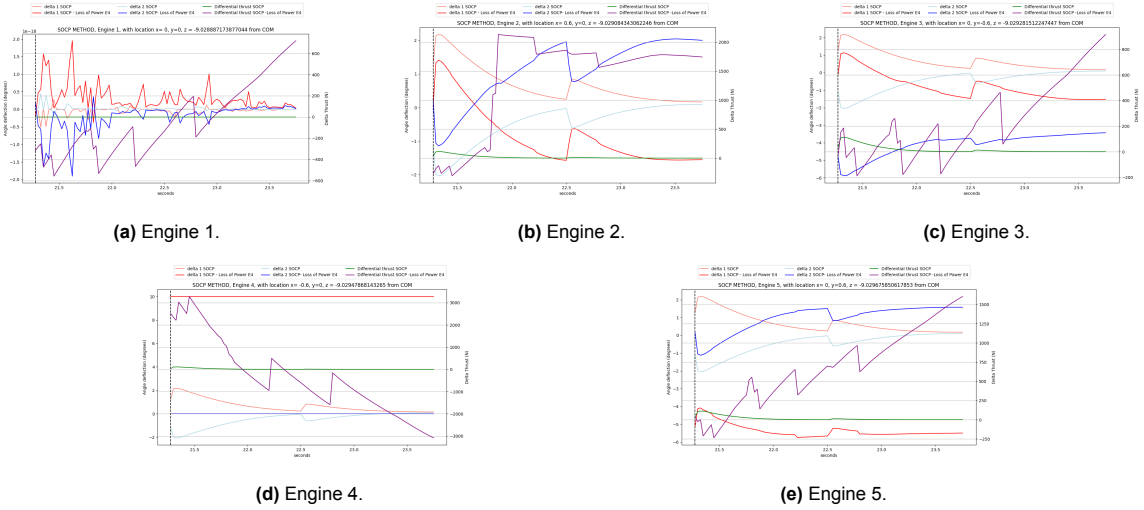


Figure B.14: Difference of AD output from command for Loss of power of Engine 4 compared to a non-faulty case.

SOCP formulation



(a) Engine 1. **(b)** Engine 2. **(c)** Engine 3.

(d) Engine 4. **(e)** Engine 5.

Figure B.15: Recovery of the SOCP formulation in case of a loss of power of engine 4.

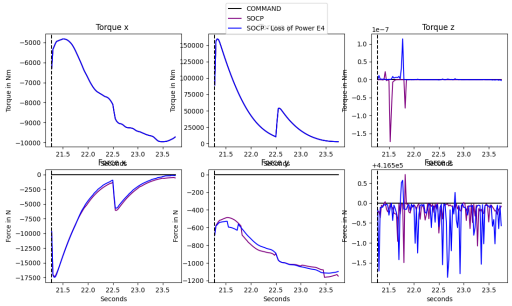


Figure B.16: Difference of SOCP output from command for Loss of power of Engine 4 compared to a non-faulty case.

C

Mixed Integer Second Order Cone Programming (MISOCP) formulation - including the Reaction Control System (RCS)

As the engines of the RCS are generally not deflectable, we can define the vector of deflections as

$$\begin{bmatrix} \sin(\delta_{i,2}) \\ -\sin(\delta_{i,1})\cos(\delta_{i,2}) \\ \cos(\delta_{i,1})\cos(\delta_{i,2}) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \text{for } \delta_{i,1} = \delta_{i,2} = 0. \tag{C.1}$$

Then the direction of the thrust is defined by the three dimensional rotation matrix R_i as input matrix. The magnitude of thrust is either the nominal or none, as the engine is not throttleable, therefore,

$$T_i = \begin{cases} T_{nom,i} & if\ ON \\ 0 & if\ OFF \end{cases} \tag{C.2}$$

Incorporating this as a constraint such that

$$\begin{bmatrix} F_{x,i} \\ F_{y,i} \\ F_{z,i} \end{bmatrix} = T_{nom,i} \cdot x_i \cdot R_i \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, x_i = \begin{cases} 1 & \text{if } i\ ON \\ 0 & \text{if } i\ OFF \end{cases} \tag{C.3}$$

turns the problem into a Mixed Integer Second-Order Conic Programming (MISOCP) problem. As this makes the problem non-convex, it increases the complexity significantly and cannot be solved by embedded convex solvers like ECOS. Branch and bound algorithms have been explored for MISOCPs, however, as the goal is real-time implementation, it is assumed to be computatively too heavy. Another approach is relax and round. Here, the MISOCP is relaxed into a continuous solution for thrust, whereafter the solution values are rounded into integer solutions after solving the relaxed problem. This approach whereby an optimal solution value for one problem is calculated recursively from the optimal values of slightly different problems is called Dynamic Programming (DP) [35].
Another approach of obtaining a convex problem while incorporating the RCS is by introducing a time dependent solution of the thrust by taking the integral of applied thrust over one timestep. Instead of only having two possible state values, it is then possible to have more attainable values of thrust by adjusting the amount of time the engine is turned on. Such that, turning the RCS system ON for only a partial amount of time of the timestep, results in output thrust solutions below the nominal. Thereby, the thruster firing time becomes a decision variable, $t \in [0, \Delta t]$, where Δt is one timestep, instead of the boolean $x_i \in \{0, 1\}$. However, there will still always be a minimum amount of thrust when the engine is turned on, keeping the problem non-convex.