NTNU
Norwegian University of
Science and Technology

Model Predictive Control Allocation

Martin Bøgseth Hanger

Master of Science in Engineering Cybernetics
Submission date: June 2011
Supervisor: Tor Arne Johansen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem Description

Kongsberg Defence Systems (KDS) is a business within Kongsberg Gruppen ASA. KDS works within military and civilian development/production. In the line of products are rockets, simulators, space application products, and also KDS is developing a small aircraft/missile. The process which describes the aircrafts performance is given by the size and shape of the aircraft body, its wings and control surfaces. The aircraft is controlled by regulating about its roll, pitch and yaw axes. The regulator commands enter actuator servos and are transferred to actual mechanical deflections of the control surfaces. How the transfer from regulator to surface deflections is done has an impact on the aircrafts properties and is called control allocation.

In an earlier project work report, multiple control allocation methods were documented, e.g. Linear Programming, Quadratic Programming and Redistributed Pseudoinverse. All of these proved to be potential control allocation methods for aircraft and missiles. MPC is a control algorithm which is used extensively in the process industry, but can also be applied as a control allocation method. It is expected to yield good results for aircraft and missiles. This thesis will develop a control allocation method based on MPC for processes with fast dynamics like aircraft and missiles. Further, MPC's performance will be compared to that of the previously mentioned control allocation methods, in applications with different actuator configurations.

Proposal of progression:

- Give a summary of the control allocation methods Linear Programming and Redistributed Pseudoinverse for flight control systems and document these.
- Develop a MPC control allocation (MPCA) method for flight control systems to an aircraft/missile.
- Develop in cooperation with KDS, an aircraft/missile model with simple dynamics.
- Develop a simple autopilot to be used while testing control allocation methods.
- Test step response on the aircraft/missile model for the control allocation methods Linear Programming, Redistributed Pseudoinverse and MPCA.
- Document the results, make an assessment of the methods, with emphasis on pros/cons and applications of the MPCA method.

Assignment given:	January 24, 2011.
Supervisor:	Tor Arne Johansen,
	Department of Engineering Cybernetics, NTNU.
Co-supervisor:	Åge Skullestad,
	Kongsberg Defence and Aerospace.

Preface

This thesis is the result of the work undertaken in the spring of 2011 at the Norwegian University of Science and Technology (NTNU). I would like to thank my supervisor Tor Arne Johansen at the Department of Engineering Cybernetics for his invaluable guidance, for giving me good ideas to work with and providing helpful advice. Thanks are also extended to my co-supervisor Åge Skullestad at Kongsberg Defense and Aerospace, who always contributed with quality advice and recommendations, aswell as providing background material when required.

Special thanks to my fellow office students Tor Marius Jensen, Pål Skønberg Løvik and Andreas Jørgensen, for being awesome friends and colleagues. The last year wouldn't be the same without you!

Thanks to Sindre Pedersen for taking the time to reviewing my thesis, doing spellchecking and content control.

Lastly, I would like to extend my best regards and thanks to my family and close friends, for supporting and encouraging me throughout my years at NTNU.

Abstract

This thesis developes a control allocation method based on the Model Predictive Control algorithm, to be used on a missile in flight. The resulting Model Predictive Control Allocation (MPCA) method is able to account for actuator constraints and dynamics, setting it aside from most classical methods. A new effector configuration containing two groups of actuators with different dynamic authorities is also proposed. Using this configuration, the MPCA method is compared to the classical methods Linear Programming and Redistributed Pseudoinverse in various flight scenarios, highlighting performance differences aswell as emphasizing applications of the MPCA method. It is found to be superior to the two classical methods in terms of tracking performance and total cost. Nevertheless, some restrictions and weaknesses are revealed, but countermeasures to these are proposed. The newly developed convex optmization solver CVXGEN is utilized successfully in the method evaluation. Providing solve times in milliseconds even for large problems, CVXGEN makes real-time implementations of the MPCA method feasible.

Table of Contents

List of Figures **iv**

Abbreviations **v**

1 Introduction **1**

2 Background **5**

2.1 Flight Control Systems 5

2.2 Autopilot 7

2.3 Control Allocation 7

2.3.1 Introduction 8

2.3.2 Control Allocation Problem Formulation 8

2.3.3 Optimality and the Attainable Control Set 9

2.4 Model Predictive Control 9

2.5 Missile Effector Configurations 11

2.6 CVXGEN 13

3 Control Allocation Methods **17**

3.1 Redistributed Pseudoinverse 17

3.2 Linear Programming 19

3.3 Model Predictive Control Allocation 20

3.4 Motivational Test 22

4 Missile Model **27**

4.1 Coordinate Systems 27

4.2 Kinematics 29

4.3 Rigid Body Dynamics 29

4.4 Translational Dynamics 30

4.5 Aerodynamic Forces and Moments 32

4.6 Complete Nonlinear Model 33

4.7 Simplified Model 34

4.7.1 Longitudal Model 34

4.7.2 Lateral Model 35

4.8 Actuator Dynamics 36

5 Test Setup **39**

5.1 Implementation 39
5.1.1 Autopilot 39
5.1.2 Prediction Module 41
5.1.3 Control Allocation Module 42
5.1.4 Actuator Dynamics 43
5.1.5 Missile Model 43
5.2 Control Allocation Methods 45
5.3 Virtual Control Prediction 45
5.4 Test plan 46
5.4.1 Part I - Tail fins only 46
5.4.2 Part II - Trimmed flight 46
5.4.3 Part III - Agile Flight 47
5.4.4 Part IV - Honoring Actuator Constraints 47
5.4.5 Part V - Prediction Comparison 48

6 Results **49**
6.1 Part I - Tail fins only 49
6.2 Part II - Trimmed flight 53
6.3 Part III - Agile flight 57
6.4 Part IV - Honoring Actuator Constraints 61
6.5 Part V - Prediction Comparison 65

7 Discussion **71**
7.1 Test with tail fins only 71
7.2 Performance in trimmed flight 72
7.3 Performance in agile flight 73
7.4 Honoring actuator constraints 74
7.5 On virtual control prediction 75
7.6 CVXGEN 75

8 Conclusion and Further Work **79**

Bibliography **84**

A Model Parameters **85**

B Missile Model Nomenclature **87**

C CVXGEN Code **89**

D RP CA Embedded Matlab Code **93**

E Conference paper **97**

List of Figures

1.1 Split control configuration 2

2.1 The flight control system as part of the homing loop. 5
2.2 The elements of the flight control system. 6
2.3 A discrete MPC scheme. 10
2.4 Missile with four fins in x configuration. View from behind. 12
2.5 Missile with four fins and wing ailerons. View from behind. 13
2.6 General purpose parser solver structure. Turns a single problem instance into a single optimal point. 14
2.7 Automatic code generator solver structure. Provides optimal points for many different problem instances. 14

3.1 MPCA Virtual Control Tracking 24
3.2 LP CA Virtual Control Tracking 24
3.3 MPCA Actuator Response 25
3.4 LP CA Actuator Response 25
3.5 Cumulative Cost 26

4.1 Aircraft Coordinate Systems [1] 28
4.2 Step response of a fast actuator 37
4.3 Step response of a slow actuator 38

5.1 Test setup overview. 40
5.2 Missile autopilot block. 41
5.3 Virtual control prediction block. 42
5.4 MPCA block. 43
5.5 Actuator Dynamics block. 44
5.6 Model block. 44

6.1 Part I - Tail fins only: Trajectory step response 50
6.2 Part I - Tail fins only: Actuator response. 51
6.3 Part I - Tail fins only: Virtual control tracking. 52
6.4 Part II - Trimmed flight: Trajectory tracking. 54
6.5 Part II - Trimmed flight: Actuator response. Left: Wing ailerons. Right: Tail fins. 55
6.6 Part II - Trimmed flight: Virtual control tracking. 56
6.7 Part II - Trimmed flight: Cumulative cost. 57

6.8 Part III - Agile flight: Trajectory tracking. 58
6.9 Part III - Agile flight: Actuator response. Left: Wing ailerons. Right: Tail fins. 59
6.10 Part III - Agile flight: Virtual control tracking. 60
6.11 Part III - Agile flight: Cumulative cost. 61
6.12 Part IV - Honoring Actuator Constraints: Trajectory tracking. . . . 62
6.13 Part IV - Honoring Actuator Constraints: Actuator response. Left: Wing ailerons. Right: Tail fins. 63
6.14 Part IV - Honoring Actuator Constraints: Actuator command. Left: Wing ailerons. Right: Tail fins. 64
6.15 Part IV - Honoring Actuator Constraints: Cumulative cost. 65
6.16 Part V - Prediction Comparison: Trajectory tracking. 66
6.17 Part V - Prediction Comparison: Actuator response. Left: Wing ailerons. Right: Tail fins. 67
6.18 Part V - Prediction Comparison: Virtual control tracking. Left: Wing ailerons. Right: Tail fins. 68
6.19 Part V - Prediction Comparison: Cumulative cost. 69

7.1 CVXGEN solve time scaling. 77
7.2 CVXGEN vs CVX solve time comparison. 78

Abbreviations

BTT Bank-to-Turn.

CA Control Allocation.

CG Center of Gravity.

CGI Cascaded Generalized Inverse.

DOF Degrees of Freedom.

FCS Flight Control System.

IMU Inertial Measurement Unit.

ISS Input-to-State Stability.

LOS Line of Sight.

LP Linear Programming.

LQR Linear Quadratic Regulator.

MPC Model Predictive Control.

MPCA Model Predictive Control Allocation.

NED North-East-Down.

ODE Ordinary Differential Equation.

PID Proportional-Integral-Derivative.

QP Quadratic Programming.

RB Rigid Body.

RHC Receding Horizon Control.

RP Redistributed Pseudoinverse.

STT Skid-to-Turn.

Chapter 1

Introduction

Some control systems are designed with redundant actuators and effectors, for reasons such as fault tolerance and design issues related to cost, response-time, size, and flexibility. Examples include flight control systems [2], which will be studied in this thesis, and dynamic positioning systems for ships using thrusters [3].

In aerospace applications, one usually talks about navigation of an aircraft in terms of rotation about the three rotational degrees of freedom, called roll, pitch and yaw for rotations about the x-, y- and z-axis, respectively. Traditionally for aircraft, each of these rotations are tied to a group of control surfaces (effectors) - ailerons for roll, elevator for pitch and rudder for yaw. These surfaces are intended to generate moments about a single axis, but due to cross-coupling effects, this is often not the case. An actuator is responsible for deflecting the individual effectors. The traditional surfaces are commonly split up to reduce individual actuator demands and increase redundancy. In later years new aerodynamic controls like thrust vectoring[1] have also been added to serve as moment generators on the aircraft body, in addition to control surfaces. Future control concepts include using large arrays of small surfaces as moment generators [4].

Control algorithm design for systems with effector redundancy is challenging since multiple combinations of the available control effectors can generate the same desired control. In addition to this, actuator constraints (e.g. limited deflection) should be accounted for. In order to systematically manage such control design challenges, one may decompose the control problem into two parts - a controller that commands a desired virtual control input and a control allocation module that maps the virtual control input into the redundant actuators (Figure 1.1). Since there are more degrees of freedom available in the actuator system than virtual control variables, the available degrees of freedom in the actuator system can be used to satisfy actuator constraints and to meet secondary objectives such as fault tolerance, power

[1]Thrust vectoring is the ability of an aircraft, rocket or other vehicle to manipulate the direction of the thrust from its engine(s) or motor in order to control the attitude or angular velocity of the vehicle.

consumption minimization, and actuator wear minimization.

Other benefits of having a split control configuration include easy reconfiguration in case of actuator change, separated regulation tuning, and lastly that the control allocation method can be arbitrary [5]. In general, the control allocation problem can be formulated as an optimization problem with a main objective of minimizing the difference between the resulting control effect and the desired virtual control command, and where actuator and effector constraints show up in the problem's constraint set. The main difference between different control allocation methods are related to how the optimization problem is formulated, which models are used, and which numerical algorithm is employed to solve it.

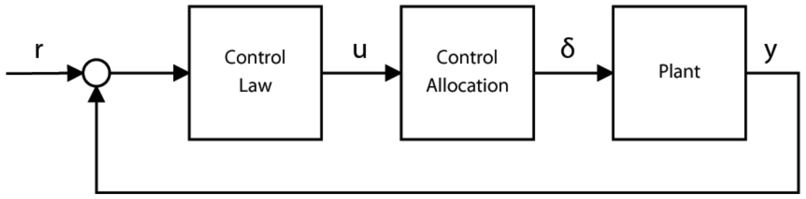

Figure 1.1: Split control configuration

In the classic formulations of the constrained control allocation problem, the actuator dynamics are neglected [2]. This is done under the assumption that the actuator dynamics are orders of magnitude faster than the aircraft dynamics, and thus can be ignored, or that all dynamic phenomena are accounted for by the controller that commands the virtual control to the control allocation module. In some cases this may be an unrealistic and inconvenient assumption, i.e. when the actuator dynamics are limiting the control performance because response times and different dynamic authorities of the actuators are not taken into account.

For systems where actuator dynamics are known, the interactions between the control allocation algorithm and the actuator dynamics working on the aircraft body become more complex, requiring a more sophisticated control allocation method. Actuators can have different response times, e.g. a fast actuator can be used to achieve fast transient response, while slow actuators can be used for steady state or trimmed flight, to improve power efficiency. A Model Predictive Control (MPC) allocation scheme will be able to optimally make use of such properties.

Sophisticated dynamic actuator models may be incorporated by using the MPC framework to solve the constrained control allocation problem [6],[7],[8],[9]. MPC is an optimization-based control algorithm which can be used in control allocation, beeing able to handle actuator dynamics as well as actuator saturation. MPC utilizes a model of the plant in predicting outputs and states, where in control allocation this model describes the actuator dynamics. Because of the predictive nature of the controller, the calculated control can pre-act to the actuator system dynamics to improve performance.

Implementing the numerical optimization problem for the optimal control allocation in real time is a challenging task and has been approached in a number of ways - a few of which are discussed in this paragraph. There is a limited amount of computing time available between successive samples, and the control allocation module is required to have a solution available when this time ends. Instead of demanding that the optimal control allocation is computed exactly at each sample, the dynamic online optimization approach in [10] will at each time instant move in the direction towards an optimal control allocation, but optimality is achieved only asymptotically. The method is extended to the case with actuator dynamics in [11]. While the dynamic online optimization approach reduces the online computational requirements, one may also use multi-parametric programming to pre-compute an explicitly represented piecewise affine solution function. The remaining online computations correspond to the evaluation of a piecewise linear function resulting from multi-parametric programming and explicit MPC [12], [13]. While this is highly attractive from the online processing point of view, its memory consumption and offline processing does not scale very well - in particular when considering control efficiency matrices that are time- or state-dependent due to nonlinear or time-varying characteristics like in fault tolerant control allocation [14]. Online optimization using off-the-shelf or customized quadratic programming (QP) solvers are studied in the context of linear actuator and effector models in [15], [16], [17], [18]. For nonlinear effector models, the use of sequential quadratic programming is proposed in [19].

In this thesis another, fairly new, approach is pursued. A family of highly customized QP solvers that are automatically generated using CVXGEN [20], [21] are employed to solve MPC-based dynamic control allocation problems. CVXGEN has the unique feature that the C code of the customized solvers is completely standard and standalone, i.e. portable, and extremely efficient since the key structural properties of the QP problem are exploited in the automatic code generation. This leads to code with only static data structures which is almost branch-free, and where for-loops are rolled out for efficiency, leading to deterministic execution on pipeline processor architectures. Performance improvement also comes for low software overhead as the CVXGEN targets small-scale problems, in some contrast to most off-the-shelf solvers that target large-scale problems. Orders of magnitude faster execution compared to state-of-the-art off-the-shelf solvers have been reported on test problems, including MPC problems [20],[21]. This makes it interesting to study CVXGEN's performance in challenging control allocation problems that are of relatively small scale compared to typical MPC problems.

This thesis builds on the author's previous project work on control allocation, where several of the classical methods were implemented and tested on a 6DOF missile model [22]. In the project work actuator dynamics were ignored, and the present thesis will compare the performance of the most promising classical methods against that of the MPC allocation, this time taking actuator dynamics into consideration. A new missile effector configuration where actuators have different dynamics is developed for this purpose.

The main ideas and results of this thesis are included in the article "Dynamic Model Predictive Control Allocation using CVXGEN" by Hanger et al. This article is aimed for submission at the Ninth IEEE International Conference on Control and Automation in Santiago, Chile. The article is included in Appendix E.

The paper is organized as follows. Chapter 2 introduces some background material relevant to the thesis. In Chapter 3, the control allocation methods which are employed are presented in detail. In Chapter 4, the missile model used when testing control allocation is derived. Chapter 5 presents the test setup, where implementation issues are discussed and a test plan is put forward. Chapter 6 presents the results of the testing, and these are discussed in Chapter 7. Chapter 8 holds some concluding remarks, aswell as a look at some potential further work.

Chapter 2

Background

This paper focuses on testing control allocation methods on a missile in flight. In order to implement and test the control allocation methods, a framework must be built, e.g. to provide the control allocation module with sensible inputs. Because of this some background material is presented. Some of the concepts and components that are presented will be part of the framework used while testing.

2.1 Flight Control Systems

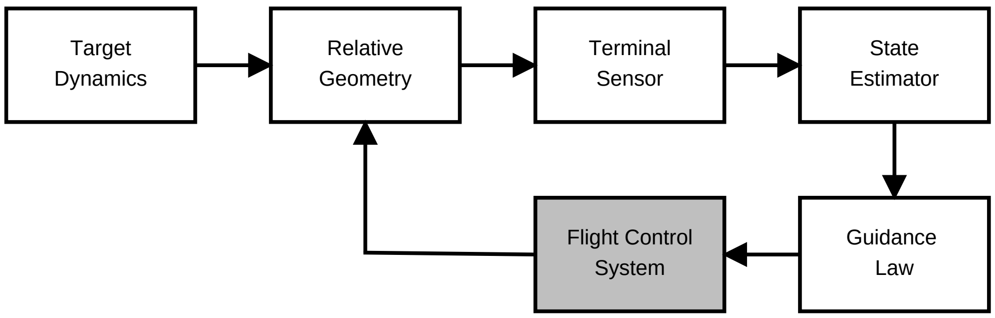

Figure 2.1: The flight control system as part of the homing loop.

The flight control system is the key element that allows a missile to follow the desired steering commands. This system is typically an element in a homing loop. In a human controlled aircraft, there is no need for a homing loop, as all control commands come from the pilot inputs like a stick or yoke, and pedals. Because a missile is an autonomous aircraft, the control commands must be automatically generated, thus the need for a homing loop. A simplified diagram of a generic homing loop is shown in Figure 2.1. A short description of the various blocks follow.

The missile motion combines with the target motion to form the geometry between the two, in the Relative Geometry block. The terminal sensor measures the line-of-sight (LOS) angle[1], which is fed to the state estimator. This block will in turn estimate the LOS angular rate, often by means of a Kalman Filter. The guidance law block takes the LOS angular rate as input, and provides a steering command to the flight control system, completing the loop.

In this thesis' case, the goal of the guidance will not be hitting a target, but performing aerial maneuvers to stress and test the control allocation module within the flight control system block. A typical maneuver is applying reference steps in horizontal and vertical position where the amplitude can be increased incrementally, requiring larger and larger control commands, increasingly making the CA module's job more challenging. Because of this, the actual setup will be much simpler than the one in Figure 2.1. In fact, only the guidance law and flight control system blocks will be utilized, as there will be no need for the others.

Turning the attention to the flight control system block, its basic elements are shown in Figure 2.2.

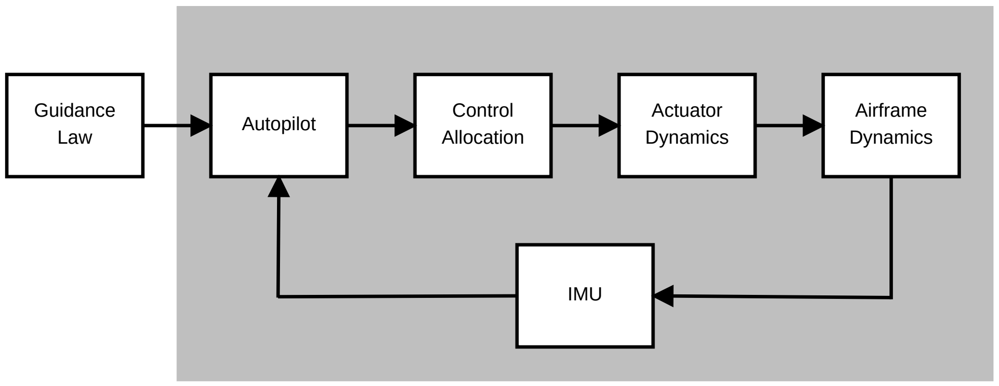

Figure 2.2: The elements of the flight control system.

The inputs to the flight control system come from the guidance law. This input needs to be tracked to perform the desired maneuver. Further into the loop, the autopilot takes the guidance command and feedback measurements and enter them into a set of mathematical equations. The resulting output of these equations is called the virtual control. More material on autopilots is presented in Section 2.2.

The control allocation module will take the virtual control command from the autopilot, and distribute the control among the available actuators (more on control allocation in Section 2.3 and Chapter 3). The actuators will react to the control signal in accordance to their dynamics described in the actuator dynamics block, in turn creating moments about the missile according the equations of motion in the

[1]The angle between the inertial reference and the missile-to-target line-of-sight vector is called the LOS angle.

airframe dynamics block. Lastly, an Inertial Measurement Unit[2] (IMU) will provide the feedback measurements to the autopilot.

The specifics of the actuator dynamics and the missile airframe dynamics are presented in detail in Chapter 4. Since all measurements made in an IMU are readily available from the missile model, there is no real need for an IMU block, as they can be fed back to the autopilot directly from the missile model. Of course, in a real application such measurements must be made in an IMU, so it is reasonable to display it as a part of the flight control system.

2.2 Autopilot

The missile autopilot has an important job in the flight control system. Using the reference command from the guidance law, it utilizes a set of mathematical equations to calculate the virtual control commands fed to the control allocation module. In essence, it is the part of the system responsible for making the missile follow the reference trajectory. Missile autopilots are most often split into two parts - longitudal and lateral.

A longitudal autopilot will control the pitching motion and therefore also the flight height of the missile. Such an autopilot will take a height reference from the guidance law, and by mathematical calculation create the pitching command which will make the missile move as desired.

The lateral autopilot is responsible for making the missile turn. There are two main groups of lateral autopilots, skid-to-turn (STT) and bank-to-turn (BTT). This paper will use a BTT autopilot. This is because a missile with an asymmetrical cross section (which most modern missiles have) will have larger acceleration capabilities in the pitch plane than the yaw plane, and therefore needs to roll to maximize their maneuverability. Another moment speaking for BTT autopilots is that modern missile engines often require small sideslip angles.

BTT autopilots can be designed using many techniques, e.g. PID control, Linear Quadratic Control, backstepping and feedback linearization [23].

2.3 Control Allocation

This section presents some background material on control allocation in general, since this subject is the main focus of the thesis.

[2]An IMU is an electronic device that measures an aircraft's velocity, orientation, and gravitational forces. This is done using a combination of accelerometers and gyroscopes measuring accelerations rates and rotational rates, respectively.

2.3.1 Introduction

Adding a control allocation module essentially splits the control design into two separate parts; a control law for generating the desired control variables (also known as the virtual control), and the control allocation part for the distribution of control power. This has many benefits, some listed in [5] include easy reconfiguration in case of actuator change, separated regulation tuning, and lastly that the control allocation method can be arbitrary. Because of this last fact there exists a lot of different control allocation methods, ranging from simple to complex. A few of these methods are described in Chapter 3, and their performance will be evaluated in this thesis.

An increased number of control effectors compared to the number control variables naturally leads to the need for a control allocation scheme. This is because the relationship between individual effectors and which axis it generates moments about becomes unclear, as some effectors can be used to generate moments about multiple axes. Multiple combinations of the available control effectors can generate the same desired moment in roll, pitch and yaw, bringing redundancy into the control system. The redundancy can be exploited in a number of ways. In addition to performing their primary guidance control tasks, the actuators may also be controlled to perform some secondary objective. Examples include fuel minimization, drag minimization, life-cycle optimization or radar cross section minimization. Another advantage is that one also has some degree of backup control in case of actuator failure or damage. The control allocation module's job is primarily to generate the actuator commands that satisfy the virtual control command, and if there is excess control power, this should be made use of in one of the mentioned ways.

2.3.2 Control Allocation Problem Formulation

Consider the equation

$$B\delta = u \tag{2.1}$$

where u is the virtual control, $B \in \mathbb{R}^{m\times n}$ is the control effectiveness matrix, δ is the control vector, n is the number of control effectors, and m is the number of axes to control. Aerospace applications of control allocation usually has roll, pitch and yaw moments as the virtual control input u, making $m = 3$. The control allocation problem can be described as finding a δ satisfying (2.1), given B and u.

The control effectiveness matrix B varies as a function of Mach number and angle of attack, but is usually assumed constant at each sample instant for simplicity. In addition, the control vector δ is usually bounded:

$$\delta_{min} \leq \delta \leq \delta_{max} \tag{2.2}$$

making the control allocation problem yet more challenging.

2.3.3 Optimality and the Attainable Control Set

A system with four control effectors, i.e. $\delta \in \mathbb{R}^4$ and a virtual control input $u \in \mathbb{R}^3$ will be considered when explaining these concepts. This is for simplicity - the results could also be generalized for an arbitrarily dimensioned system.

The attainable control set Ω is the set of all attainable control vectors δ. The space of the attainable control vectors create a 4-dimensional box (since $\delta \in \mathbb{R}^4$), where the actuator bounds (2.2) create the outer limits of the box. This boundary is usually denoted $\partial(\Omega)$. The space Ω can be formulated by

$$\Omega = \Big\{ \ \delta \mid \delta_{min} \leq \delta_i \leq \delta_{max}, \ \forall \ i \in \{1,2,3,4\} \ \Big\} \tag{2.3}$$

The image of Ω under the control effectiveness map $B \in \mathbb{R}^{3\times 4}$ creates a 3-dimensional space denoted Φ. Since for most control allocation problems, $n > n$, the map from Ω to Φ is not one-to-one. Control allocation methods try to determine controls $\delta \in \Omega$, given some desired virtual control input $u \in \Phi$. Methods which can allocate controls for all vectors in Φ are termed optimal. Methods which allocate over a smaller set, i.e. a subset $\Pi \subset \Phi$, are termed non-optimal.

Because of their ability to allocate controls for all vectors in Φ, optimal methods generally give better results when tracking a virtual control input u. The reasons for choosing non-optimal methods are that they in general are simpler to implement and have faster execution. [24] presents a method for finding the attainable control set. [25] reviews the geometry of the attainable objects.

2.4 Model Predictive Control

Model Predictive Control (MPC) is a control algorithm based on solving a finite horizon open-loop optimization problem at each sampling instant. Such controllers rely on an internal dynamic model of the process, used to predict the behaviour of the system. The system to be controlled is usually described (or more commonly, approximated) by one or more ordinary differential equations (ODEs). Because MPC is a discrete algorithm, the ODEs are usually converted to discrete difference equations. The MPC objective cost function is often on the form

$$V(k) = \sum_{i=1}^{T} Q(i)\Big(\hat{x}(k+i \mid k) - r(k+i \mid k)\Big)^2 + R(i)\Big(\hat{u}(k+i \mid k)\Big)^2 \tag{2.4}$$

where $\hat{x}$ is the estimated state, r is the reference trajectory, $\hat{u}$ is the optimal control sequence and T is the prediction horizon length. The first term in $V(k)$ represents that the state x should track the reference r. The various states are weighed with $Q(i)$ to reflect relative tracking importance between states. The second term in the cost function will penalize the use of control input u, with weighing factors $R(i)$.

The main advantage of MPC is its ability to handle constraints. Both input constraints (bounds on u), like the saturation of an actuator, and state constraints (bounds on x), like keeping the level of a fluid between bounds, can be handled with ease.

The system model is initialized with the most recent sample of the states, and the controller uses the combination of these and the internal model to optimize the objective cost function such that the cost is minimized and all constraints are honored. The controller will only use the first step of the calculated control sequence as plant input. This optimization based approach is the main difference from conventional control strategies, where a pre-computed control law is usually applied for each sample time. The basics of MPC are displayed in Figure 2.3.

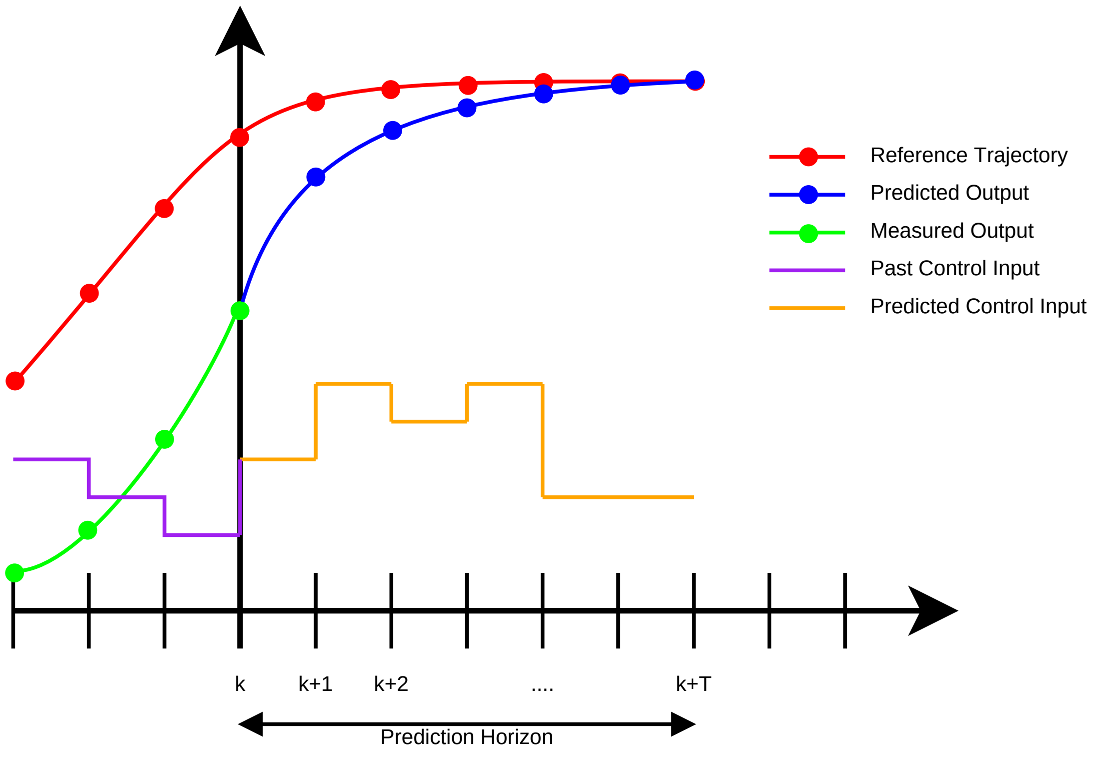

Figure 2.3: A discrete MPC scheme.

An explanation of Figure 2.3 follows. At time k the current plant state is sampled. The cost function is minimized while honoring constraints, leading to a optimal control strategy for the horizon interval $[\ k,\ \ k+T\ ]$. The predicted optimal output is the blue line which converges towards the red reference, like reflected in the cost function (2.4) . The optimal control input is shown in orange.

The control strategy explores state trajectories emanating from the sampled starting point, and finds the one minimizing cost. Only the first control step is applied to the plant, and the plant state is then sampled again and the same procedure is repeated, giving a new control step and a new predicted state path. Because the horizon keeps beeing pushed forward, MPC is sometimes called receding horizon control (RHC).

The way MPC handles constraints allows for plant operation closer to the optimal working point. It has been widely applied in the chemical and petroleum industries because accounting for constraints is especially important in these applications. The MPC strategy is also expected to behave well in a control allocation perspective, because of its predictive nature and ability to handle actuator dynamics. Given an estimate of the control allocated craft's future trajectory, it enables the craft to utilize actuators with different time constants to their full extent. This also opens possibilities to restrict the use of costly actuators when not neccesary. This cost can be either connected to e.g. a power/fuel consumption or radar cross section concern. Model Predictive Control Allocation will be discussed further in Chapter 3.

For a detailed description of Model Predictive Control, see [26].

2.5 Missile Effector Configurations

A missile's effector configuration plays a major role in the way it is controlled. Some background on effectors is presented in this section, along with two configurations, one of them tailored to be utilized with the MPCA.

A missile has a set of actuators, which control a set of corresponding effectors (control surfaces). The effectors are usually in the form of fins or flaps. When actuated, these create forces about the missile body, causing it to move and therefore making the missile able to steer and follow a desired trajectory. The actuator is the controllable part of the actuator-effector setup, and the one considered in mathematical derivation. Actuator control is denoted by the symbol δ_i, where the subscript denotes the individual actuator's control. The set of all actuator controls forms the control vector δ.

There are many ways to place the effectors on the missile, but the most usual is to place them at the back end of the missile. Such missiles are categorized as tail-controlled missiles. One regular effector configuration is placing four fins evenly spread on the missile tail. If set perpendicular to the horizontal and vertical planes of the missile, they resemble a + and this is subsequently called the "+ configuration". If shifted by 45°, the setup is called "x configuration". The latter configuration will be the basis for this thesis' test configuration. This decision is made based on the results of the previous project work, where the x configuration was found preferrable to the + configuration in terms of performance and redundance utilization.

An x configuration missile is displayed in Figure 2.4. In the figure, the slanted thick lines represent the missile's wings. The rotation of the fins is chosen such that deflection with the trailing edge down is positive.

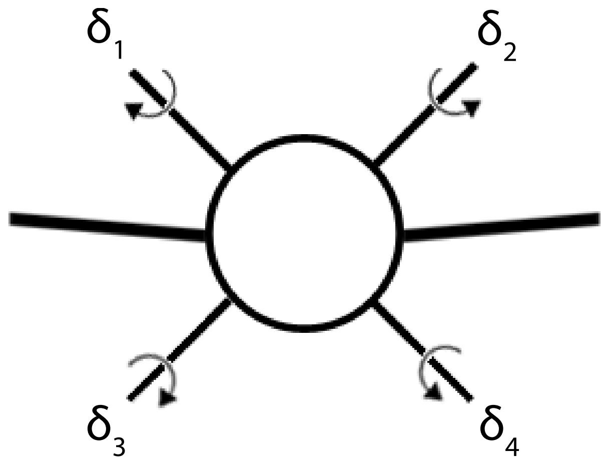

Figure 2.4: Missile with four fins in x configuration. View from behind.

For such a missile, the control commands $\delta_i,\ i \in \{1,2,3,4\}$ can be combined to form the axis specific control commands δ_R, δ_P and δ_Y for roll, pitch and yaw, respectively:

$$\delta_R = \delta_1 - \delta_2 + \delta_3 - \delta_4 \tag{2.5}$$
$$\delta_P = \delta_1 + \delta_2 + \delta_3 + \delta_4 \tag{2.6}$$
$$\delta_Y = -\delta_1 + \delta_2 + \delta_3 - \delta_4 \tag{2.7}$$

The main focus of this thesis is to use a Model Predictive Control Allocation scheme, which account for actuator dynamics, and is able to make use of their different dynamic ranges. Because of this, another effector configuration is proposed. It builds on the x configuration (Figure 2.4), but adds ailerons to the missile wings, see Figure 2.5. The ailerons are defined to have positive rotation with the trailing edge down. This new configuration will be used when testing control allocation methods.

The four "regular" tail fins are controlled by fast actuators, while new wing ailerons are controlled by slower actuators. These ailerons will be more efficient than the tail fins in the roll plane, while having similar and lesser efficiency in the pitch and yaw planes, respectively. The motivation for using this configuration is that the wing ailerons can be used while in trimmed flight, i.e. "calm" conditions, while the tail fins are added on when needed in agile flight. Because of the different dynamic authorities of the actuator groups, MPCA is expected to handle this configuration well.

The control commands $\delta_i,\ i \in \{1 \ldots 6\}$ can be combined to form the axis specific control commands δ_R, δ_P and δ_Y :

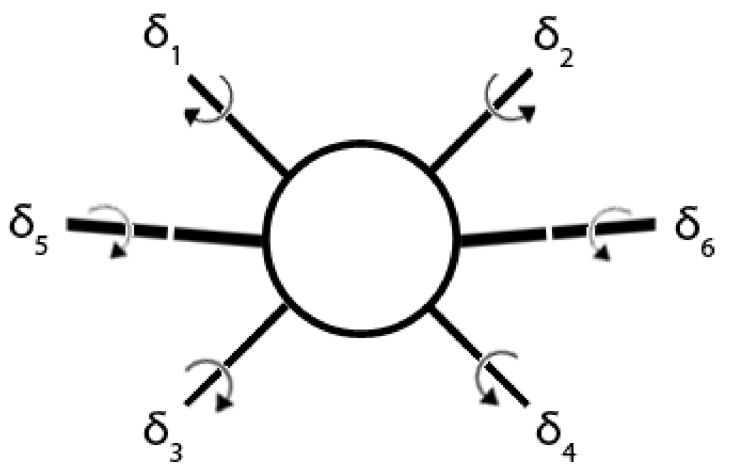

Figure 2.5: Missile with four fins and wing ailerons. View from behind.

$$\delta_R = \delta_1 - \delta_2 + \delta_3 - \delta_4 + \delta_5 - \delta_6 \tag{2.8}$$
$$\delta_P = \delta_1 + \delta_2 + \delta_3 + \delta_4 + \delta_5 + \delta_6 \tag{2.9}$$
$$\delta_Y = -\delta_1 + \delta_2 + \delta_3 - \delta_4 - \delta_5 + \delta_6 \tag{2.10}$$

2.6 CVXGEN

Part of this thesis is using and testing the new CVXGEN convex optimization solver, released in 2010 by Jacob Mattingley and Stephen Boyd [27]. Testing this solver and comparing it with others is interesting because it is state-of-the-art, and its applications may be used for both prototyping and real-time use.

Convex optimization is widely used because it has a number of applications, e.g. control, circuit design and networking [27]. Such problems can be solved reliably and efficiently with well developed methods and tools [28], [29]. Parser solvers like CVX [30] and YALMIP [31] accepts a convex optimization problem specified in high-level language, but their solve times are in the scale of seconds or minutes, which makes them unable for use in real-time systems. They also require extensive libraries and have large footprints. However, in the development phase of algorithms or methods based on convex optimization, they can be a good choice, as run-time and footprint are usually not of great concern at an early stage (no real-time requirements).

Conventionally, the step from a general purpose parser solver to a specialized high-speed solver requires significant development time, extensive modeling and specialist knowledge of optimization and numerical algorithms. The work is also often done by hand, limiting their applications.

CVXGEN is a software tool that automatically generates C-code that compiles into a convex optimization solver, from a high level language specification. The C-code of the customized solvers is completely standard, standalone and extremely efficient because key structural properties of the QP problem are exploited. This leads to code with only static data structures which is almost branch-free, with deterministic execution on pipeline processor architectures. The generated solvers are very reliable and robust [27], but also fast compared to parser solvers. With solve times in microseconds or milliseconds, the generated solvers lend themselves to implementation in real-time applications with operation speeds in Hz or kHz. CVXGEN's footprint is also simple, generating a flat, library-free solver.

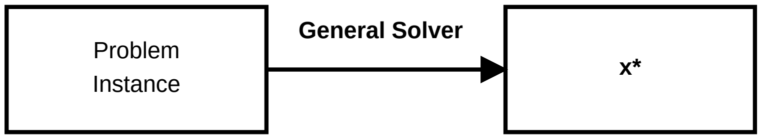

Figure 2.6: General purpose parser solver structure. Turns a single problem instance into a single optimal point.

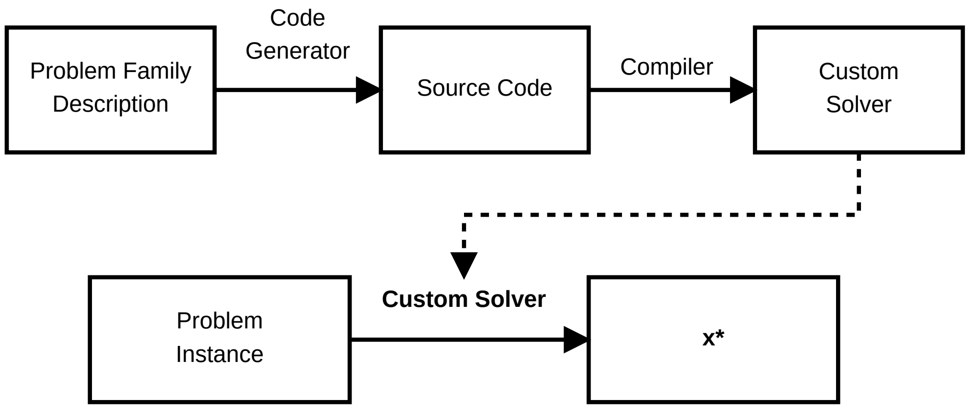

Figure 2.7: Automatic code generator solver structure. Provides optimal points for many different problem instances.

The CVXGEN solver is currently available through a web interface on the project's web page `http://www.cvxgen.com`. An optimization problem specification can be entered through a MATLAB-like programming language on the web interface. Syntax specifics can be found in CVXGEN's documentation [27]. The problem is entered through a fixed and structured setup, specifying problem dimensions, parameters, variables, cost function and constraints.

The custom C solver is automatically generated on the web interface by the click of a button. After compilation it is available for download as a zipped archive. In addition to C code, a MATLAB interface is also available, making the custom solver available for e.g. prototyping and initial testing in the MATLAB environment. The

MATLAB version will be utilized in this thesis.

The downloaded solver is used by calling a pre-made function, with the problem instance's specific parameters as function input. Solver settings can also be entered when calling the solver. After the call, the solver solves the convex optimization problem with respect to the instance parameters, and outputs the globally optimal variables.

CVXGEN lends itself naturally to MPC problems, see [21] for a detailed overview.

Chapter 3

Control Allocation Methods

The methods presented in this chapter are the Redistributed Pseudoinverse, Linear Programming and Model Predictive Control Allocation. All three methods will be implemented as part of a flight control system on a missile model. There are several other methods, for an overview, see e.g. [32], [8], [33], [15] or [2].

3.1 Redistributed Pseudoinverse

If $m = n$, the control efficiency matrix B is square, and the control allocation problem can be solved with by a simple matrix inverse:

$$\delta = B^{-1}u \tag{3.1}$$

For most control allocation applications, $n > m$, so an inverse is not possible. In this case, a pseudoinverse can be used. The pseudoinverse solution is the 2-norm solution to the control allocation problem (2.1). It can be formulated as a minimization problem:

$$\min_{\delta} \frac{1}{2}\delta^T\delta \tag{3.2}$$

subject to (2.1).
Solving this problem explicitly [8] leads to the standard Moore-Penrose[1] pseudoinverse:

$$\delta = B^T(BB^T)^{-1}u \tag{3.3}$$
$$\delta = B^{\dagger}u \tag{3.4}$$

where † denotes the pseudoinverse operation, $B^{\dagger} = B^T(BB^T)^{-1}$.

[1]The Moore-Penrose pseudoinverse is the most widely known type of matrix pseudoinverse, named after E. H. Moore and R. Penrose, which independently described it in 1920 and 1955, respectively.

The regular pseudoinverse approach can be extended by adding a weighting matrix $W \in \mathbb{R}^{n \times n}$ to the minimization problem (3.2), as done in e.g. [8]. The corresponding minimization problem becomes

$$\min_{\delta} \frac{1}{2} \delta^T W \delta \tag{3.5}$$

subject to (2.1).
Solving for δ gives the weighted pseudoinverse

$$\begin{aligned} \delta &= W^{-1}B^T(BW^{-1}B^T)^{-1}u && (3.6) \\ \delta &= B_W^{\dagger}u && (3.7) \end{aligned}$$

where $B_W^{\dagger} = W^{-1}B^T(BW^{-1}B^T)^{-1}$.

The redistributed pseudoinverse is an iterative version of the regular pseudoinverse, which can guarantee that the actuator constraints are honored. The method also goes by the name Cascaded Generalized Inverse (CGI), and is discussed in e.g. [8], [32]. It allocates the control as a series of weighted pseudoinverse steps, though regular pseudoinverse also can be used. Only the weighted version will be presented, but it is easily transformed to regular inverse by taking $W = I$.

In the weighted case, the method starts off by solving a weighted pseudoinverse, $\delta = W^{-1}B^T(BW^{-1}B^T)^{-1}u$. If any of the actuators saturate in this solution, they are set to their constrained limits and removed from the next iterate. This can be done by zeroing out or removing their corresponding columns in B and W. Then the new problem $\delta' = W'^{-1}B'^T(B'W'^{-1}B'^T)^{-1}u$ is solved, where B' and W' are the new control effectiveness and weighting matrices without the saturated controls from the previous step. If no control saturated during the first iteration, the solution is kept and the method terminates.

If on the other hand the first iteration had saturated actuators, the iterative process continues until no new actuators saturate, or all actuators saturate. This way, actuator constraints are honored, while at the same time the control power is distributed amongst the remaining actuators. The number of iterates is bounded by n, so an upper convergence limit is always known. This makes the redistributed pseudoinverse a deterministic method in turns of run-time, but also fast since n usually is a low number (under 10) for regular aerospace applications.

Though the method executes slower than only performing a single pseudoinverse, it can guarantee not exceeding actuator bounds. It should be noted that this method has no way of optimizing the control allocation with regard to some secondary objective when there may be excess control power available.

3.2 Linear Programming

The control allocation problem (2.1) can also be solved using linear programming (LP) techniques. This method can exploit the control redundancy by optimizing a secondary objective if the primary one is feasible. The standard constrained control allocation optimization problem is

$$\min_{\delta} J(\delta) \tag{3.8}$$

subject to (2.1) and (2.2). $J(\delta)$ represents a linear scalar-valued function of the control actuator positions to be minimized. This scalar-valued function can be formulated in several ways, one of which is the error minimization formulation

$$\min_{\delta} J = ||B\delta - u||_1 \tag{3.9}$$

subject to (2.2).

The error minimization formulation has a favorable property when the desired u is unattainable. While other formulations offer no guidance as to which solution to use in this case, this formulation picks it by minimizing the objective error. The error minimization formulation can be converted to a standard[2] linear programming problem [34]:

$$\min_{\delta} J = [0 \cdots 0 \quad 1 \cdots 1] \begin{bmatrix} \delta \\ s \end{bmatrix} \tag{3.10}$$

subject to

$$\begin{bmatrix} s \\ -\delta \\ \delta \\ -B\delta + s \\ B\delta + s \end{bmatrix} \geq \begin{bmatrix} \mathbf{0} \\ -\delta_{max} \\ \delta_{min} \\ -u \\ u \end{bmatrix} \tag{3.11}$$

where $s \in \mathbb{R}^m$ is a vector of slack variables. This standard linear program can be solved using ordinary linear programming solvers.

Now, if the linear program is solved and the solution is such that $J = 0$, the virtual control u is attainable, and some "excess" control power may be available. In this case, a secondary objective can be optimized for, e.g. as mention earlier, minimizing fuel consumption. The ability to optimize with respect to a secondary problem comes from the system's actuator redundancy - multiple solutions to the control allocation problem may exist, and solving a secondary problem may pick one of them which is optimal in some manner. The secondary problem [8] can be formulated as follows

[2]Standard Linear Programming canonical form.

$$\min_{\delta} J = \mathbf{W}_u^T s \tag{3.12}$$

subject to

$$\begin{bmatrix} s \\ -\delta \\ \delta \\ -\delta + s \\ \delta + s \end{bmatrix} \leq \begin{bmatrix} \mathbf{0} \\ -\delta_{max} \\ \delta_{min} \\ -\delta_p \\ \delta_p \end{bmatrix} \tag{3.13}$$

where δ_p is the control effector vector found in the primary problem and $\mathbf{W}_u^T \in \mathbb{R}^{m\times 1}$ is a weighting matrix chosen to reflect what kind of secondary objective to optimize for. One example is to have the actuators contributing most to the drag penalized more, this in turn reducing fuel consumption. Another possibility is to penalize the actuators most subject to wear, to extend actuator life.

Another secondary objective function can be formulated as [35]

$$\min_{\delta} J = \mathbf{W}_p^T(\delta - \delta_p) \tag{3.14}$$

subject to (2.1) and (2.2). These constraints make sure the primary objective is reached. Here $\mathbf{W}_p^T$ is a weighting matrix allowing for flexibility in the optimization, and δ_p is a preferred control vector, which also could be zero. This minimizes the actuator position from a preferred position, e.g. one that minimizes radar cross section.

The primary and secondary objective can also be merged together in what is called a mixed optimization problem. A combination of the primary objective function and the most recently proposed secondary objective function is

$$\min_{\delta} J = ||B\delta - u||_1 + \alpha||\mathbf{W}_p^T(\delta - \delta_p)||_1 \tag{3.15}$$

subject to (2.2). Here α is a scaling factor weighing the relative importance between objectives. α is usually small ($<< 1$) to reflect the importance of the primary objective. [15] converted this mixed optimization problem into a standard linear program, which can be solved with any linear program solver.

3.3 Model Predictive Control Allocation

The previously described methods assume a static relationship between actuator input and response, not taking different dynamic authorities of the actuators into account. Actuators can have different time constants and delays, i.e. a fast actuator can be used for good transient response, while slow actuators can be used for steady state or trimmed flight. This can improve the power efficiency of the missile. Most control allocation methods are unable to exploit such properties.

Section 2.4 discussed the optimization-based control algorithm Model Predictive Control (MPC). This algorithm can be used in control allocation and is able to handle actuator dynamics aswell as actuator saturation [6], [7], [9], [8]. In this section, a Model Predictive Control Allocation scheme is developed to be used in this thesis.

MPC uses a model of the plant in predicting outputs and states, and in control allocation this model is that of the actuator dynamics. Because of the predictive nature of the controller, the calculated control can pre-act to the actuator system dynamics to improve performance.

Assuming that all control effectors have dynamics modelled as second order systems, the model for effector i will be on the form

$$\dot{\delta}_i = A_{\delta_i}\delta_i + B_{\delta_i}\delta_{cmd,i} \tag{3.16}$$

where $\delta_{cmd,\ i}$ is the commanded control input, and δ_i is the actual actuator response. Note that δ_i is a two-dimensional vector, i.e. $\delta_i \in \mathbb{R}^2$, where $\delta_i(1)$ represents the actuator position, and $\delta_i(2)$ represents actuator velocity. The actuator position is the most relevant, and will be the measure utilized in this paper. Actuator velocity and belonging constraints can also be of interest, and is easily added to the MPCA formulation. More on actuator modelling is presented in Section 4.8.

For a system with n effectors, the set of actuators can be combined to form a full actuator model:

$$\begin{bmatrix} \dot{\delta}_1 \\ \vdots \\ \dot{\delta}_n \end{bmatrix} = \begin{bmatrix} A_{\delta_1} & 0 & \cdots & 0 \\ 0 & A_{\delta_2} & \cdots & 0 \\ \vdots & & \ddots & \\ 0 & & \cdots & A_{\delta_n} \end{bmatrix} \begin{bmatrix} \delta_1 \\ \vdots \\ \delta_n \end{bmatrix} + \begin{bmatrix} B_{\delta_1} \\ \vdots \\ B_{\delta_n} \end{bmatrix} \begin{bmatrix} \delta_{cmd,1} \\ \vdots \\ \delta_{cmd,n} \end{bmatrix} \tag{3.17}$$

Often such a system is only denoted by

$$\dot{\boldsymbol{\delta}} = A_\delta \boldsymbol{\delta} + B_\delta \boldsymbol{\delta}_{cmd} \tag{3.18}$$

The MPC control allocation problem is posed as follows: For the constrained system

$$\begin{aligned} \dot{\boldsymbol{\delta}}(t) &= A_\delta \boldsymbol{\delta}(t) + B_\delta \boldsymbol{\delta}_{cmd}(t) \\ \tau(t) &= B\boldsymbol{\delta}(t) \\ \delta_{min} \leq &\,\boldsymbol{\delta} \leq \delta_{max} \end{aligned} \tag{3.19}$$

find $\boldsymbol{\delta}_{cmd}$ such that $\tau(t)$ tracks $u(t)$ as closely as possible, $u(t)$ beeing the virtual control vector.

Note that δ_{min} and δ_{max} can be scalars or vectors containing the various actuators' constraints. If an actuator has no constraints, the corresponding entries in $\boldsymbol{\delta}$, δ_{min} and δ_{max} is simply removed. Since $\delta_i \in \mathbb{R}^2$, position and velocity constraints are placed on odd and even numbers in $\delta_{min/max}$, respectively.

This model is used to predict the commanded control inputs $\hat{\boldsymbol{\delta}}_{cmd}$, the control $\hat{\boldsymbol{\delta}}$ and outputs $\hat{\tau}$,

$$\begin{aligned} \hat{\boldsymbol{\delta}}_{cmd} &= [\ \hat{\boldsymbol{\delta}}_{cmd}(k+1|k),\ \cdots,\ \hat{\boldsymbol{\delta}}_{cmd}(k+T-1|k)\ ] \quad (3.20) \\ \hat{\boldsymbol{\delta}} &= [\ \hat{\boldsymbol{\delta}}(k+1|k),\ \cdots,\ \hat{\boldsymbol{\delta}}(k+T|k)\ ] \quad (3.21) \\ \hat{\tau} &= [\ \hat{\tau}(k+1|k),\ \cdots,\ \hat{\tau}(k+T|k)\ ] \quad (3.22) \end{aligned}$$

where T is the length of the prediction horizon, and k is the current time step.

The MPC algorithm finds the optimal set of $\hat{\boldsymbol{\delta}}_{cmd}$ by minimizing a cost function on the form

$$\begin{aligned} J(\cdot) &= \sum_{j=1}^{T} W_t(j)[\ \hat{\tau}(k+j\ |k) - u(k+j)\ ]^2 \\ &\quad +\alpha \sum_{j=1}^{T-1} \sum_{i=1}^{n} W_u(i)[\ \delta_{cmd,i}(k+j-1)\ ]^2 \quad (3.23) \end{aligned}$$

subject to (3.19).

In (3.23), W_t is a weight matrix weighing the importance of tracking u at time j, W_u weighs how expensive actuator i is to use and α weighs the relative importance between the tracking and control terms. T is the prediction horizon length and n is the number of actuators. Only the first commanded control sample $\hat{\boldsymbol{\delta}}_{cmd}(k+1|k)$ is applied to the actuators, the whole algorithm is repeated when computing the consequtive $\boldsymbol{\delta}_{cmd}$.

A Model Predictive Control Allocation (MPCA) scheme for a re-entry vehicle is presented in [35]. [9] extends the MPCA by proposing an augmentation that guarantees input-to-state (ISS) stability of the overall system while preserving performance. [7] extends the MPCA by providing asymptotic tracking of time-varying control input commands.

3.4 Motivational Test

To illustrate the potential of MPCA, a simple test is conducted. This will only test the input/output performance of the CA module in the flight control system. To judge its performance, the MPCA is compared to a LP version of the same problem.

The system has one input $u \in \mathbb{R}^1$, consisting of a sine with increasing and decreasing frequency. There are two actuators δ_1 and δ_2, both modeled as second order systems (3.16). Actuator 1 will be fast but expensive to use, while actuator 2 will be slow and inexpensive. The actuator coefficients and corresponding cost weight are

$$\omega_{0,1} = 150,\ \zeta_1 = 0.7,\ W_1 = 1 \qquad \omega_{0,2} = 10,\ \zeta_2 = 0.9,\ W_2 = 0.1 \tag{3.24}$$

This means that the CA module should restrict the use of actuator 1, since actuator 2 costs less to use. In addition, actuator 2 will be more efficient than actuator 1, indicated in the control efficiency matrix B,

$$B = [\ 0.3\ 0.8\ ] \tag{3.25}$$

The cost function used to compute the optimal set of actuator commands $\boldsymbol{\delta}_{cmd}$ is

$$J_{MPC} = \sum_{t=1}^{T+1} \Big(\tau(t) - u(t)\Big)^2 + \alpha \sum_{t=0}^{T} \Big(\big(W_1 \delta_{cmd,1}(t)\big)^2 + \big(W_2 \delta_{cmd,2}(t)\big)^2\Big) \tag{3.26}$$

The cost function (3.26) is minimized with respect to the constraint set

$$\boldsymbol{\delta}(t+1) = A_\delta \boldsymbol{\delta}(t) + B_\delta \boldsymbol{\delta}_{cmd}(t), \quad t = 0 \ldots T \tag{3.27}$$

$$\tau(t) = B\boldsymbol{\delta}(t), \quad t = 0 \ldots T+1 \tag{3.28}$$

$$\delta_{min} \leq \boldsymbol{\delta}(t) \leq \delta_{max}, \quad t = 0 \ldots T \tag{3.29}$$

A second order extrapolation is used to predict the virtual control input for the MPCA. A similar LP problem is formulated as comparison to the MPCA:

$$J_{LP} = ||\tau_{cmd}(t) - u(t)||_1 + \alpha\Big(||W_1 \delta_{cmd,1}(t)||_1 + ||W_2 \delta_{cmd,2}(t)||_1\Big) \tag{3.30}$$

subject to

$$\delta_{cmd,min} \leq \boldsymbol{\delta}_{cmd} \leq \delta_{cmd,max} \tag{3.31}$$

where $\tau_{cmd} = B\boldsymbol{\delta}_{cmd}$. A control allocation problem was tested on the MPC and LP control allocation formulations. Simulation values is summarized in Table 3.1. In the actuator response plots, the dashed lines represent actuator bounds. The total cost J_{MPC} and J_{LP} is also compared. Note that only the current sample is used in the calculation of the cumulative value of J_{MPC} (making it an instant cost like for LP).

As seen in Figure 3.1, the MPCA tracks u with little error. The results of using LP CA are less satisfactory, see Figure 3.2. Because this formulation ignores actuator dynamics, it calculates that using only actuator 2 will be sufficient, as seen in Figure 3.4. The MPCA is aware of the actuators different dynamics, and utilizes them both

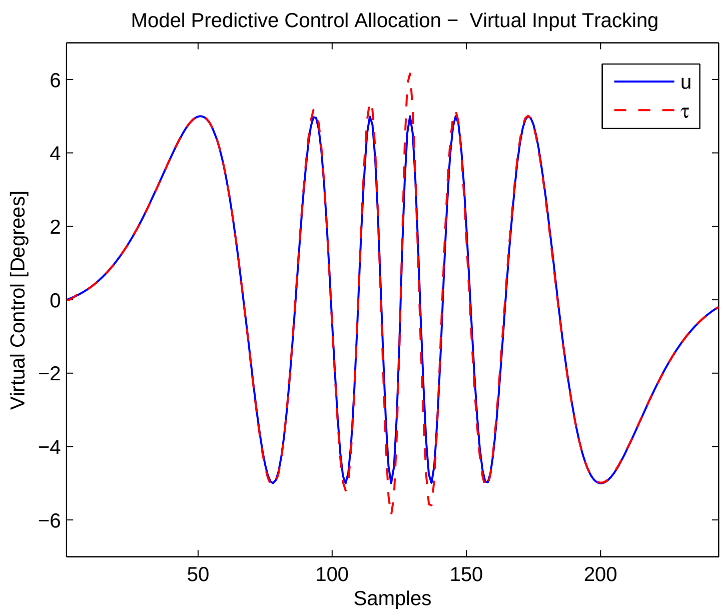

Figure 3.1: MPCA Virtual Control Tracking

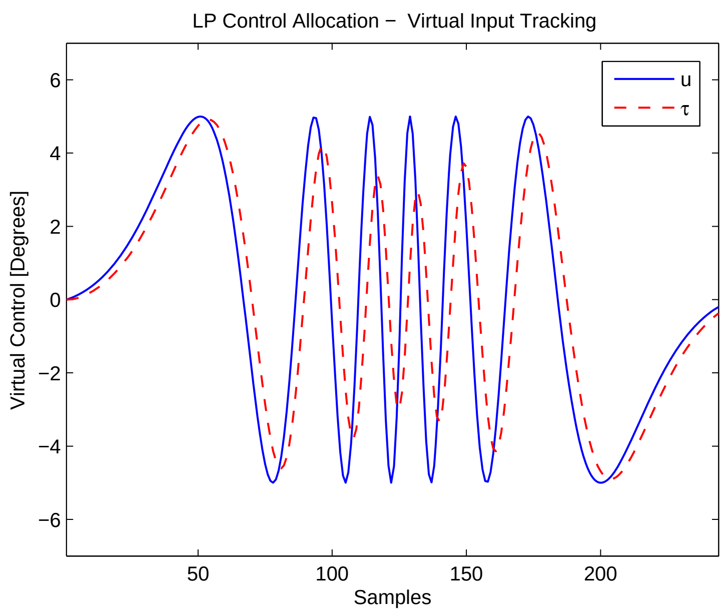

Figure 3.2: LP CA Virtual Control Tracking

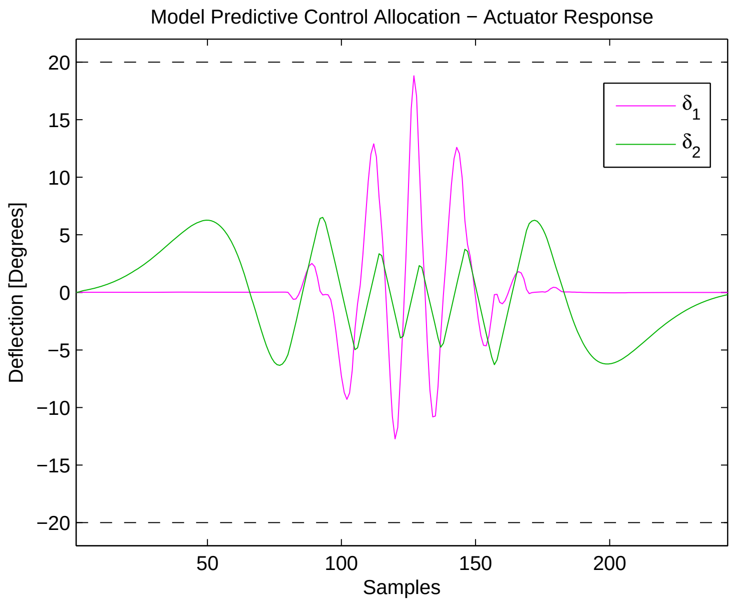

Figure 3.3: MPCA Actuator Response

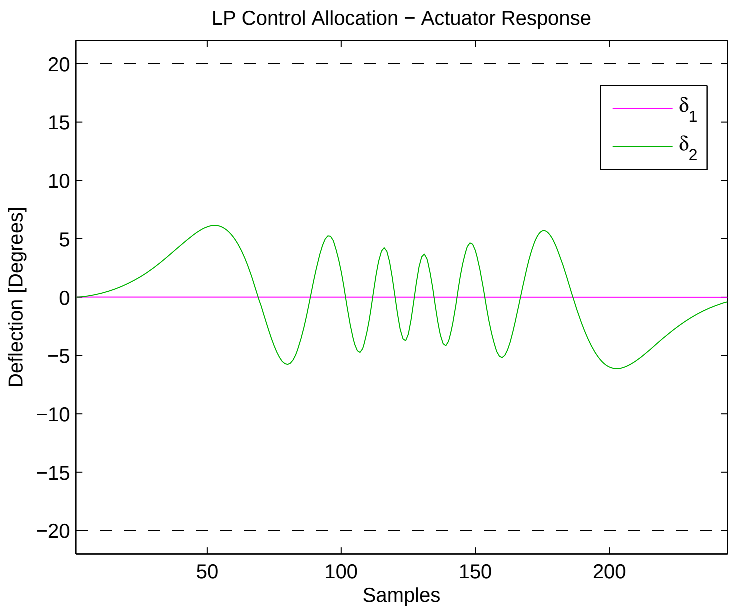

Figure 3.4: LP CA Actuator Response

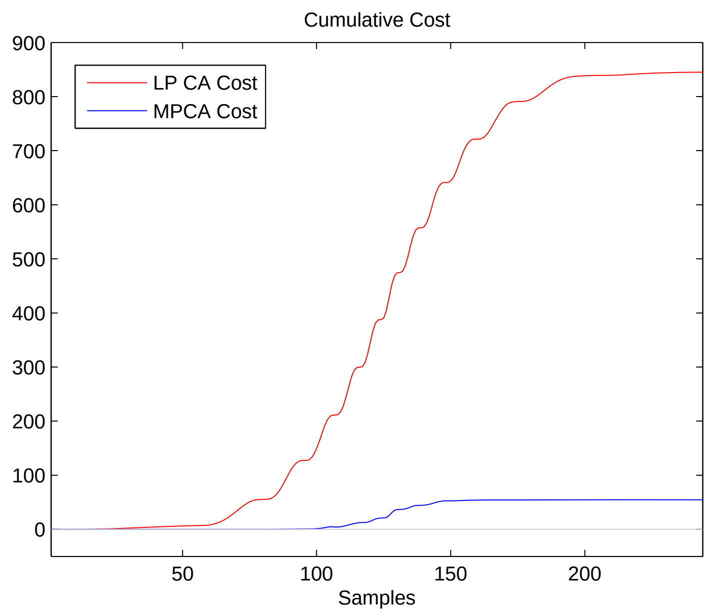

Figure 3.5: Cumulative Cost

α	0.01
δ_{min}	$-20°$
δ_{max}	$20°$
T	5

Table 3.1: Simulation values.

to achieve satisfactory tracking, see Figure 3.3. Lastly, one can look at the accumulated cost in Figure 3.5, which summarizes why taking actuator dynamics into account is beneficial when they span a large dynamic range.

Chapter 4

Missile Model

In this section the missile model will be derived. The model will be used when testing the control allocation methods, and will provide a more realistic setting than just testing control allocation as an input-output block, and more like an actual flight control system setup. This chapter makes use of a wide set of symbols, and the reader is referenced to the missile model nomenclature in Appendix B for descriptions.

4.1 Coordinate Systems

When deriving equations of motion for aircraft, e.g. a missile, several coordinate systems are commonly used to aid the derivation. The most utilized are

- Body Coordinate System
- Wind Coordinate System
- Stability Coordinate System
- Earth Coordinate System

This thesis will mostly consider the body and wind coordinate systems:

Body System

- Center in the missile center of gravity (CG).
- x-axis positive forward, through the missile nose.
- y-axis perpendicular to the body x-axis, and positive to the right of the x-axis.
- z-axis perpendicular to body x-y plane, positive downwards.

The axes of the body system usually have subscript "B".

Wind System

- Center in the missile center of gravity (CG).
- x-axis positive in the direction of the oncoming air (relative wind).
- y-axis perpendicular to the wind x-axis, and positive to the right of the x-axis.

- z-axis perpendicular to wind x-y plane, positive downwards.

The axes of the wind system usually have subscript "W". The body and wind coordinate systems are shown in Figure 4.1, together with the stability axes. This Figure also shows the *angle of attack* α and *sideslip angle* β, defined as

$$\alpha := \arctan(\tfrac{w}{u}) \tag{4.1}$$
$$\beta := \arcsin(\tfrac{v}{v_T}) \tag{4.2}$$

where v_T is the *total speed* $v_T = \sqrt{u^2 + v^2 + w^2}$.

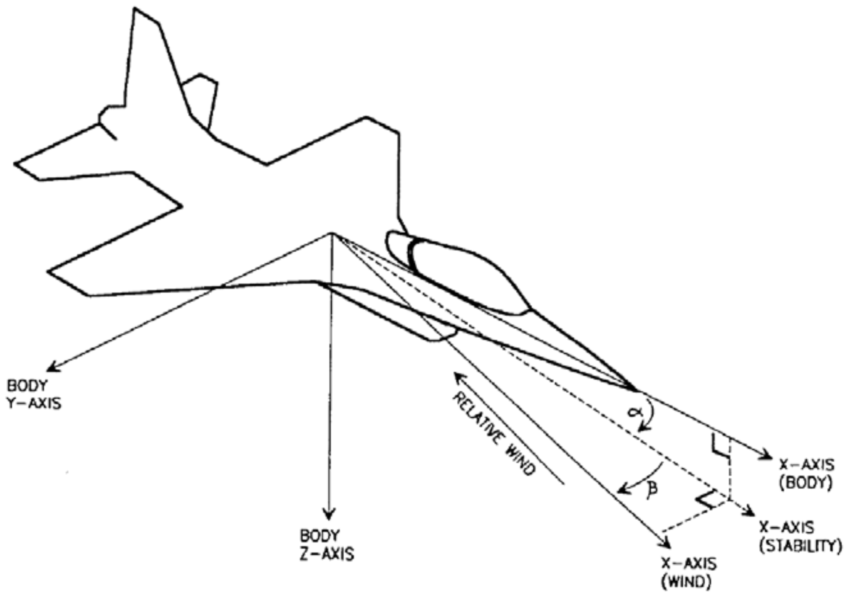

Figure 4.1: Aircraft Coordinate Systems [1]

The model velocity vector is

$$\nu := [\ u\ v\ w\ p\ q\ r\ ]^T \tag{4.3}$$

and the position/attitude vector

$$\eta := [\ x_E\ y_E\ z_E\ \phi\ \theta\ \psi\ ]^T \tag{4.4}$$

where the elements are defined according to the body coordinate system in Figure 4.1. Subscript E denotes Earth Coordinate System. The rotation matrix between the body and wind systems is

$$\mathbf{R}_B^W = \begin{bmatrix} \cos(\alpha)\cos(\beta) & \sin(\beta) & \sin(\alpha)\cos(\beta) \\ -\cos(\alpha)\sin(\beta) & \cos(\beta) & -\sin(\alpha)\sin(\beta) \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \tag{4.5}$$

4.2 Kinematics

The kinematic equations for translation of the body-fixed coordinate system B (with elements like in (4.3)) with respect to a local NED coordinate system can be expressed in terms of Euler angle rotations:

$$\begin{bmatrix} \dot{x_E} \\ \dot{y_E} \\ \dot{z_E} \end{bmatrix} = \mathbf{R}_B^{NED} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \mathbf{R}_{z,\psi}\mathbf{R}_{y,\theta}\mathbf{R}_{x,\phi} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{4.6}$$

This can be expanded to

$$\begin{bmatrix} \dot{x_E} \\ \dot{y_E} \\ \dot{z_E} \end{bmatrix} = \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{4.7}$$

$$\begin{bmatrix} \dot{x_E} \\ \dot{y_E} \\ \dot{z_E} \end{bmatrix} = \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi c\phi s\theta \\ s\psi c\theta & c\psi c\phi + s\phi s\theta s\psi & -c\psi s\phi + s\theta s\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{4.8}$$

The kinematic equations for attitude become

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi / c\theta & c\phi / c\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}, \ c\theta \neq 0 \tag{4.9}$$

For equations (4.7), (4.8) and (4.9), s, c and t represent the $\sin(\cdot)$, $\cos(\cdot)$ and $\tan(\cdot)$ operations, respectively.

4.3 Rigid Body Dynamics

The derivation of the rigid body dynamics is based on and adapted from [36] and [37]. The aircraft rigid-body kinetic equations can be expressed as [38], [39]

$$m(\dot{\nu}_1 + \nu_2 \times \nu_1) = \tau_1 \tag{4.10}$$

$$I_{CG}\dot{\nu}_2 + \nu_2 \times (I_{CG}\nu_2) = \tau_2 \tag{4.11}$$

where $\nu_1 := [\ u\ v\ w\ ]^T$, $\nu_2 := [\ p\ q\ r\ ]^T$, $\tau_1 := [\ X\ Y\ Z\ ]^T$ and $\tau_2 := [\ L\ M\ N\ ]^T$.

The system can be written on compact form as

$$M_{RB}\dot{\nu} + C_{RB}(\nu)\nu = \tau_{RB} \tag{4.12}$$

where

$$M_{RB} = \begin{bmatrix} mI_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I_{CG} \end{bmatrix} \quad C_{RB}(\nu) = \begin{bmatrix} mS(\nu_2) & 0_{3\times3} \\ 0_{3\times3} & -S(I_{CG}\nu_2) \end{bmatrix} \tag{4.13}$$

$S(\cdot)$ denotes the skew symmetric[1] operation. The inertia tensor I_{CG} is defines as

$$I_{CG} := \begin{bmatrix} I_{xx} & 0 & -I_{xz} \\ 0 & I_{yy} & 0 \\ -I_{xz} & 0 & I_{zz} \end{bmatrix} \tag{4.14}$$

where I_{ii} are moments of inertia and $I_{ij},\ i \neq j$ are products of inertia. The elements I_{xy} and I_{yz} are zero because of the missile's symmetry.

τ_{RB} is the vector of forces and moments acting on the missile body. This vector can be split into two parts, one representing the forces of gravity, and the other those from aerodynamics and control.

$$\tau_{RB} = -g(\eta) + \tau \tag{4.15}$$

Taking a closer look at the gravity part $-g(\eta)$, it can be expressed as

$$g(\eta) = -(R_B^{NED})^T \begin{bmatrix} f_G \\ 0_{3\times1} \end{bmatrix} = \begin{bmatrix} mg\sin(\theta) \\ -mg\cos(\theta)\sin(\phi) \\ -mg\cos(\theta)\sin(\phi) \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.16}$$

where $f_G = [\ 0\ 0\ mg\ ]^T$. Inserting (4.15) into (4.12) gives

$$M_{RB}\dot{\nu} + C_{RB}(\nu)\nu + g(\eta) = \tau \tag{4.17}$$

which is the desired rigid body missile model formulation. Equation (4.17) can also be expressed in component form:

$$m(\dot{u} + qw - rv + g\sin(\theta)) = X \tag{4.18}$$
$$m(\dot{v} + ur - wp - g\cos(\theta)\sin(\phi)) = Y \tag{4.19}$$
$$m(\dot{w} + vp - qu - g\cos(\theta)\cos(\phi)) = Z \tag{4.20}$$
$$I_{xx}\dot{p} - I_{xz}(\dot{r} + pq) + (I_{zz} - I_{yy}qr) = L \tag{4.21}$$
$$I_{yy}\dot{q} + I_{xz}(p^2 - r^2) + (I_{xx} - I_{zz})pr = M \tag{4.22}$$
$$I_{zz}\dot{r} - I_{xz}\dot{p} + (I_{yy} - I_{xx})pq + I_{xz}qr = N \tag{4.23}$$

4.4 Translational Dynamics

The missile translational dynamics can be derived by applying a force balance in the wind coordinate system. The differential equations for missile wind speed V, angle of attack α and sideslip angle β are derived in this section, and will ultimately be

[1] A skew-symmetric matrix is a square matrix whose transpose is also its negative.

used instead of the differential equations for u, v and w in the complete model.

The forces can be written as

$$\sum F^W = m\left(\begin{bmatrix}\dot{V}\\0\\0\end{bmatrix} + \begin{bmatrix}p_W\\q_W\\r_W\end{bmatrix} \times \begin{bmatrix}V\\0\\0\end{bmatrix}\right) = m\begin{bmatrix}\dot{V}\\r_W V\\-q_W V\end{bmatrix} \tag{4.24}$$

Force balance gives

$$F^W_{x,engine} - \mathcal{D} + mg^W_x = m\dot{V} \tag{4.25}$$
$$F^W_{y,engine} - \mathcal{C} + mg^W_y = mr_W V \tag{4.26}$$
$$F^W_{z,engine} - \mathcal{L} + mg^W_z = -mq_W V \tag{4.27}$$

where

$$\begin{bmatrix}g^W_x\\g^W_y\\g^W_z\end{bmatrix} = \mathbf{R}^W_B(\mathbf{R}^{NED}_B)^T\begin{bmatrix}0\\0\\g\end{bmatrix} \tag{4.28}$$

$$\begin{bmatrix}F^W_{x,engine}\\F^W_{y,engine}\\F^W_{z,engine}\end{bmatrix} = \mathbf{R}^W_B\begin{bmatrix}F^B_{x,engine}\\0\\0\end{bmatrix} \tag{4.29}$$

By using Equation (4.25) the differential equation for V can be found:

$$\dot{V} = \frac{1}{m}\left(F^W_{x,engine} - \mathcal{D} + mg^W_X\right) \tag{4.30}$$

To find differential equations for α and β, the relative angular rates between body and wind axes is studied

$$\mathbf{R}^W_B\begin{bmatrix}p\\q\\r\end{bmatrix} - \begin{bmatrix}p_W\\q_W\\r_W\end{bmatrix} = \mathbf{R}^W_B\begin{bmatrix}0\\\dot{\alpha}\\0\end{bmatrix} - \begin{bmatrix}0\\0\\\dot{\beta}\end{bmatrix} \tag{4.31}$$

By combining (4.31) with (4.26) and (4.27), the following expressions can be found

$$\begin{aligned}\dot{\alpha} &= -p\cos(\alpha)\tan(\beta) + q - r\sin(\alpha)\tan(\beta)\\ &\quad + \frac{1}{mV\cos(\beta)}\left(F^W_{z,engine} - \mathcal{L} + mg^W_z\right)\end{aligned} \tag{4.32}$$

$$\dot{\beta} = p\sin(\alpha) - r\cos(\alpha) + \frac{1}{mV}\left(F^W_{y,engine} - \mathcal{C} + mg^W_y\right) \tag{4.33}$$

In addition to (4.30),(4.32) and (4.33), the equation set (4.18) - (4.23) can be solved with respect to $\dot{p}$, $\dot{q}$ and $\dot{r}$:

$$\dot{p} = \frac{1}{I_{xx}I_{zz}-I_{xz}^2}\Big(I_{zz}L + I_{xz}N + (I_{yy}I_{zz} - I_{zz}^2 - I_{xz}^2)qr + I_{xz}(I_{xx} - I_{yy} + I_{zz})pq\Big) \quad (4.34)$$

$$\dot{q} = \frac{1}{I_{yy}}\Big(M + I_{xz}(r^2 - p^2) + (I_{zz} - I_{xx})pr\Big) \quad (4.35)$$

$$\dot{r} = \frac{1}{I_{xx}I_{zz}-I_{xz}^2}\Big(I_{xz}L + I_{xx}N + (I_{xz}^2 - I_{xx}I_{yy} + I_{xx}^2)pq + I_{xz}(I_{yy} - I_{zz} - I_{xx})qr\Big) \quad (4.36)$$

Equations (4.30), (4.32) - (4.36) form a system $\dot{x} = f(x, F_{engine}, F_{aero})$ fully describing the missile's movements (F_{engine} represent engine forces and F_{aero} represent aerodynamic forces).

4.5 Aerodynamic Forces and Moments

This section will look further into the aerodynamic moments L, M, N and forces $\mathcal{C}$, $\mathcal{D}$ and $\mathcal{L}$. Each of these depend on the dynamic pressure:

$$Q = \frac{1}{2}\rho V^2 \quad (4.37)$$

The moments about the missile axes can be approximated by

$$L = Qsl\Big(C_{L\beta}\beta + \frac{l}{V}C_{Lp}p + \frac{l}{V}C_{Lr}r + \sum_{i=1}^{n}(C_{L\delta_i}\delta_i)\Big) \quad (4.38)$$

$$M = Qsl\Big(C_{M0} + C_{M\alpha}\alpha + \frac{l}{V}C_{M\dot{\alpha}}\dot{\alpha} + \frac{l}{V}C_{Mq}q + \sum_{i=1}^{n}(C_{M\delta_i}\delta_i)\Big) \quad (4.39)$$

$$N = Qsl\Big(C_{N\beta}\beta + \frac{l}{V}C_{Np}p + \frac{l}{V}C_{Nr}r + \sum_{i=1}^{n}(C_{N\delta_i}\delta_i)\Big) \quad (4.40)$$

The forces working on the missile body is given by the equations

$$\mathcal{D} = Qs\Big(C_{\mathcal{D}0} + C_{D\alpha}\alpha^2\Big) \quad (4.41)$$

$$\mathcal{C} = Qs\Big(C_{\mathcal{C}\beta}\beta + \frac{l}{V}C_{\mathcal{C}\dot{\beta}}\dot{\beta} + \frac{l}{V}C_{\mathcal{C}r}r + \sum_{i=1}^{n}(C_{\mathcal{C}\delta_i}\delta_i)\Big) \quad (4.42)$$

$$\mathcal{L} = Qs\Big(C_{\mathcal{L}\alpha}\alpha + \frac{l}{V}C_{\mathcal{L}\dot{\alpha}}\dot{\alpha} + \frac{l}{V}C_{\mathcal{L}q}q + \sum_{i=1}^{n}(C_{\mathcal{L}\delta_i}\delta_i)\Big) \quad (4.43)$$

The sum terms $\sum_{i=1}^{n}(C_{j\delta_i}\delta_i)$, $j \in \{L, M, N, \mathcal{C}, \mathcal{L}\}$ represent the forces and moments generated by the missile actuators/effectors. The control vectors can be combined to form their axis specific forms δ_R, δ_P and δ_Y like described in Section 2.5. This is done to reduce the number of coefficients, essentially making the model less complex. In addition to the axis specific forms of δ, the corresponding sum of coefficients can be defined, e.g. the lift coeffients for a missile with four actuators in the x configuration,

$$C_{L\delta_R} := C_{L\delta_1} - C_{L\delta_2} + C_{L\delta_3} - C_{L\delta_4} \tag{4.44}$$
$$C_{L\delta_P} := C_{L\delta_1} + C_{L\delta_2} + C_{L\delta_3} + C_{L\delta_4} \tag{4.45}$$
$$C_{L\delta_Y} := -C_{L\delta_1} + C_{L\delta_2} + C_{L\delta_3} - C_{L\delta_4} \tag{4.46}$$

Similar expressions can be found for the other drag, lift and moment coefficients. This means that the last sum terms in Equations (4.38) - (4.43) can be replaced with general terms containing δ_R, δ_P and δ_Y and their corresponding coefficients. The specifics of these is given by the control effector setup.

4.6 Complete Nonlinear Model

In the rest of the derivation, some simplifications will be made to the model equations:

- The terms mg_z^W and mg_y^W in (4.32) and (4.33) have no significance on the missile dynamics and will be removed.
- C_{M0} in (4.38) is assumed to be neglible.
- All terms containing $\dot{\alpha}$ and $\dot{\beta}$ in (4.39), (4.42) and (4.43) are assumed to be very small and are neglected.
- For the lateral drag force $\mathcal{C}$ in (4.42), the effects of the yaw angular velocity r are small and neglected.
- For the lift force $\mathcal{L}$ in (4.43), the effects of the pitch angular velocity q are small and neglected.

Using these simplifications together with (2.5) - (2.7), the system (4.30), (4.32) - (4.36) can be rewritten. The following equations are on the form $\dot{x} = f(x, u)$, $x = [\ V\ \alpha\ \beta\ p\ q\ r\ ]^T$, $u = [\ \delta_R\ \delta_P\ \delta_Y\ ]^T$, and describe the complete nonlinear model.

$$
\begin{aligned}
\dot{V} &= \frac{1}{m}\left(F^W_{x,engine} - Qs(C_{D0} + C_{D\alpha}\alpha^2) + mg^W_x\right) && (4.47)\\
\dot{\alpha} &= -p\cos(\alpha)\tan(\beta) + q - r\sin(\alpha)\tan(\beta) + \frac{1}{mV\cos(\beta)}\Big(F^W_{z,engine} \\
&\quad -Qs(C_{\mathcal{L}\alpha}\alpha + C_{\mathcal{L}\delta_R}\delta_R + C_{\mathcal{L}\delta_P}\delta_P + C_{\mathcal{L}\delta_Y}\delta_Y)\Big) && (4.48)\\
\dot{\beta} &= p\sin(\alpha) - r\cos(\alpha) + \frac{1}{mV}\Big(F_{y,engine} \\
&\quad -Qs(C_{\mathcal{C}\beta}\beta + C_{\mathcal{C}\delta_R}\delta_R + C_{\mathcal{C}\delta_P}\delta_P + C_{\mathcal{C}\delta_Y}\delta_Y)\Big) && (4.49)\\
\dot{p} &= \frac{I_{zz}Qsl}{I_{xx}I_{zz} - I^2_{xz}}\Big(C_{L\beta}\beta + \frac{l}{V}C_{Lp}p + \frac{l}{V}C_{Lr}r + C_{L\delta_R}\delta_R + C_{L\delta_P}\delta_P + C_{L\delta_Y}\delta_Y\Big) \\
&\quad + \frac{I_{xz}Qsl}{I_{xx}I_{zz} - I^2_{xz}}\Big(C_{N\beta}\beta + \frac{l}{V}C_{Np}p + \frac{l}{V}C_{Nr}r + C_{N\delta_R}\delta_R + C_{N\delta_P}\delta_P + C_{N\delta_Y}\delta_Y\Big) \\
&\quad + \frac{1}{I_{xx}I_{zz} - I^2_{xz}}\Big((I_{yy}I_{zz} - I^2_{zz} - I^2_{xz})qr + I_{xz}(I_{xx} - I_{yy} + I_{zz})qp\Big) && (4.50)\\
\dot{q} &= \frac{Qsl}{I_{yy}}(C_{M\alpha}\alpha + \frac{l}{V}C_{Mq}q + C_{M\delta_R}\delta_R + C_{M\delta_P}\delta_P + C_{M\delta_Y}\delta_Y) \\
&\quad + \frac{I_{xz}}{I_{yy}}(r^2 - p^2) + \frac{1}{I_{yy}}(I_{zz} - I_{xx})pr && (4.51)\\
\dot{r} &= \frac{I_{xz}Qsl}{I_{xx}I_{zz} - I^2_{xz}}\Big(C_{L\beta}\beta + \frac{l}{V}C_{Lp}p + \frac{l}{V}C_{Lr}r + C_{L\delta_R}\delta_R + C_{L\delta_P}\delta_P + C_{L\delta_Y}\delta_Y\Big) \\
&\quad + \frac{I_{xx}Qsl}{I_{xx}I_{zz} - I^2_{xz}}\Big(C_{N\beta}\beta + \frac{l}{V}C_{Np}p + \frac{l}{V}C_{Nr}r + C_{N\delta_R}\delta_R + C_{N\delta_P}\delta_P + C_{N\delta_Y}\delta_Y\Big) \\
&\quad + \frac{1}{I_{xx}I_{zz} - I^2_{xz}}\Big((I_{xz} - I_{xx}I_{yy} + I^2_{xx})pq + I_{xz}(I_{yy} - I_{zz} - I_{xx})qr\Big) && (4.52)
\end{aligned}
$$

4.7 Simplified Model

For an aircraft, it is common to assume that the longitudal modes can be decoupled from the lateral modes. The reason for this is that the aircraft fuselage has a length much larger than its width and height. Also, the longitudal velocity is much larger than the vertical and transverse velocities.

4.7.1 Longitudal Model

In the longitudal model, we ignore the effect of the lateral modes, and set the states $\beta = r = p = \phi = 0$. Also, we neglect the roll and yaw inputs, i.e. $\delta_R = \delta_Y = 0$.

The equations for the longitudal model then become

$$\begin{aligned}
\dot{\alpha} &= q - \frac{Qs}{mV}(C_{\mathcal{L}\alpha}\alpha + C_{\mathcal{L}\delta_P}\delta_P) && (4.53)\\
\dot{q} &= \frac{Qsl}{I_{yy}}(C_{M\alpha}\alpha + \frac{l}{V}C_{Mq}q + C_{M\delta_P}\delta_P) && (4.54)\\
\dot{\theta} &= q && (4.55)\\
a_z^W &= -\frac{Qs}{m}(C_{\mathcal{L}\alpha}\alpha + C_{\mathcal{L}\delta_P}\delta_P) && (4.56)
\end{aligned}$$

This can be posed as a linear state space system $\dot{x}_{long} = A_{long}x_{long} + B_{long}u_{long}$,

$$\begin{bmatrix}\dot{\alpha}\\ \dot{q}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\frac{-Qs}{mV}C_{\mathcal{L}\alpha} & 1 & 0\\ \frac{Qsl}{I_{yy}}C_{M\alpha} & \frac{Qsl^2}{I_{yy}V}C_{Mq} & 0\\ 0 & 1 & 0\end{bmatrix}\begin{bmatrix}\alpha\\ q\\ \theta\end{bmatrix} + \begin{bmatrix}\frac{-Qs}{mV}C_{\mathcal{L}\delta_P}\\ \frac{Qsl}{I_{yy}}C_{M\delta_P}\\ 0\end{bmatrix}\delta_P \qquad (4.57)$$

with measurement $y_{long} = C_{long}x_{long} + D_{long}u_{long}$,

$$\begin{bmatrix}a_z^W\\ q\\ \theta\end{bmatrix} = \begin{bmatrix}\frac{-Qs}{m}C_{\mathcal{L}\alpha} & 1 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\alpha\\ q\\ \theta\end{bmatrix} + \begin{bmatrix}\frac{-Qs}{m}C_{\mathcal{L}\delta_P}\\ 0\\ 0\end{bmatrix}\delta_P \qquad (4.58)$$

4.7.2 Lateral Model

In the lateral model, we ignore the effect of the longitudal modes, and set the states $\alpha = q = \theta = 0$, and the pitch input $\delta_P = 0$. In addition to this, we assume $\cos(\phi) \approx 1$.

The equations for the lateral model are

$$\begin{aligned}
\dot{\beta} &= -r - \frac{Qs}{mV}(C_{\mathcal{C}\beta}\beta + C_{\mathcal{C}\delta_R}\delta_R + C_{\mathcal{C}\delta_Y}\delta_Y) && (4.59)\\
\dot{p} &= \frac{I_{zz}Qsl}{I_{xx}I_{zz} - I_{xz}^2}(C_{L\beta}\beta + \frac{l}{V}C_{Lp}p + \frac{l}{V}C_{Lr}r + C_{L\delta_R}\delta_R + C_{L\delta_Y}\delta_Y)\\
&+ \frac{I_{xz}Qsl}{I_{xx}I_{zz} - I_{xz}^2}(C_{N\beta}\beta + \frac{l}{V}C_{Np}p + \frac{l}{V}C_{Nr}r + C_{N\delta_R}\delta_R + C_{N\delta_Y}\delta_Y) && (4.60)\\
\dot{r} &= \frac{I_{xz}Qsl}{I_{xx}I_{zz} - I_{xz}^2}(C_{L\beta}\beta + \frac{l}{V}C_{Lp}p + \frac{l}{V}C_{Lr}r + C_{L\delta_R}\delta_R + C_{L\delta_Y}\delta_Y)\\
&+ \frac{I_{xx}Qsl}{I_{xx}I_{zz} - I_{xz}^2}(C_{N\beta}\beta + \frac{l}{V}C_{Np}p + \frac{l}{V}C_{Nr}r + C_{N\delta_R}\delta_R + C_{N\delta_Y}\delta_Y) && (4.61)\\
\dot{\phi} &= p && (4.62)\\
\dot{r} &= r && (4.63)\\
a_y^W &= -\frac{Qs}{m}(C_{\mathcal{C}\beta}\beta + C_{\mathcal{C}\delta_R}\delta_R + C_{\mathcal{C}\delta_Y}\delta_Y) && (4.64)
\end{aligned}$$

This can be posed as a linear state space system $\dot{x}_{lat} = A_{lat}x_{lat} + B_{lat}u_{lat}$, where

$$x_{lat} = [\ \beta\ p\ r\ \phi\ \psi\ ]^T \tag{4.65}$$

$$u_{lat} = [\ \delta_R\ \delta_Y\ ]^T \tag{4.66}$$

$$A_{lat} = \begin{bmatrix} \frac{Qs}{mV}C_{\mathcal{C}\beta} & 0 & -1 & 0 & 0 \\ \kappa(I_{zz}C_{L\beta} + I_{xz}C_{N\beta}) & \kappa\frac{l}{V}(I_{zz}C_{Lp} + I_{xz}C_{Np}) & \kappa\frac{l}{V}(I_{zz}C_{Lr} + I_{xz}C_{Nr}) & 0 & 0 \\ \kappa(I_{xz}C_{L\beta} + I_{xx}C_{N\beta}) & \kappa\frac{l}{V}(I_{xz}C_{Lp} + I_{xx}C_{Np}) & \kappa\frac{l}{V}(I_{xz}C_{Lr} + I_{xx}C_{Nr}) & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{4.67}$$

$$B_{lat} = \begin{bmatrix} \frac{-Qs}{mV}C_{\mathcal{C}\delta_R} & \frac{-Qs}{mV}C_{\mathcal{C}\delta_Y} \\ \kappa(I_{zz}C_{L\delta_R} + I_{xz}C_{N\delta_R}) & \kappa(I_{zz}C_{L\delta_Y} + I_{xz}C_{N\delta_Y}) \\ \kappa(I_{xz}C_{L\delta_R} + I_{xx}C_{N\delta_R}) & \kappa(I_{xz}C_{L\delta_Y} + I_{xx}C_{N\delta_Y}) \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{4.68}$$

where $\kappa = \frac{Qsl}{I_{xx}I_{zz}-I_{xz}^2}$. The measurement $y = [a_y^W\ p\ r\ \phi\ \psi\ ]^T = C_{long}x_{long}+D_{long}u_{long}$ has system matrices

$$C_{lat} = diag\Big(\ [\tfrac{-Qs}{m}C_{\mathcal{C}\beta} \quad 1 \quad 1 \quad 1 \quad 1]\ \Big) \tag{4.69}$$

$$D_{lat} = \begin{bmatrix} \frac{-Qs}{m}C_{\mathcal{C}\delta_R} & \frac{-Qs}{m}C_{\mathcal{C}\delta_R} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{4.70}$$

4.8 Actuator Dynamics

The actuators are modeled using a second order approximation,

$$\ddot{\delta} - 2\zeta\omega_0\dot{\delta} - \omega_0^2\delta = \omega_0^2\delta_{cmd} \tag{4.71}$$

where ω_0 is the actuator's natural frequency and ζ is its damping ratio. δ_{cmd} is the commanded actuator input, while δ is the actual actuator response. Equation (4.71) can be rewritten as a state-space system

$$d = A_\delta d + B_\delta d_{cmd} \tag{4.72}$$

where

$$d = [\ d_1\ d_2\ ]^T, \quad d_1 = \delta, \quad d_2 = \dot{\delta} \tag{4.73}$$

$$A_\delta = \begin{bmatrix} 0 & 1 \\ -\omega_0^2 & -2\zeta\omega_0 \end{bmatrix}, \quad B_\delta = \begin{bmatrix} 0 \\ \omega_0^2 \end{bmatrix} \tag{4.74}$$

It should be noted that actuator position d_1 is most relevant when such models are used in a control allocation scope. Constraints on actuator velocity d_2 can also be present, but are ignored in this thesis.

Two examples of step responses are shown for a typical fast and slow actuators.

Fast actuator, $\zeta = 0.7$, $\omega_0 = 150$

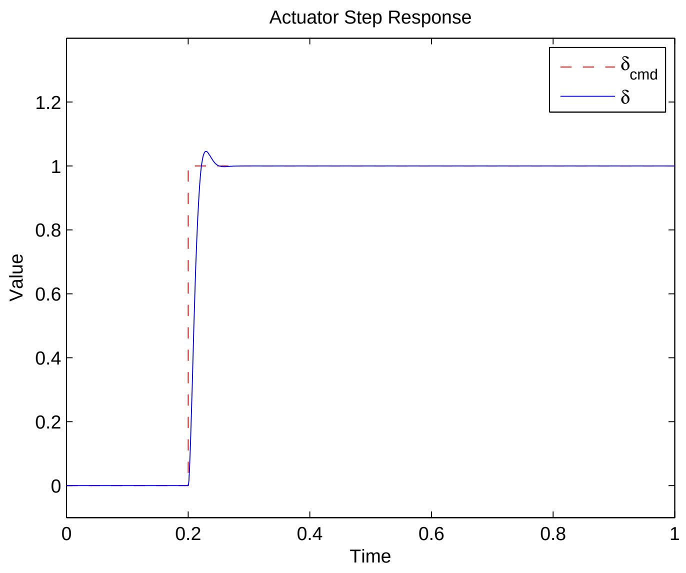

Figure 4.2: Step response of a fast actuator

Slow actuator, $\zeta = 0.95$, $\omega_0 = 10$

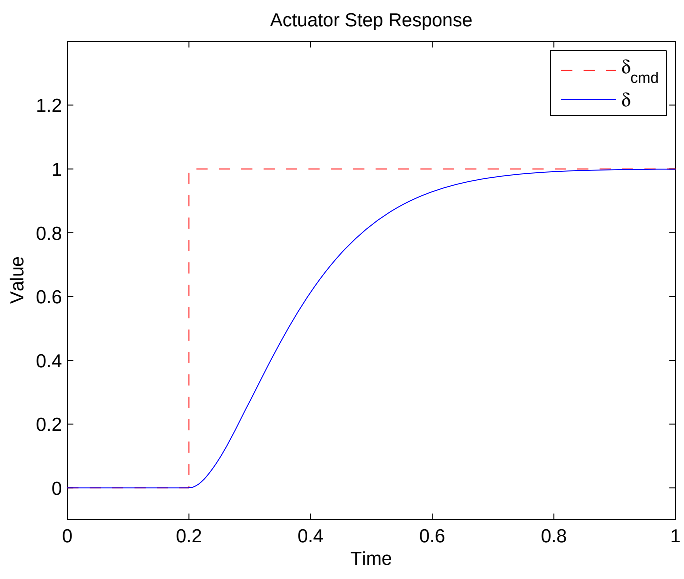

Figure 4.3: Step response of a slow actuator

Chapter 5

Test Setup

In this chapter, the control allocation test setup will be presented. This is to clarify how the tests this thesis' results are based on were conducted, and to make the tests repeatable. First, the way the setup is implemented is explained in a module-wise manner. Second, the control allocation methods utilized are discussed, followed by a brief explanation of the various virtual control prediction methods used. Lastly, the test plan is presented.

5.1 Implementation

In this section, the setup in which control allocation methods are tested is presented in detail. The setup consists of several modules working together to simulate a flight control system working on a missile body:

- Autopilot
- Prediction Module
- Control Allocation Module
- Actuator Dynamics
- Missile Airframe Dynamics

The setup is created using the MATLAB and SimuLink framework. Figure 5.1 presents an overview of the setup. The inner workings of the different modules are presented in the sections that follow.

5.1.1 Autopilot

The missile autopilot is part of the flight control system, displayed in Figure 2.2. The autopilot block consists of two loops. The outer position loop regulates the longitudal y position and the lateral z position to their reference values y_{ref} and z_{ref}, which are given by the guidance law. Inside the position loop lies the angle loop, regulating the missile roll, pitch and yaw angles. See Figure 5.2 for a detailed view of the autopilot block. The output of the autopilot forms the virtual control vector

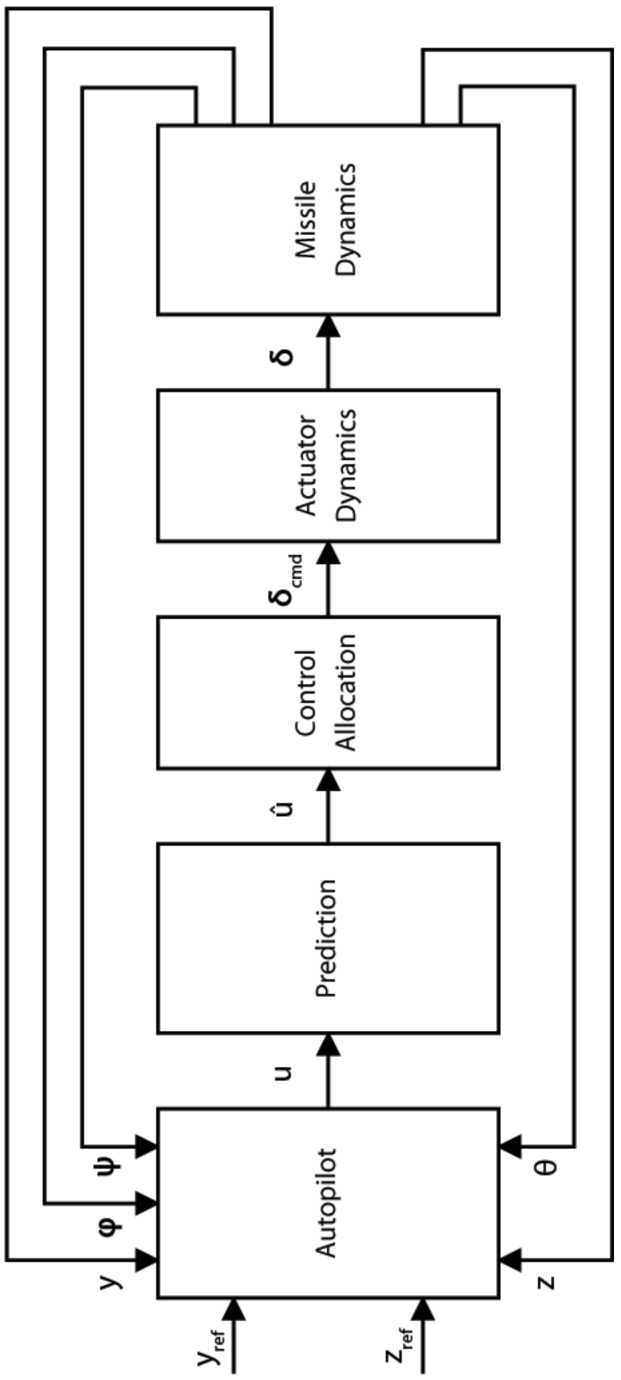

Figure 5.1: Test setup overview.

$u = [\ \delta_R^* \ \delta_P^* \ \delta_Y^* \ ]^T$, which is fed to the control allocation module.

Measurements of z and y are obtained from the acceleration equations (4.56) and (4.64),

$$z(t) = \int\int a_z^W(t)\ dt\ dt \qquad (5.1)$$

$$y(t) = \int\int a_y^W(t)\ dt\ dt \qquad (5.2)$$

Measurements of roll, pitch and yaw angles θ, ϕ and ψ are also obtained from the model.

The horizontal and vertical positions are controlled by PI-regulators, while all the angles roll, pitch and yaw are controlled by P-regulators. These types of regulators are chosen for simplicity, since the focus of this thesis is testing control allocation methods. The bank-to-turn logic simply commands the missile to roll to turn, while keeping the yaw angle as small as possible.

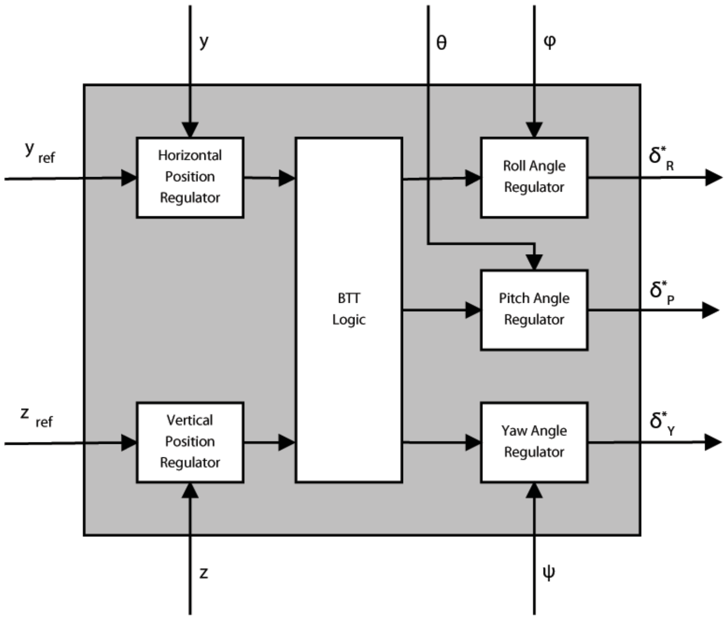

Figure 5.2: Missile autopilot block.

5.1.2 Prediction Module

The model predictive control allocation needs an estimate of the future virtual control input $\hat{u}(k+i)$, $i \in \{0 \dots T\}$, where T is the prediction horizon and k is the current time step.

An Embedded MATLAB Function block implements this prediction. More on the various ways of creating this prediction is discussed in Section 5.3. A block diagram of the prediction module is shown in Figure 5.3.

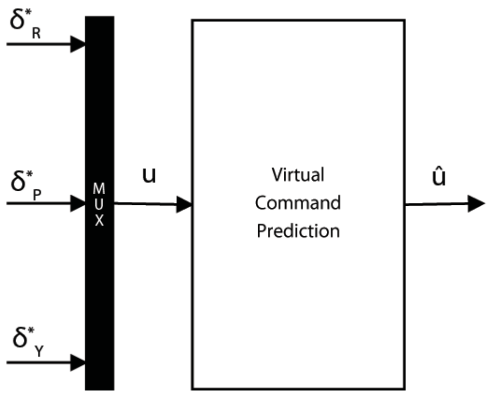

Figure 5.3: Virtual control prediction block.

5.1.3 Control Allocation Module

The control allocation module is the heart of this test setup. It is implemented using an Embedded MATLAB Function block, which executes the specific control allocation method.

The MPCA method, which is the main focus of this thesis, is implemented differently than the other methods that are considered. Because of the predictive nature of the method, it needs an estimate of the virtual control $\hat{u}$ throughout the prediction horizon. In addition, it requires knowledge of the current actuator states $\boldsymbol{\delta}$. These inputs are available from the Prediction and Actuator Dynamics blocks, and are fed to the Embedded MATLAB Function block responsible for the control allocation. This block makes an external call to the CVXGEN solver, which uses the current data to solve the MPCA problem. The MPCA setup is shown in Figure 5.4. The CVXGEN code for the MPCA problems can be found in Appendix C.

The other methods RP and LP do not require an estimate of u, just the current value. Since the first value of the predicted vector $\hat{u}$ is the current value $u(k)$, this can be used while ignoring the rest of the values in the vector. In addition the current actuator state vector $\boldsymbol{\delta}$ is not needed and therefore ignored.

Besides from the mentioned modifications, the CA methods are implemented in the same way as MPCA, with an Embedded MATLAB Function block. The LP method uses an CVXGEN-generated solver, while the RP method is written directly in the Function block. For CVXGEN code for the LP CA method, see Appendix C. The code for the RP CA is found in Appendix D.

It should also be mentioned that the control efficiency matrix B used in the control allocation methods is constant in the simulations performed in is thesis. This is done for simplicity. In a real application, this matrix would be a function of speed, air density, angle of attack and sideslip.

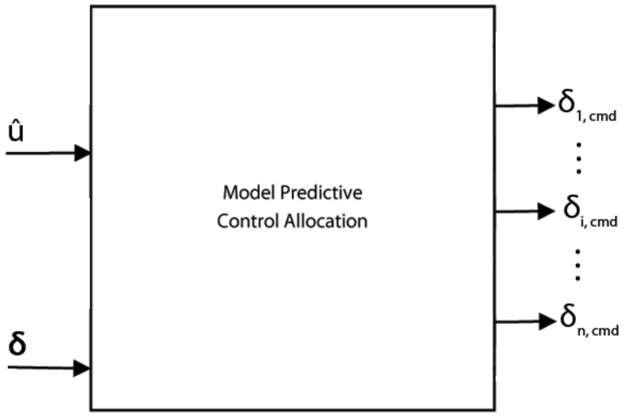

Figure 5.4: MPCA block.

5.1.4 Actuator Dynamics

The actuator dynamics are simply implemented as transfer function blocks in the Simulink diagram. The second order system (4.71) can be rewritten as a transfer function

$$H(s) = \frac{\delta}{\delta_{cmd}}(s) = \frac{\omega_0^2}{s^2 + 2\zeta\omega_0 s + \omega_0^2} \tag{5.3}$$

The individual control commands δ_i are combined to form the axis specific control commands δ_R, δ_P and δ_Y for roll, pitch and yaw in the manner described in Section 2.5. These are then fed to the missile model. See Figure 5.5 for a diagram.

5.1.5 Missile Model

The longitudal and lateral models are implemented in the Simulink diagram with regular blocks. The longitudal model uses equations (4.53)-(4.53). The lateral model uses equations (4.59)-(4.64).

It should be mentioned that such a decoupled model setup is valid only for small angles, but this is assumed to be sufficient for testing control allocation. A diagram of the model block is shown in Figure 5.6. The model data can be found in Appendix A.

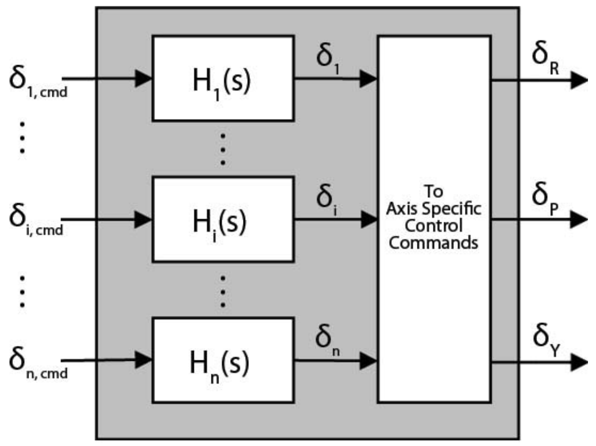

Figure 5.5: Actuator Dynamics block.

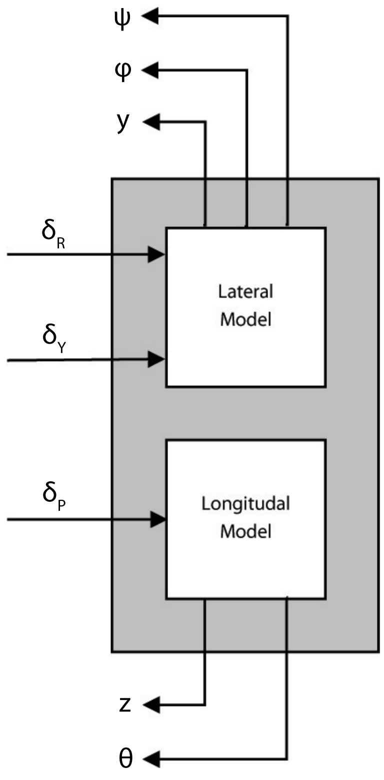

Figure 5.6: Model block.

5.2 Control Allocation Methods

The three control allocation methods considered in this thesis represent three very different types. The RP method is based on the pseudoinverse, and thus represents a simple solution. It is able to take actuator saturation limits into consideration, but ignores actuator dynamics. Still, this is a simple method which is easily implemented without the use of optimization solvers, and this simplicity it is perhaps its strongest trait.

The second method is the LP method, implemented with the mixed optimization formulation (3.15). This is an optimization based method, and is *optimal* (Section 2.3.3), which serves as a contrast to the RP method, though more cumbersome to implement. While being able to handle actuator constraints, it disregards the actuator dynamics. Still, because of the optimality property, it is expected to perform better than RP.

Being the main part of this thesis, the MPCA method is the last one to stand trial. It is, as the LP method, optimization based and optimal. Aswell as handling actuator constraints, it is the only method taking actuator dynamics into account. Because of this last property, the method is expected to perform the best in a case where actuators take on different dynamic ranges. In such a scenario the method can set itself aside from others by exploiting its knowledge of the actuator dynamics to increase the overall system's performance. Note that this method is also the most complex, but with the use of the CVXGEN solver, its implementation is simple and the execution times are expected to be very low. The MPCA is designed to use a sample time of $\bar{T} = 0.05s$. It will have an prediction horizon of $T = 5$, which means that the horizon will span $\bar{T} \cdot T = 0.2s$.

5.3 Virtual Control Prediction

As mentioned in Section 5.1.2, the MPCA module needs an estimate of the future virtual control $\hat{u}(k+i),\ i \in \{0 \ldots T\}$, where T is the prediction horizon and k is the current time step. This thesis considers three different ways of performing this prediction.

The first, denoted type 1, is simply to hold the current virtual control value through the prediction horizon, i.e. $\hat{u}(k+i) = u(k),\ \forall i$.

The next approach, type 2, is based on curve fitting. The T last samples are remembered, and a curve is fitted through the data using interpolation. A second order interpolation is sufficient and has been tested to yield good results. The second order polynomial is then used for extrapolation throughout the prediction horizon.

Last to be considered, denoted type 3, is an approach that uses knowledge of the missile reference trajectory. For the roll and yaw channel, a vector is formed by using the derivative of a low-pass filtered version of y_{ref}. For the pitch channel, a low-pass filtered and differentiated version of z_{ref} is used. These prediction vectors can be denoted $\hat{u}_{ff}(k+i),\ i \in \{\ 0 \dots T\ \}$. This vector is then merged with a vector of the current virtual control values, like the one type 1 prediction generates:

$$\hat{u}(k+i) = \gamma u(k) + (1-\gamma)\hat{u}_{ff}(i),\ i \in \{\ 0 \dots T\ \} \tag{5.4}$$

where $\gamma = \texttt{linspace}(1, \sigma, T)$ is a vector with T linear steps between 1 and σ. σ is a number between 0 and 1, typically 0.5, reflecting the importance between tracking and prediction. This merging of the vectors makes for good tracking, while also taking future dynamics into consideration.

5.4 Test plan

In this section the test plan is presented. It will consist of multiple tests, which will be used to judge the performance of the Model Predictive Control Allocation scheme. The plan will be split into five main tests.

5.4.1 Part I - Tail fins only

In this test, the missile will have tail fins only, i.e. be in the x-configuration (see Section 2.5). The control efficiency matrix is

$$B = \begin{bmatrix} 0.3 & -0.3 & 0.4 & -0.4 \\ 0.8 & 0.8 & 0.6 & 0.6 \\ -0.6 & 0.6 & 0.5 & -0.5 \end{bmatrix} \tag{5.5}$$

All actuators 1, 2, 3, 4 (Figure 2.4) will have the same characteristics and cost weight,

$$\omega_0 = 150,\ \zeta = 0.7,\ W = 1 \tag{5.6}$$

These are considered to be fast actuator dynamics. Actuator limits are $\delta_{min} = -20°$, $\delta_{max} = 20°$.

This test will investigate a scenario where MPCA perhaps is *not* a good choice. The method's performance will be compared to that of RP and LP. The prediction for the MPCA will be type 1.

5.4.2 Part II - Trimmed flight

In this test, the missile will be in the new configuration with wing ailerons. The missile will be tasked with performing a simple maneuvre, simulating trimmed flight. The performance of MPCA with type 1 prediction will be compared to that of RP and LP.

This test will investigate how the MPCA handles actuators with different dynamic ranges, and how it is able to allocate only those with low cost when necessary. The actuator characteristics and cost weight are summarized below.

$$\begin{aligned} &\omega_{0,1} = 150,\ \zeta_1 = 0.7,\ W_1 = 1 \qquad &&\omega_{0,2} = 150,\ \zeta_2 = 0.7,\ W_2 = 1 \\ &\omega_{0,3} = 150,\ \zeta_3 = 0.7,\ W_3 = 1 \qquad &&\omega_{0,4} = 150,\ \zeta_4 = 0.7,\ W_4 = 1 \\ &\omega_{0,5} = 15,\ \zeta_5 = 0.9,\ W_5 = 0.1 \qquad &&\omega_{0,6} = 15,\ \zeta_6 = 0.9,\ W_6 = 0.1 \end{aligned} \tag{5.7}$$

Actuators 1-4 are the fast tail fins, while actuators 5-6 are the slow wing ailerons (This is reflected in the size of the actuators natural frequency ω_0 - larger is faster). See Figure 2.5 for actuator placements.

The control efficiency matrix is

$$B = \begin{bmatrix} 0.3 & -0.3 & 0.4 & -0.4 & 0.8 & -0.8 \\ 0.8 & 0.8 & 0.6 & 0.6 & 0.7 & 0.7 \\ -0.6 & 0.6 & 0.5 & -0.5 & -0.3 & 0.3 \end{bmatrix} \tag{5.8}$$

Actuator limits are $\pm 20°$ for the tail fins, and $\pm 10°$ for the wing ailerons.

5.4.3 Part III - Agile Flight

In Part III of the test plan, the missile is in the new configuration where the actuatos have the same characteristics as in the previous test (5.7). The control effiency matrix B also similar (5.8). The actuator limits are $\pm 20°$ for the tail fins, and $\pm 10°$ for the wing ailerons. In addition there are constraints on the commanded control, $-10° \leq \delta_{cmd,j} \leq 10°$ for $j = \{5, 6\}$, i.e. for the wing ailerons (the reason for this will become clear in Part IV).

The difference from Part II is that the missile reference trajectory consists of steps, which will challenge the flight control system and control allocation. This scenario is thought to emulate "agile" flight, and will investigate how the control allocation module behaves when stressed.

The performance MPCA with type 1 prediction will be compared to that of RP and LP.

5.4.4 Part IV - Honoring Actuator Constraints

This part of the test will be similar to that of Part III, except for one detail. In the previous test, the MPCA problem formulation had constraints on δ_{cmd} aswell as δ. In this test these will be removed, revealing one of the weaknesses of MPCA.

The MPCA with constraints on δ_{cmd} will be compared to the one without.

5.4.5 Part V - Prediction Comparison

This test compares types of MPCA prediction on a missile in the new configuration, with actuator characteristics like (5.7) and control efficiency matrix B like (5.8). The types of prediction which are compared are type 1 and type 3. The former represents a very simple approach, while the latter is more advanced and also is expected to yield the best results. In the testing, the MPCA with prediction type 3 will be denoted by "MPCA+", while the one with type 1 will be denoted only "MPCA". The results of the two MPCA tests are compared to that of the RP method.

Chapter 6

Results

In this chapter, the results of the control allocation method testing are presented. The tests are described in Section 5.4. Some preliminary comments are given to the plots, but the main discussion follows in the next chapter.

6.1 Part I - Tail fins only

As mentioned in Section 5.4.1, this test will be on a missile with tail fins only. The missile is given lateral and longitudal step trajectories to follow. This step and the missile's response for the different control allocation methods are shown in Figure 6.1. Clearly, all three methods perform very similarly.

Looking at the actuator response in Figure 6.2, one can observe that the it is almost identical for the three control allocation methods though LP allocates the actuators somewhat differently.

Lastly, the virtual control u and the actual control $\tau = B\delta$ is compared in Figure 6.3, where the same conclusions as above can be drawn.

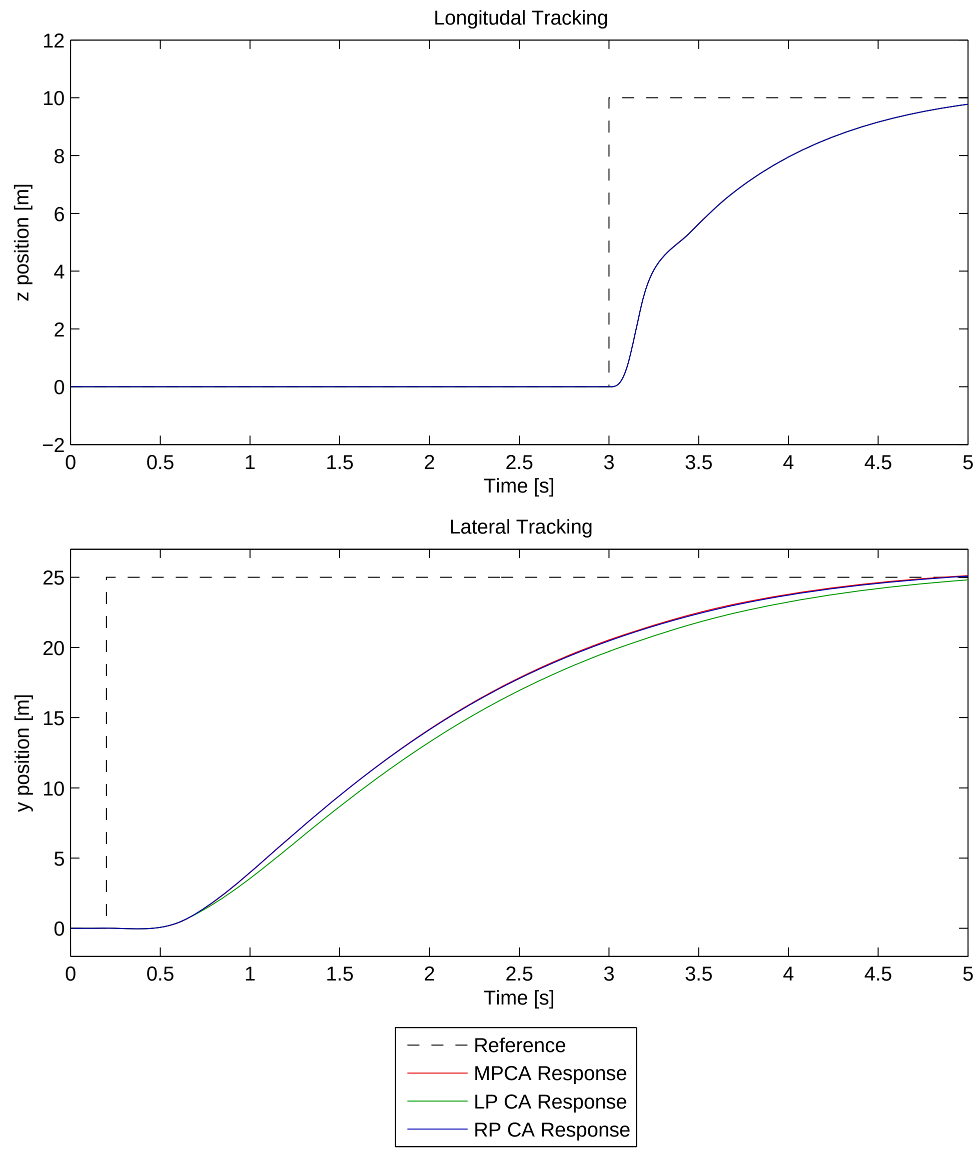

Figure 6.1: Part I - Tail fins only: Trajectory step response

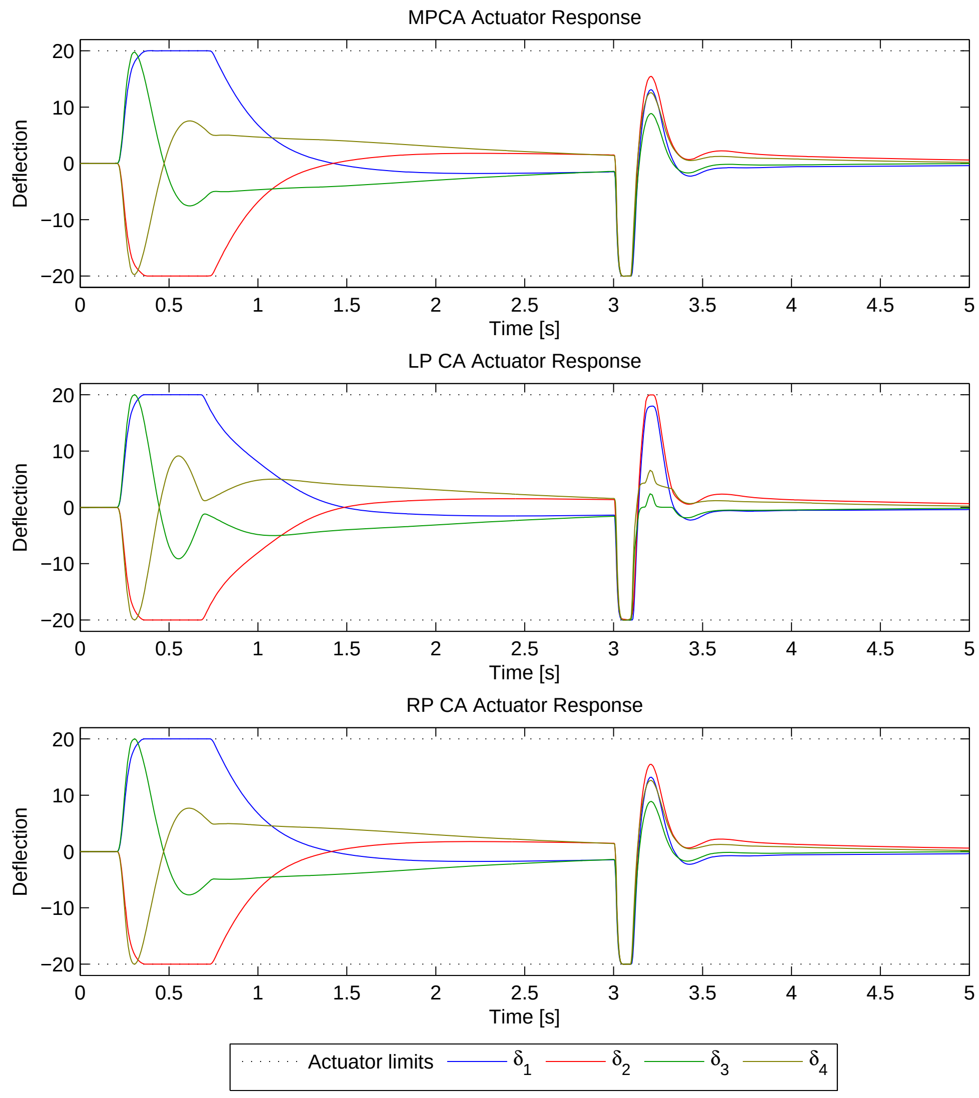

Figure 6.2: Part I - Tail fins only: Actuator response.

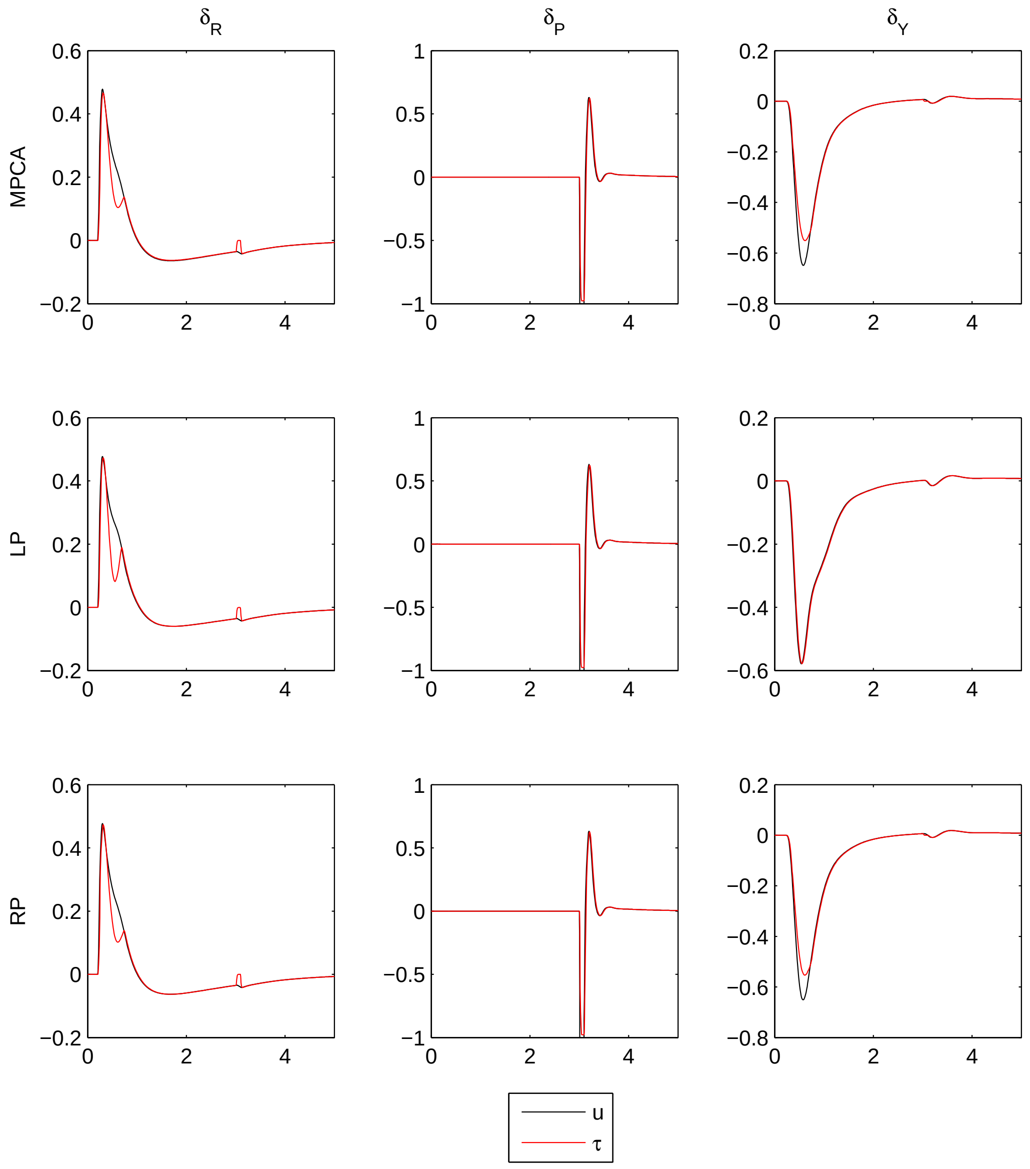

Figure 6.3: Part I - Tail fins only: Virtual control tracking.

6.2 Part II - Trimmed flight

The remaining parts of the test will be on a missile with the new configuration, tail fins plus wing ailerons. The test specifics for Part II is found in Section 5.4.2. The missile is tasked with following a longitudal trajectory.

In Figure 6.4, the reference trajectory and missile tracking performance for the longitudal and lateral cases are displayed. The three methods give roughly the same tracking.

Further, the actuator response is displayed in Figure 6.5, where the actuator limits are displayed as dashed black lines. Because of the weighting of the actuators (5.7), the wing ailerons are utilized most. Looking at the virtual control tracking in Figure 6.6, the tracking is quite similar for all three methods, but the LP method causes some oscillation at the end of the maneuvre.

Lastly, the cumulative cost can be seen in Figure 6.7.

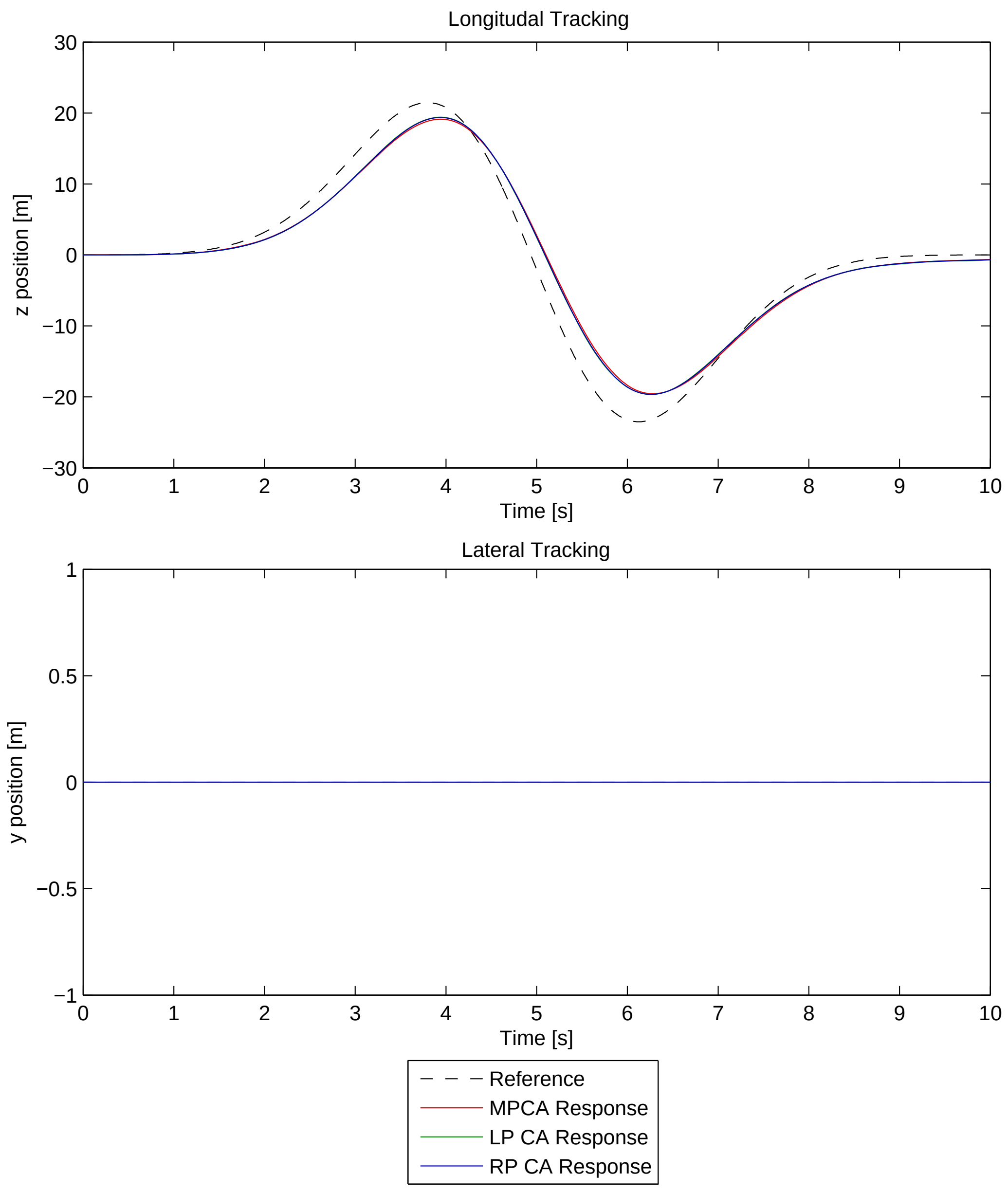

Figure 6.4: Part II - Trimmed flight: Trajectory tracking.

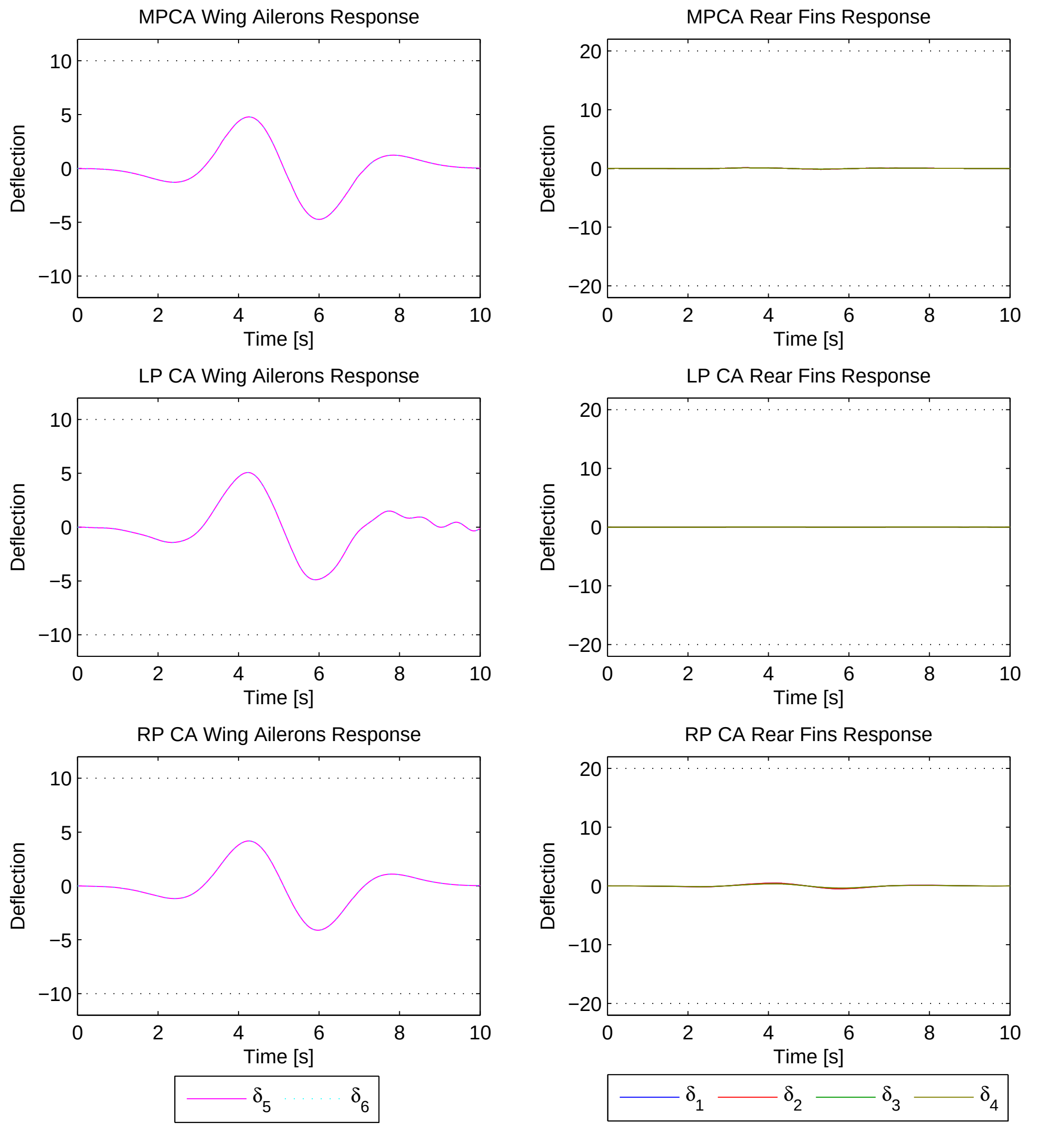

Figure 6.5: Part II - Trimmed flight: Actuator response. Left: Wing ailerons. Right: Tail fins.

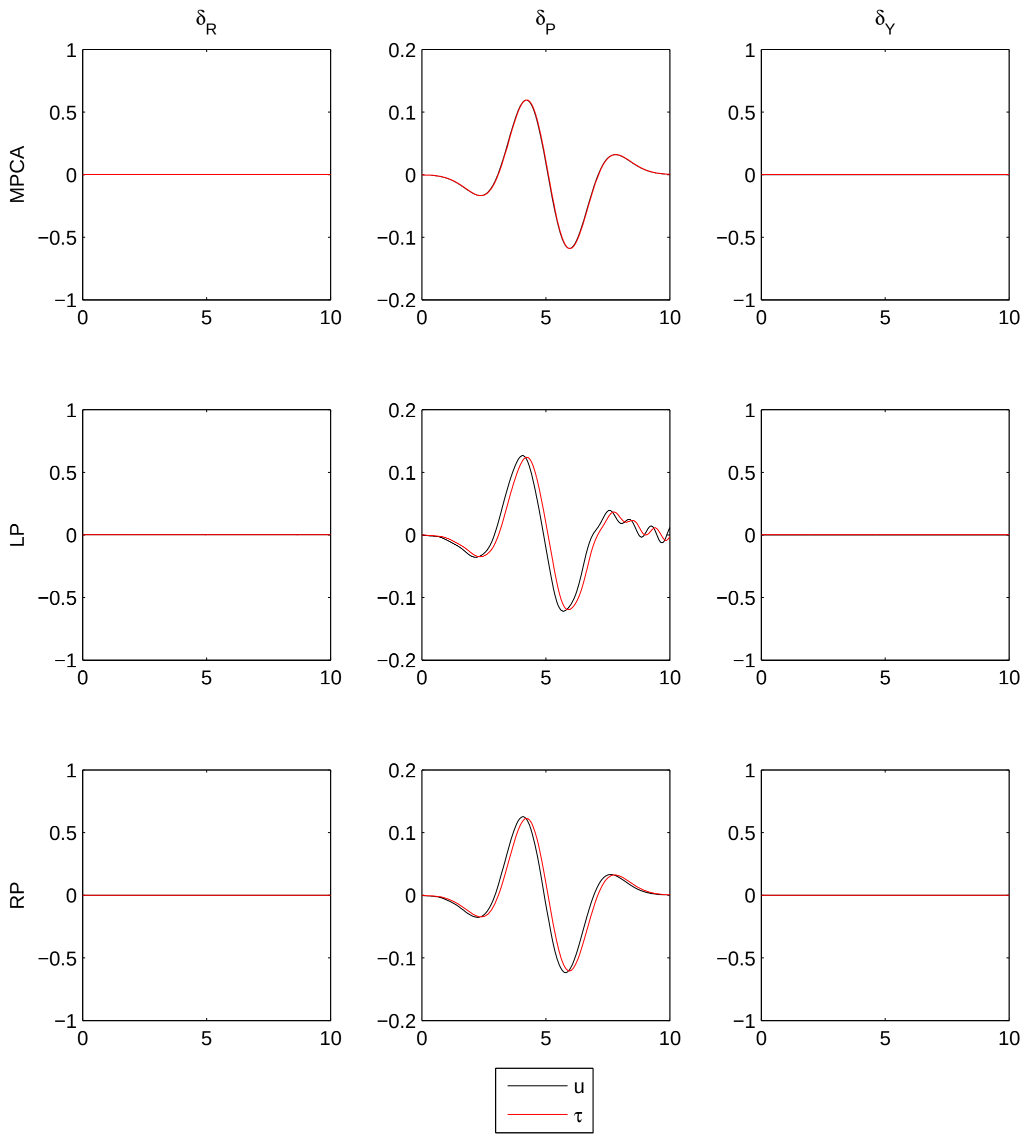

Figure 6.6: Part II - Trimmed flight: Virtual control tracking.

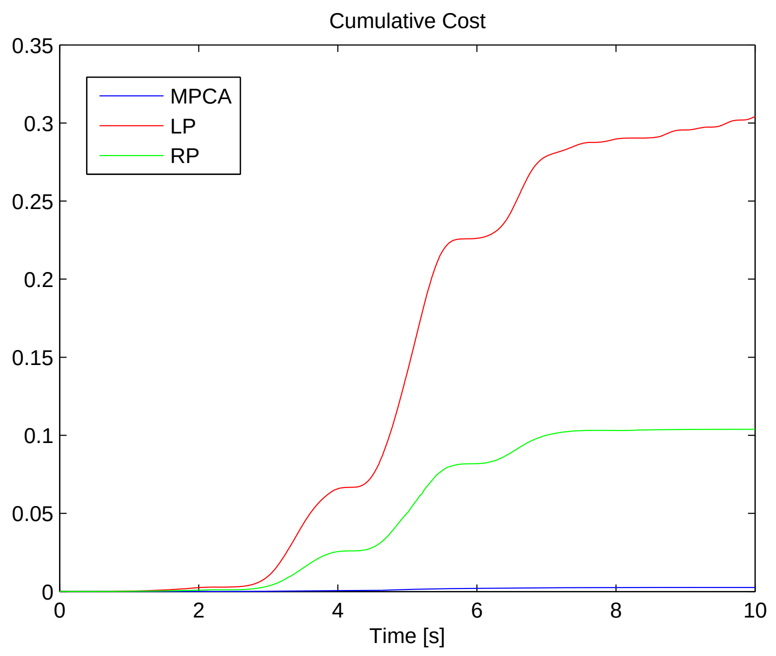

Figure 6.7: Part II - Trimmed flight: Cumulative cost.

6.3 Part III - Agile flight

The missile is tasked to follow an agile trajectory, which will increase the load on the CA module. The test specifics were presented in 5.4.3.

Figure 6.8 shows the trajectory reference and corresponding missile response. The three methods give very similar response.

Further the actuator response is shown in Figure 6.9. All actuators are held within their limits, which are shown as dashed black lines. It can be noted that the LP and RP methods have more oscillatory behaviour than the MPCA method, a behaviour which also is present in the virtual control tracking in Figure 6.10.

Finally, the cumulative costs for the three methods are shown in Figure 6.11.

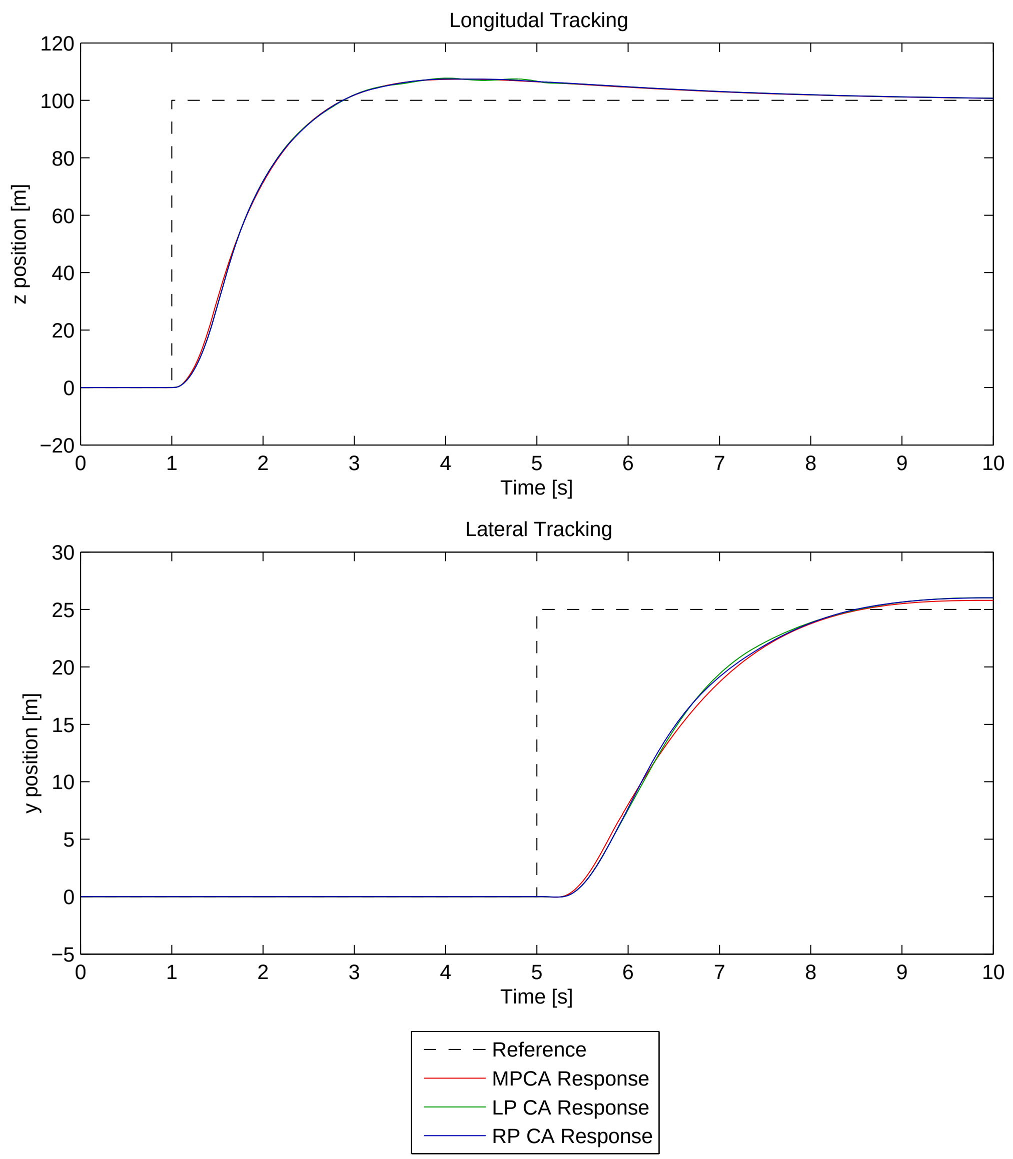

Figure 6.8: Part III - Agile flight: Trajectory tracking.

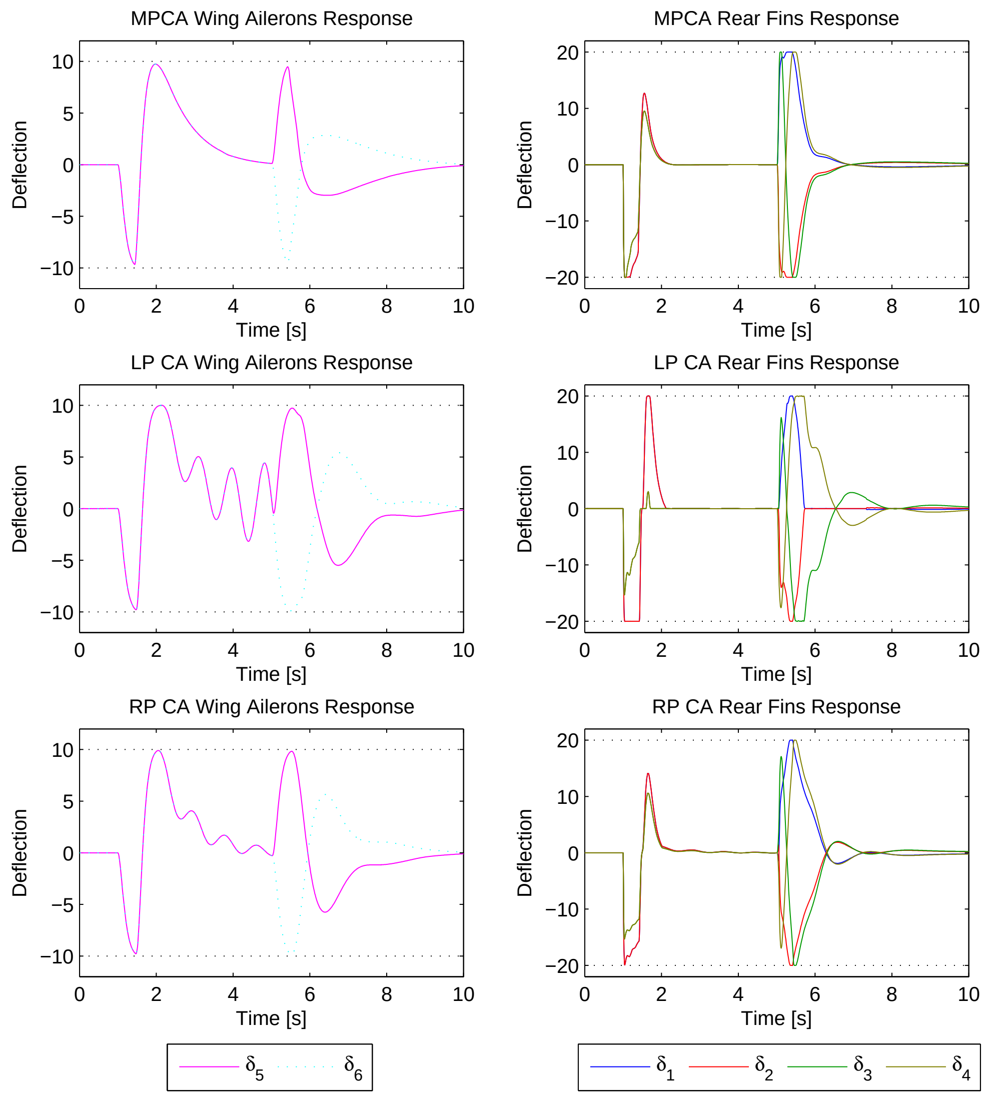

Figure 6.9: Part III - Agile flight: Actuator response. Left: Wing ailerons. Right: Tail fins.

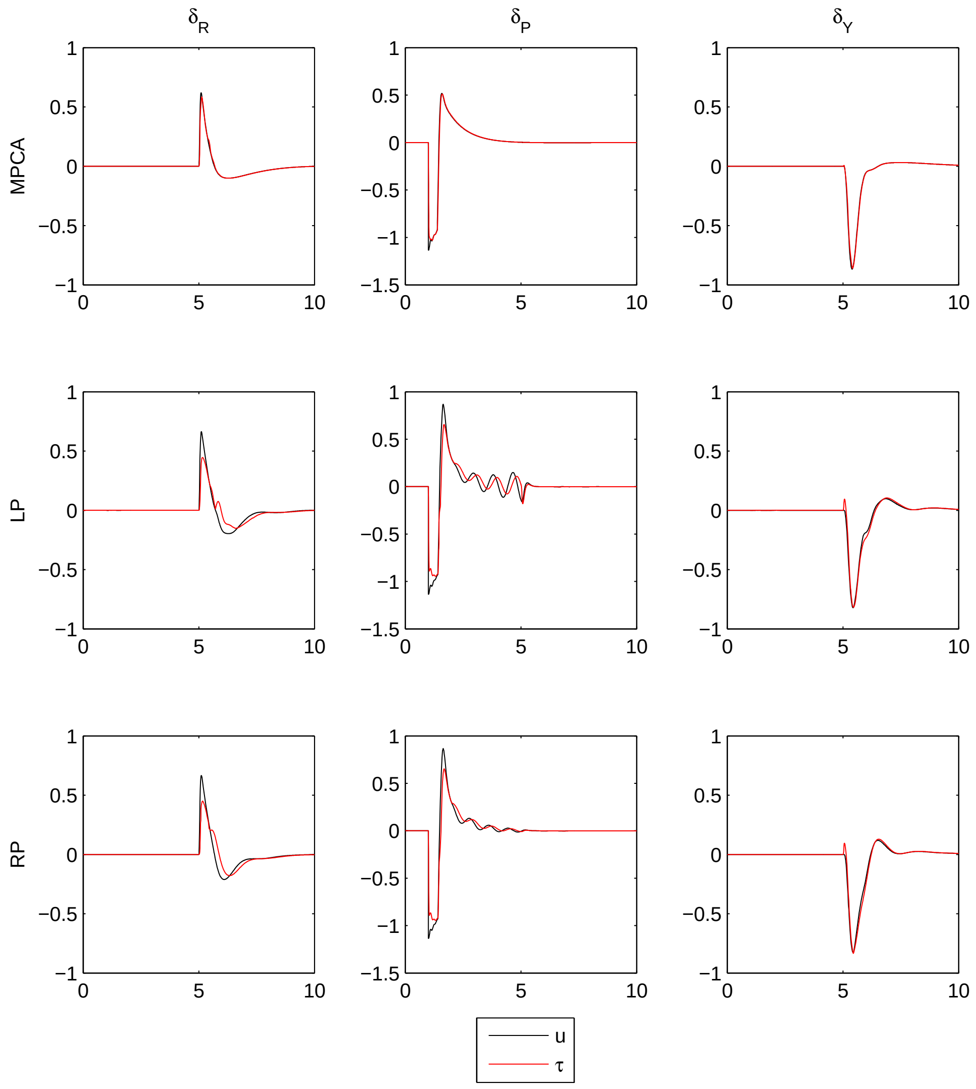

Figure 6.10: Part III - Agile flight: Virtual control tracking.

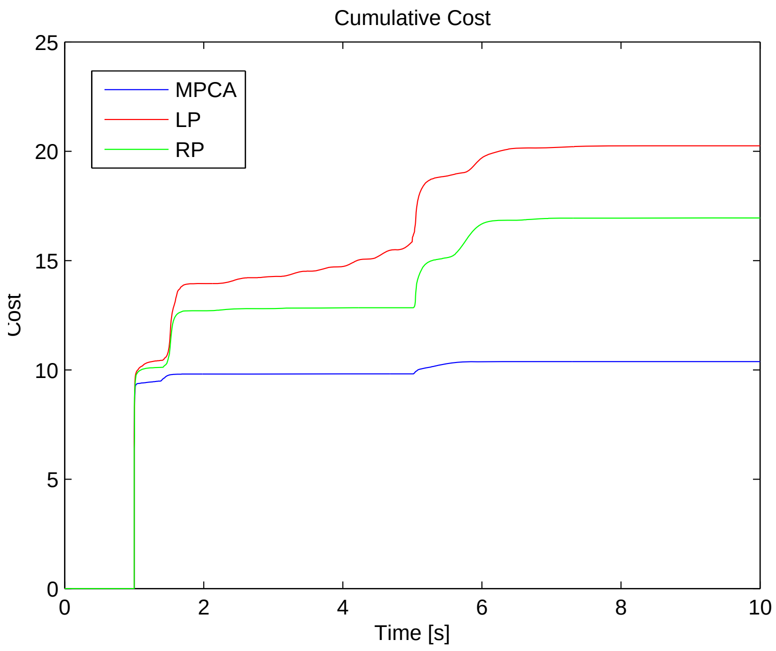

Figure 6.11: Part III - Agile flight: Cumulative cost.

6.4 Part IV - Honoring Actuator Constraints

This test will look at two versions of the MPCA control allocation method. One is equal to the one in Part III, while the other one is modified by removing the constraints on δ_{cmd}. First, one can look at the trajectory tracking in Figure 6.12, and note that they are very similar. The main difference is in the actuator response, shown in Figure 6.13. Here one can see that the actuator limits, shown in dashed black lines, are not honored.

This has connection to the actuator's control commands δ_{cmd}, shown in Figure 6.14. The cumulative cost for the two MPCA versions are shown in Figure 6.15.

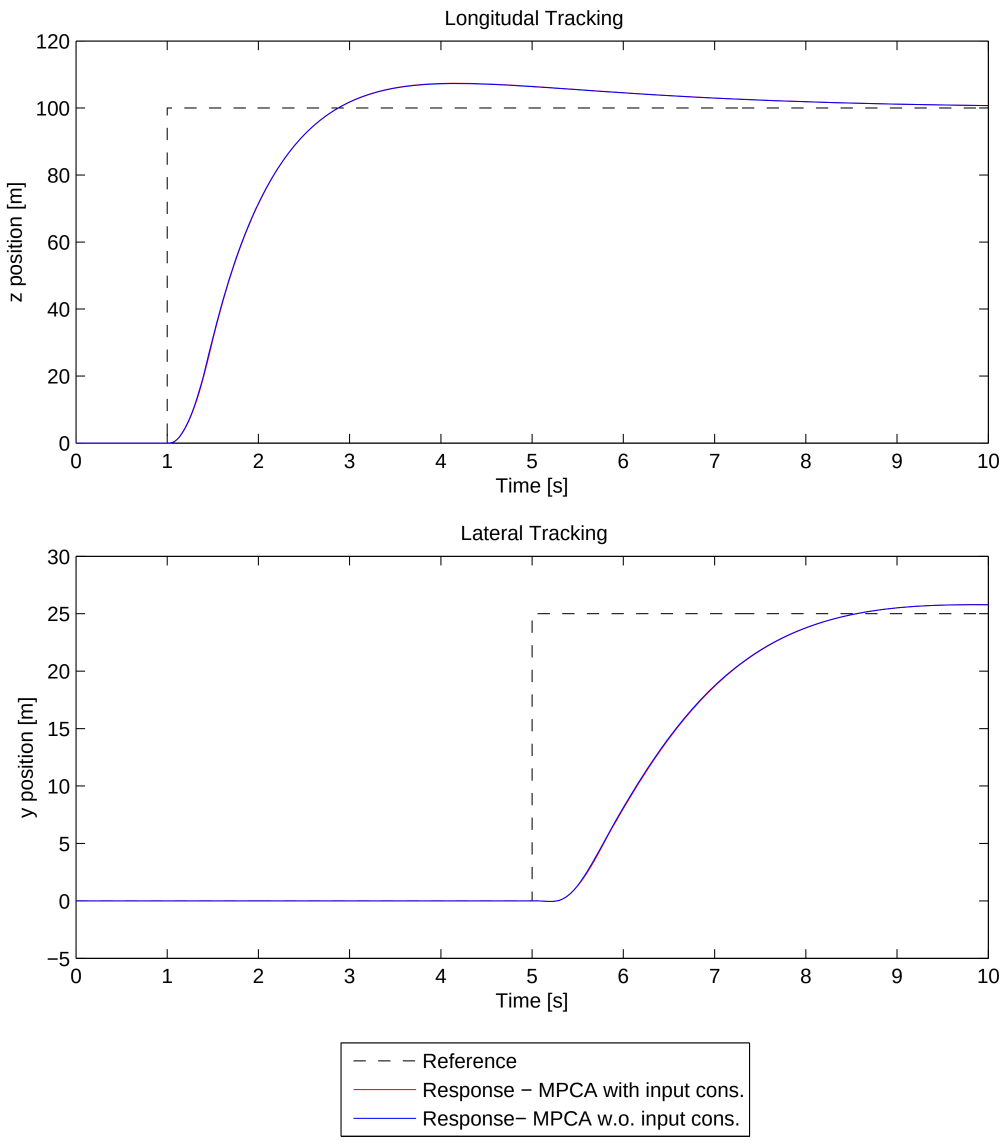

Figure 6.12: Part IV - Honoring Actuator Constraints: Trajectory tracking.

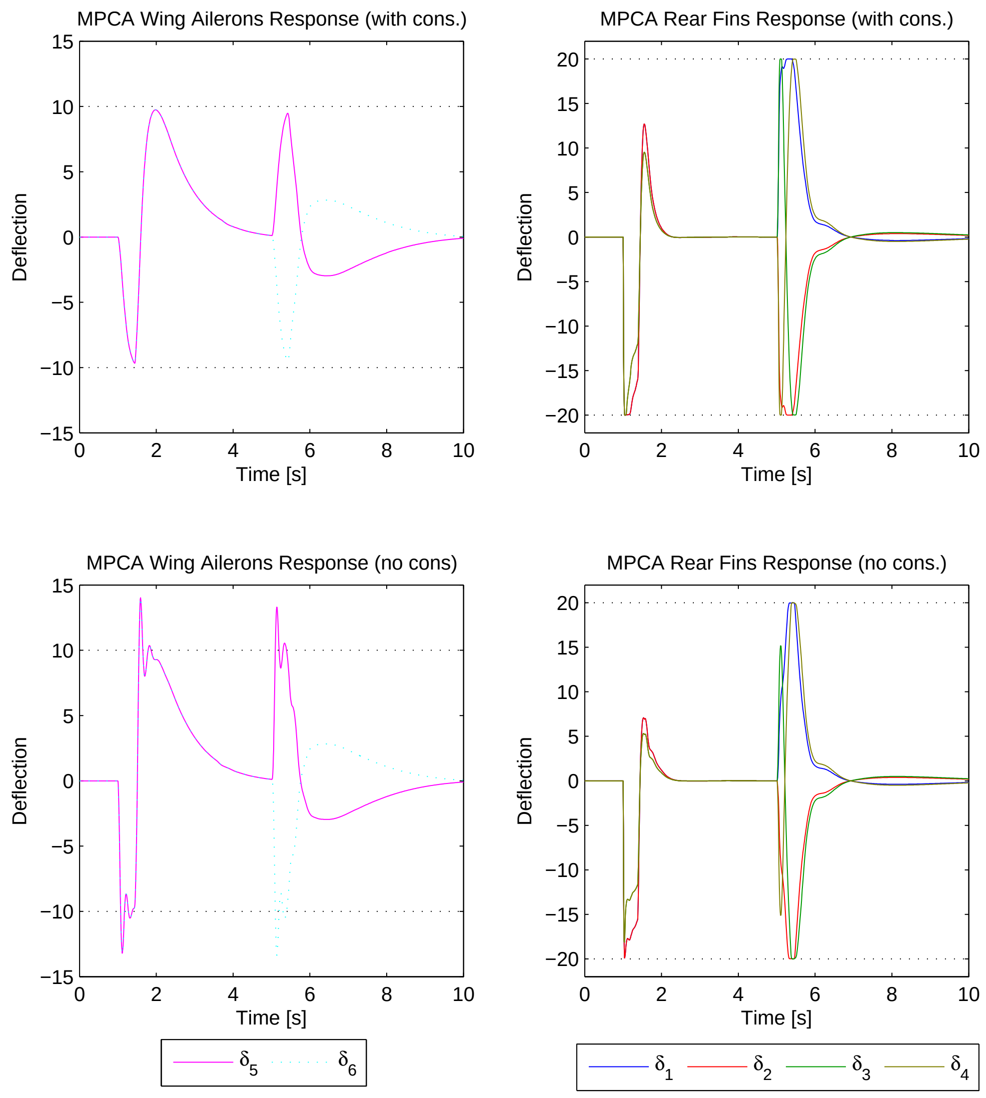

Figure 6.13: Part IV - Honoring Actuator Constraints: Actuator response. Left: Wing ailerons. Right: Tail fins.

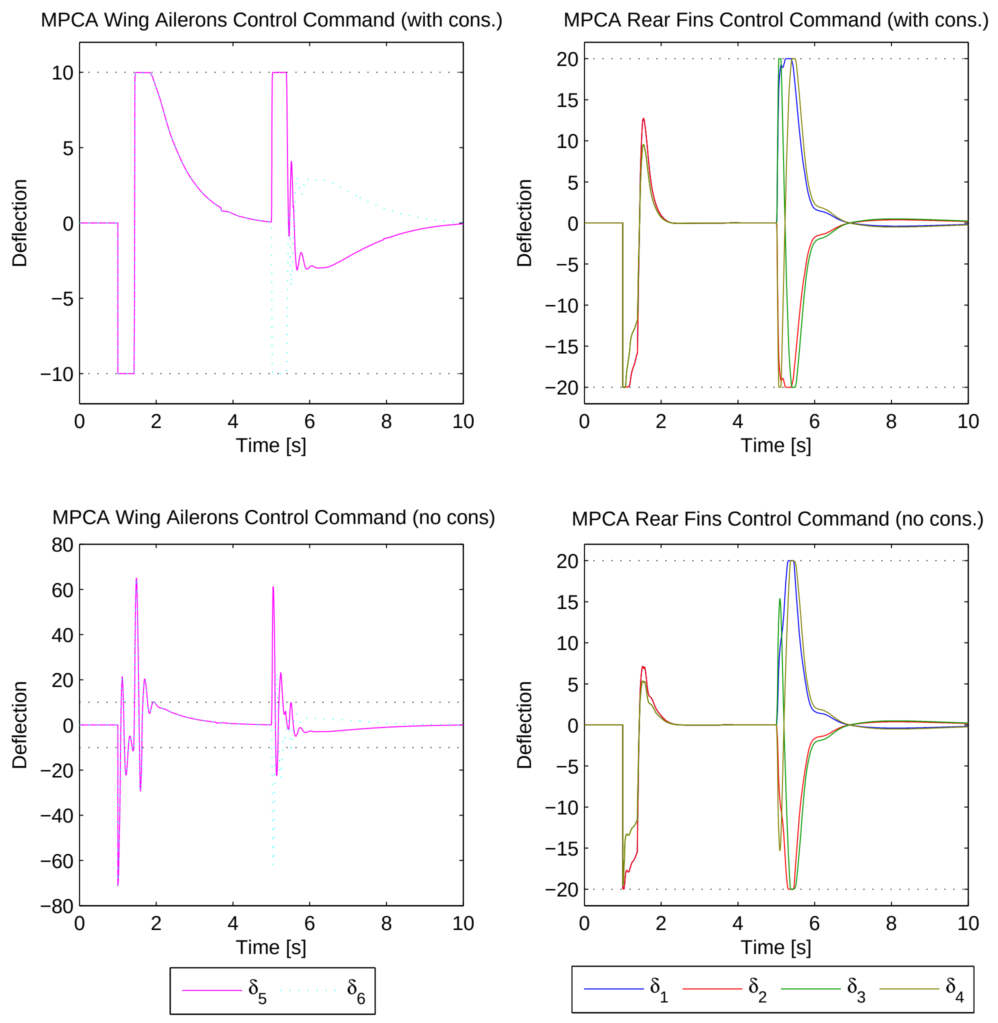

Figure 6.14: Part IV - Honoring Actuator Constraints: Actuator command. Left: Wing ailerons. Right: Tail fins.

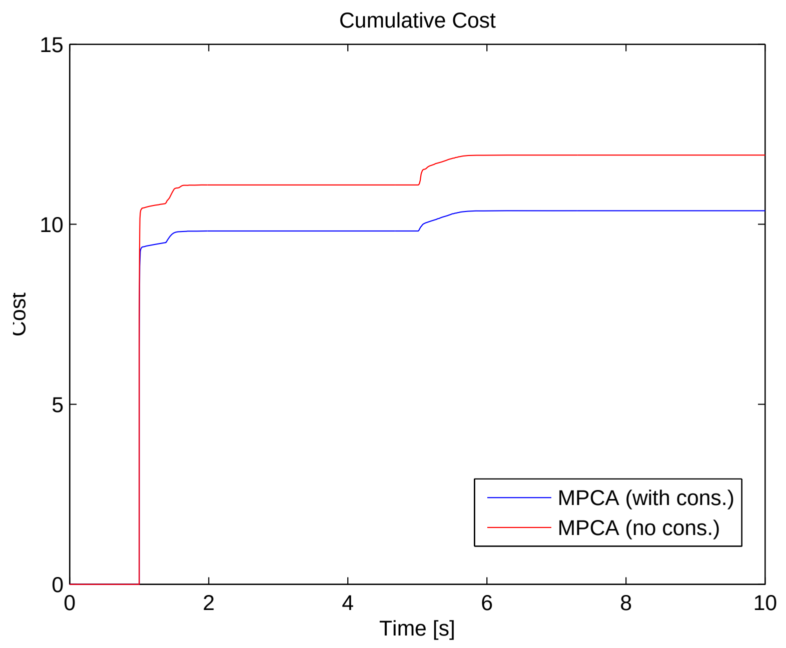

Figure 6.15: Part IV - Honoring Actuator Constraints: Cumulative cost.

6.5 Part V - Prediction Comparison

This part compares the MPCA with different virtual control predictions, type 1 (denoted MPCA) and type 3 (denoted MPCA+), representing low- and high-tier solutions, respectively. The RP method is also included in the test to serve as a contrast to the MPCA versions. The missile is given longitudal and lateral step trajectories to track, see Figure 6.16.

The actuator response is shown in Figure 6.17, and shows that the type 3 prediction MPCA will allocate actuators already before the steps occur. This is not the case for the one with type 1, as it has no means of predicting the step. Figure 6.18 shows the virtual control tracking, where the type 3 prediction MPCA has a slightly better tracking than the one with type 1.

Lastly, the cumulative cost is shown in Figure 6.19. It comes as no surprise that the type 3 MPCA accumulates the least cost, while the type 1 MPCA and RP follow suit.

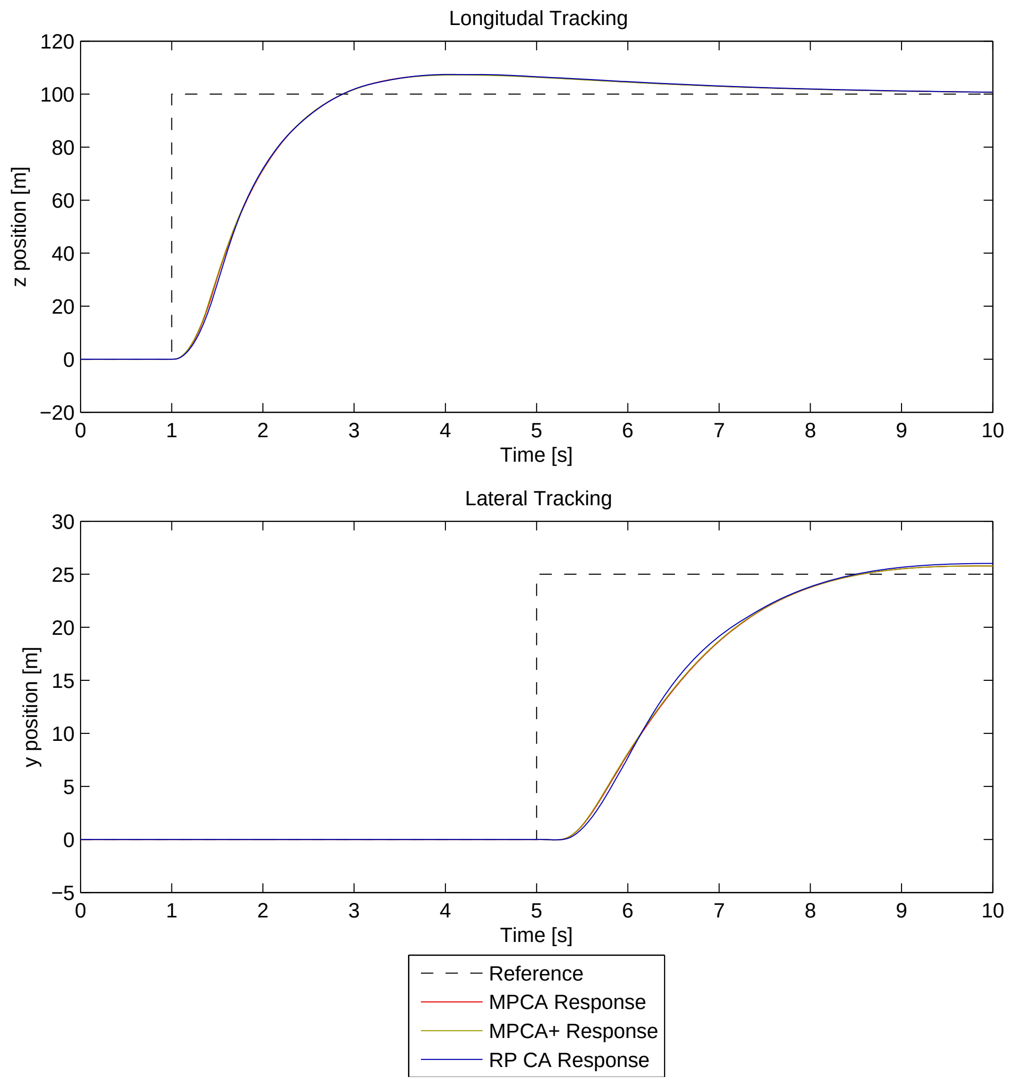

Figure 6.16: Part V - Prediction Comparison: Trajectory tracking.

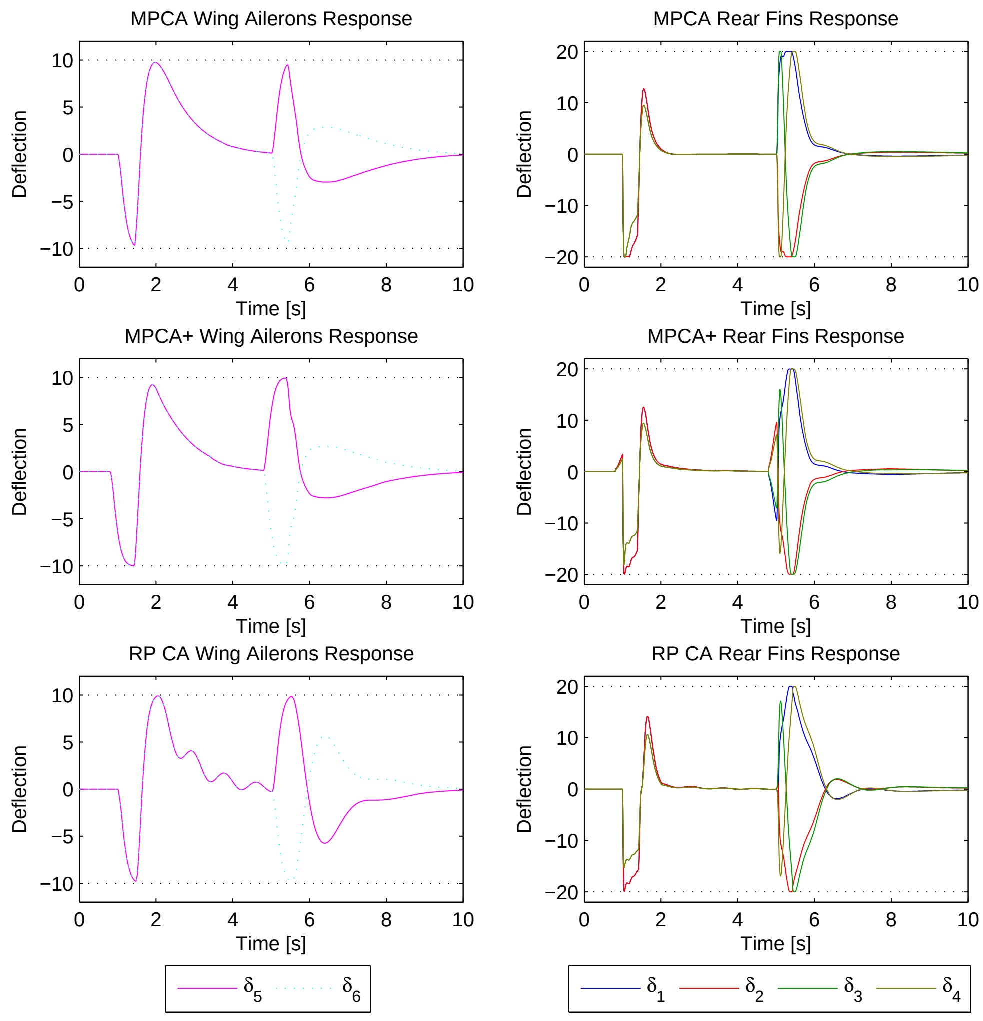

Figure 6.17: Part V - Prediction Comparison: Actuator response. Left: Wing ailerons. Right: Tail fins.

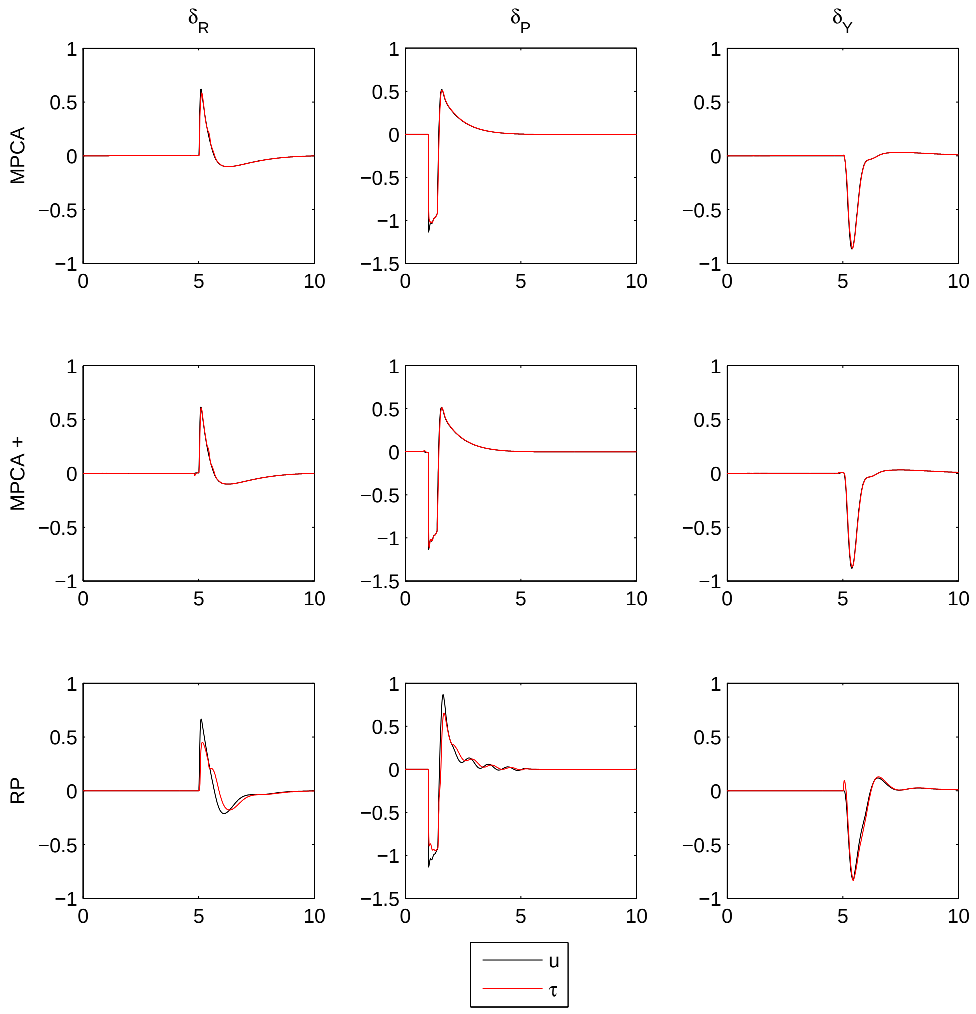

Figure 6.18: Part V - Prediction Comparison: Virtual control tracking. Left: Wing ailerons. Right: Tail fins.

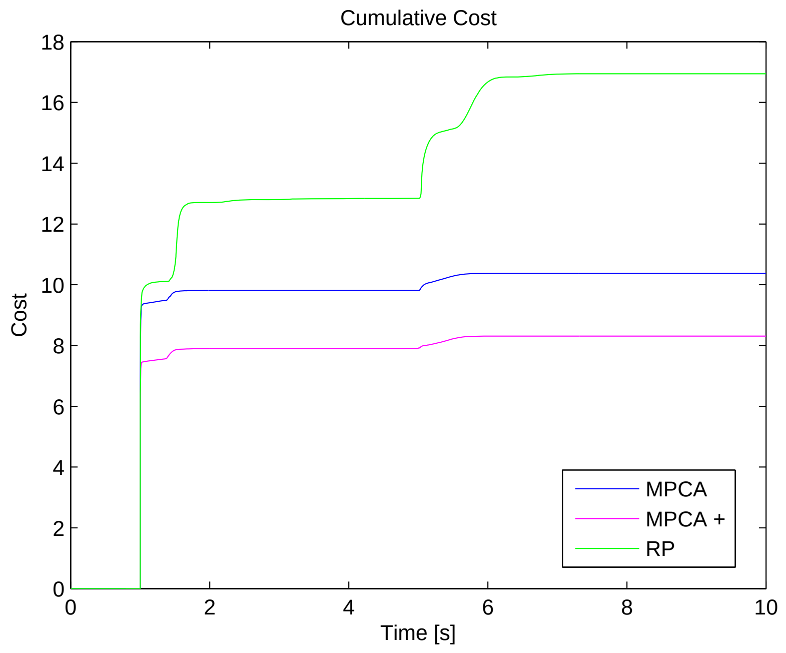

Figure 6.19: Part V - Prediction Comparison: Cumulative cost.

Chapter 7

Discussion

This chapter presents a discussion on the performance of the Model Predictive Control Allocation method. The method is compared to the methods Linear Programming and Redistributed Pseudoinverse, emphasizing drawbacks and advantages, aswell as applications of the MPCA. The discussion is based on the results presented in Chapter 6, which correpond to the test plan in Section 5.4. The performance of the CVXGEN solver will also be evaluated. It is important to emphasize that the various missile trajectories are of less importance - they are just a means of testing the control allocation module, whose performance is judged by the actuator allocation and virtual control tracking.

7.1 Test with tail fins only

It follows naturally to start of with Part I of the results, found in Section 6.1. This is a test where the missile is equipped with four tail fins which have similar and fast dynamics. Looking at the step response in Figure 6.1, the three methods behave quite similarly. One can note that the LP method is trailing slightly behind the other two, and the reason for this is made clear by studying Figure 6.2. While the MPCA and the RP method have very similar actuator response, the LP method chooses to saturate δ_1 and δ_2 while keeping the deflection of δ_3 and δ_4 low. This makes the tracking performance poorer than for the two other methods, which again can be viewed in Figure 6.3. Here, the LP chooses to track δ_Y tighter than δ_R, which is a bad choice considering that this is a bank-to-turn missile, which utilizes roll to turn.

Setting the tracking performance aside, the main result of this test is that in such a setup the performances of the three methods are very similar. Very little separates the methods in terms of tracking trajectory and virtual control. This is because the actuators all have similar and fast characteristics - in such a setup the MPCA method can to a lesser extent exploit having knowledge of the actuator dynamics, because they are so fast in the first place. This is made especially clear by comparing the MPCA and RP method - two methods on the opposite side of the complexity scale - as they yield almost exactly the same actuator response.

Because of this, the MPCA method is perhaps not a good choice in a setup where all actuators are fast, as it is unable to exploit its strong traits. Since a simple method like the RP yield the same results, this is probably a better choice because it is significantly less complex to develop and implement.

7.2 Performance in trimmed flight

Turning the attention to the next part of the test (Section 6.2), the situation is different. Now the missile is equipped with wing ailerons in addition to the tail fins. These actuator groups have different dynamics, a setup which the MPCA method can benefit from. This part of the test is simulating a trimmed flight scenario, i.e. flight under calm conditions. As previously mentioned, it is desirable to use only the wing ailerons under such conditions, as they are less costly than the tail fins. The missile is given a sinusoidal longitudal trajectory to follow, while the lateral reference is held at zero (see Figure 6.4). The reason for this is that lateral movement requires control in two axes, i.e. $m = 2$. If only wing ailerons are utilized, $n = m = 2$, and the autopilot will have a very hard time controlling the missile. In fact, it forces allocation of the tail fins to perform the maneuvre. On the other hand, longitudal movement requires only control in one axis, making $m = 1$. Now $n > m$, and the control allocation module will be able to use only the wing ailerons as control, thus making the trimmed flight test possible.

Looking at the actuator response in Figure 6.5, it is clear that the wing ailerons (on the left side) are utilized the most. The LP method is unaware of the actuator dynamics, causing oscillation at the end of the maneuvre. The MPCA and the RP do not suffer from this oscillation, but looking at the virtual control tracking in Figure 6.6 reveals that the MPCA provides better tracking performance. This is summarized in the cumulative cost plot in Figure 6.7. The LP causes oscillations, and accumulates a high cost because of the excessive actuator usage. The RP method is unaware of the actuator dynamics, and is penalized cost-wise because of the trailing tracking performance of the virtual control. The MPCA method is connected with the lowest cost, well below that of the other methods.

Based on this, one can say that the MPCA method is well suited for use in a trimmed flight scenario. It accomplishes the goal of using only the inexpensive wing ailerons, though some residual allocation of the tail fins is present due to nonzero cost on the wing ailerons. Compared to the other two methods, it is clear that the MPCA exploits its knowledge of the actuator dynamics, causing the least usage of the tail fins and the best tracking of the virtual control input.

The fact that the LP method causes oscillations shows that the choice of control allocation method actually influences the performance of the overall system. Because of this the choice should not be arbitrary, but be a consideration between method complexity and execution speed, which actuator constraints the method can handle, which configuration the actuators are in, and finally how it behaves in the overall flight control system.

A prime example of this is the LP method. In the previous project work done on control allocation [22], the method was compared to the RP and QP methods on a system where actuator dynamics were ignored, i.e. actuators had a transfer function equal to one. In that work, the LP method was considered to be the best of the three. In the present thesis, where actuator dynamics are present and actuators span different dynamic authorities, the method causes unwanted oscillations which are unacceptable. With this fact in hand, one can draw the conclusion that the LP method does not make the transition to a system with actuator dynamics very well, rendering it less suitable for such systems. The RP method, though less complex, surprisingly makes this transition better.

7.3 Performance in agile flight

Part 3 of the test sets the missile in a condition simulating agile flight. This is simulated by making the missile follow lateral and longitudal steps. This puts stress on the control allocation module, forcing it to utilize all available actuators, not just the wing ailerons. The trajectory and the missile response can be seen in Figure 6.8. Again, the three methods yield roughly the same response, but in the lateral tracking the MPCA is somewhat better than the rest.

The actuator response in Figure 6.9 reveals that the LP method, like in the previous test, causes oscillations, but this time this is also the case for the RP method. These methods are unaware of the slow actuator dynamics of the wing ailerons, and commands them as if they would respond immediately, resulting in the unwanted oscillation. Looking at the MPCA method, it shows no signs of oscillatory behaviour and clearly allocates control in the most satisfactory manner. All three methods allocate the tail fins similarily, though the LP method saturate the actuators more, which is not desireable. If possible, the actuators should never be saturated - then there always will be excess control power available in every direction, to handle disturbances and unexpected commands.

In the virtual control tracking, Figure 6.10, the same conclusions as in the previous paragraph can be drawn, and one can see that it is in the pitch axis that the oscillations appear (this is probably related to that the pitch axis controllers have the highest gains). Figure 6.11 shows the cumulative cost, and as for Part 2 of the test, the MPCA accumulates the lowest cost, while the RP and LP method follow second and third. This was to be expected, based on the oscillatory movements caused by the RP and LP movements, which create the need for additional actuator usage.

7.4 Honoring actuator constraints

In the next part of the testing, the missile is tasked with following the same step trajectory as in part 3, but this time two versions of the MPCA method are compared. The first version is equal to that of part 3, but the other is without constraints on the control command δ_{cmd} (which is is the output of the MPCA).

The step trajectory and missile response is shown in Figure 6.12, and reveal that the two versions of the MPCA yield almost identical missile response. What is more interesting is how the actuators response, which is shown in Figure 6.13. For the case without constraints on δ_{cmd}, the wing ailerons exceed their limit values of $\pm 10°$. This is unacceptable since honoring these limits is one of the key properties of a control allocator.

By looking at the actuator response for the case with constraints on δ_{cmd} in the same figure, one can notice that the wing aileron responses are similar, but have smaller slopes and most importantly do not exceed their limits. Some explanation for this lies in the plot of the actuators commanded control δ_{cmd} in Figure 6.14. For the case with constraints on δ_{cmd}, the command saturates on the limits, and thus the slope of δ is constrained. This also limits the actuators to ever exceed their limits. Next, in the case without the constraints on δ_{cmd}, the $\pm 10°$ limits are breached several times. This in itself is of no particular concern, since the important bounds are on δ. By commanding a higher deflection than the limits, the control allocation module can force the actuators to respond faster, since there is in this case no limits on the actuator speed. This can of course be a good thing, in essence it is just exploiting the knowlegde of the actuator dynamics to achieve better performance. The problem occurs when the actuator actually exceeds it limits. It is clear that the control allocation module is aware of this since it suddenly switches commanded direction when this happens. Looking back at Figure 6.13, this is reflected by observing that the actuators have an oscillating motion about their limits, converging towards them.

One may ask why the actuator does not honor the constraints, considering that the control allocation is aware of the actuator's dynamics. The reason for this is probably based on the MPCA's virtual control prediction. In this test, this prediction is type 1, which is a vector repeating the current value of u. This is not a particularly good prediction taking the future sudden step of the reference trajectory into account. The MPCA receives no prediction of the future dynamics, and when the virtual control u suddenly flats out, the damage has already been done since the actuator has been commanded further than it should have been, with no chance of honoring the bounds. The MPCA tries to compensate by reversing it's direction, but due to the dynamics of the actuator this does not happen instantaneously, thus the oscillation motion.

This reveals one of the weaknesses of the MPCA - for it to fully take use of its knowledge of the actuator dynamics, it needs a reasonable prediction of the future dynamics. By constraining the commanded control δ_{cmd} one can make sure the actuator is held between its limits regardless of the quality of the prediction. This is perhaps not such a bad solution, since the available power may not be large enough to command the actuator to achieve such slopes as seen in the lower left plot in Figure 6.14. In fact, the available power may be constrainted between $[\delta_{min}, \delta_{max}]$, because the actuators are designed not to exceed these limits. By looking at Figure 6.15, the MPCA with constraints on δ_{cmd} actually accumulates lower cost than the one without.

7.5 On virtual control prediction

Part V of the test (described in Section 5.4.5) takes on various types of virtual control predictions. Three different types are described in Section 5.3, but only type 1 and type 3 are used in the test. These two represent simple and more complex solutions respectively. Jumping to Figure 6.17, the type 3 prediction MPCA (MPCA+) clearly is aware of the trajectory steps before they occur, since it allocates actuators prematurely (middle row). It is also worth noting that the MPCA+ also saturates the actuators less than the MPCA with prediction type 1 (top row), which is a desireable property.

One would expect the MPCA+ with a $0.2s$ prediction window would make the missile's step response (Figure 6.16) $0.2s$ faster, but this is clearly not the case. Though the MPCA+ actually has a slightly faster response, it is very small and not clearly visible on a 10-second scale. Where the benefits of using type 3 prediction becomes clear is in the virtual control tracking (Figure 6.18). In this regard it can show off an almost exact tracking, being able to track even directly after the reference trajectory steps, which the MPCA type 1 can not. This is reflected by studying the accumulated cost (Figure 6.19), where using type 3 prediction cuts the cost by roughly 20% compared to the MPCA type 1.

7.6 CVXGEN

Setting the control allocation method comparison aside, a short discussion on the applied convex optimization solvers is held. In this thesis, the newly developed CVXGEN solver is utilized when solving the MPCA problem, aswell as the LP CA problem. The control allocation method testing was performed within the MATLAB environment, and the CVXGEN online interface provides solvers for this environment directly.

The online interface is user friendly and intuitive, and the process of entering a problem to having an automatically generated custom solver ready to solve it is done in minutes. Of course, since the interface is only available online there is a need for an internet connection, which does restricts its availability. If the custom solver generator could be downloaded and installed as a program the flexibility would increase, making the user experience even better. On the other hand, this may restrict the rate of bug fixing and further code optimization. Since CVXGEN is, at the time of this writing, not commercially available, the online solution will have to suffice, but to the author's opinion there is no doubt that the product has great potential. During the span of the thesis' writing, no errors were encountered and the solvers always performed well.

The use of CVXGEN leads to a customized quadratic programming solver, and used in the MATLAB environment the solver typically requires less than 1 millisecond computation time on a powerful processor. This is considered to be very fast, and there is no reason to assume that the custom c-code the CVXGEN interface can generate executes any slower. This makes implementation on a microprocessor for real-time applications feasible, although important aspects such as software code verifiability needs to be addressed carefully.

The CVXGEN problem solve time also scales well with complexity. Figure 7.1 shows the average solve time (in the MATLAB environment) for an MPCA problem where $m = 3$, $n = 6$, and the prediction horizon T was incrementally increased from 1 to 7. CVXGEN works best when the number of nonzero KKT[1] entries are below 4000 [27].

A nearly linear relationship between problem complexity and average solve time can be observed from $T = 1$ to $T = 6$. On the step from $T = 6$ to $T = 7$ the slope increases. This is also where the complexity (number of nonzero KKT entries) exceeds 4000, confirming that CVXGEN works best with complexity below this number. In fact, when a problem complexity exceeds 4000, a warning is displayed in the CVXGEN interface, stating that a solution to the problem may not be found. MPCA problem size statistics are presented in Table 7.1.

CVXGEN is also a handy tool in prototyping and testing, where the solve times are of lesser concern but still important (shorter solve times allows for more rigorous and frequent tests). The solve time difference from other commonly used convex optimization parser solvers like CVX is staggering. Figure 7.2 shows a comparison between CVX and CXVGEN for an MPCA problem where $m = 3$, $n = 6$ and $T = 1$. Obviously CVXGEN is the better choice, especially when also considering that parser solvers like CVX and Yalmip require the same amounts of code writing and adaption as CVXGEN.

[1]Karush-Kuhn-Tucker conditions, which are important in solving optimization problems.

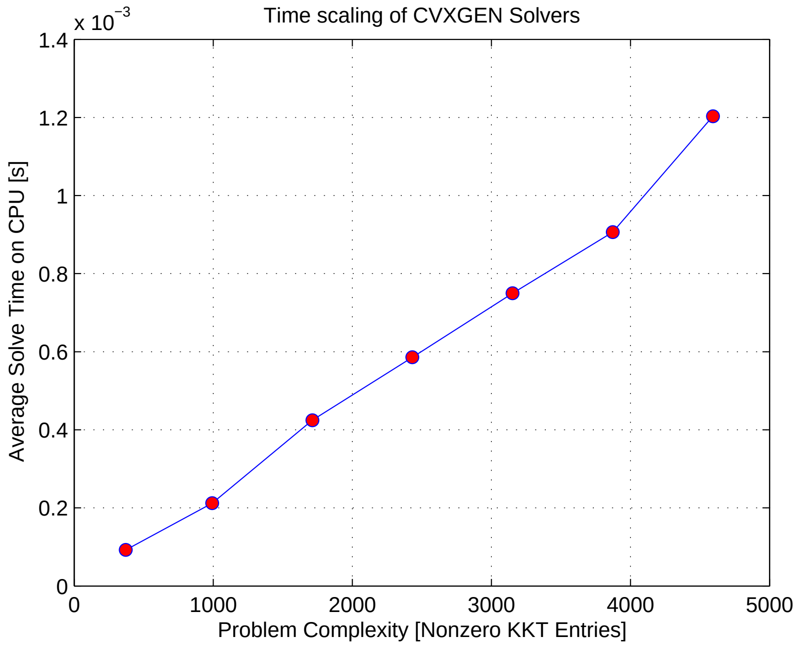

Figure 7.1: CVXGEN solve time scaling.

Prediction horizon T	1	2	3	4	5	6	7
Parameter entries	286	289	292	295	298	301	304
Original variables	24	45	66	87	108	129	150
Variables in solver	33	63	93	123	153	183	213
Equalities in solver	27	51	75	99	123	147	171
Inequalities in solver	24	48	72	96	120	144	168
KKT original non-zeros	249	639	1029	1419	1809	2199	2589
KKT non-zeros after fill-in	372	993	1713	2433	3153	3873	4593
Solver generation time	$1.8s$	$9.5s$	$24.2s$	$48.3s$	$82.0s$	$111.5s$	$159.2s$

Table 7.1: MPCA Optimization Problem Size Statistics

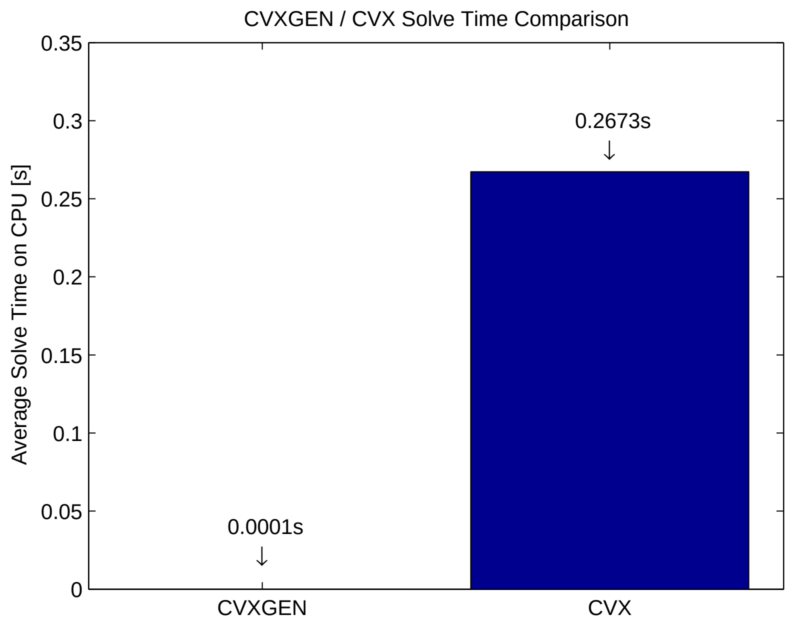

Figure 7.2: CVXGEN vs CVX solve time comparison.

Chapter 8

Conclusion and Further Work

This thesis has developed and tested a control allocation method based on the Model Predictive Control algorithm. The MPCA method can handle actuator constraints on position and velocity, aswell as accounting for actuator dynamics. This sets it aside from other classical methods, which neglects these dynamics on the basis that they are orders of magnitude faster than the missile dynamics, or that they are accounted for in another part of the flight control system. This can in some cases be an unrealistic assumption.

The MPCA method was compared to the Linear Programming- and Redistributed Pseudoinverse control allocation methods in a setting where actuators have dynamics spanning different dynamical authorities. Here, the MPCA outperformed the other methods in regard to cost and tracking performance. Using knowledge of the actuator dynamics, it can allocate different groups of actuators under different conditions, which is beneficial with regard to aspects like power consumption and actuator wear.

In the various tests of the MPCA, some issues were also uncovered. The advantages of MPCA do not really appear if used when the actuator set is fast and have homogenous dynamics and cost. Because all actuators react almost immediately, there is no need for a scheme planning the movements of the actuators in advance - methods neglecting actuator dynamics perform just as well. Also, the quality of the virtual control prediction needs to be a concern when designing the MPCA. One test revealed that actuator limits may be violated if the prediction quality is low and the MPCA has no bounds on actuator command. By adding such bounds, one can ensure that actuator limits always are honored. By increasing the quality of the prediction, cost is reduced and tracking performance increased.

CVXGEN is used to solve the MPCA problems, and performs the job brilliantly. The solve times are very low, making implementation in a real-time application feasible.

Further Work

In this thesis, the flight control system was designed with the goal of testing the control allocation module. Because of this, other parts like the autopilot were not emphasized. It would be interesting to implement the MPCA in a flight control system tailored to a specific aircraft.

A further step could be taking the use of CVXGEN to the next level, by implementing the MPCA on a microprocessor, using the custom c code. This would probably pose some challenges, but would reveal its potential in real-time applications.

More further work can be done in combining the autopilot and control allocation into one unit. Since using the Model Predictive Control Allocation method requires an estimate of the future virtual control, and this signal comes from the autopilot, combining the two methods could be feasible. This would require an estimate of the future trajectory, but this is often more available or easier to estimate than the virtual control. Since the missile dynamics can be simplified into first order linear differential equations, these can be incorporated into the hybrid MPC-autopilot-control allocator, enabling it to account for the missile dynamics, further increasing the module's performance. This is an interesting yet not unrealistic goal for future projects on MPCA.

Bibliography

[1] B. L. Stevens and F. L. Lewis, *Airplane Flight Dynamics and Automatic Flight Controls.* Darcorporation, 1992.

[2] W. Durham, "Constrained Control Allocation," *Journal of Guidance, Control and Dynamics*, vol. 16, no. 4, 1993.

[3] T. A. Johansen, T. P. Fuglseth, P. Tondel, and T. I. Fossen, "Optimal Constrained Control Allocation in Marine Surface Vessels with Rudders," *Control Engineering Practice*, vol. 16, pp. 457–464, 2008.

[4] D. L. Raney, R. C. Montgomery, and L. L. Green, "Flight Control Using Distributed Shape-Change Effector Arrays," in *Proceedings of the 41st AIAA Structures, Structural Dynamics and Materials Conference*, 2000.

[5] O. Härkegård and T. S. Glad, "Resolving Actuator Redundancy - Optimal Control vs. Control Allocation," in *Proceedings of the European Control Conference, Cambridge UK*, 2003.

[6] Y. Luo, A. Serrani, S. Yurkovich, and D. B. Doman, "Model Predictive Dynamic Control Allocation with Actuator Dynamics," in *Proceedings of the 2004 American Control Conference, Boston, MA, USA*, 2004.

[7] Y. Luo, A. Serrani, S. Yurkovich, and D. B. Doman, "Dynamic Control Allocation with Asymptotic Tracking of Time-Varying Control Input Commands," in *Proceedings of the 2005 American Control Conference, Portland, OR, USA*, 2005.

[8] M. W. Oppenheimer and D. B. Doman, "Control Allocation for Overactuated Systems," in *Proceedings of 14th Mediterranean Conference on Control and Automation*, 2006.

[9] C. Vermillion, J. Sun, and K. Butts, "Model Predictive Control Allocation for Overactuated Systems - Stability and Performance," in *Proceedings of the 46th IEEE Conference on Decision and Control, New Orleans, USA*, 2007.

[10] T. A. Johansen, "Optimizing Nonlinear Control Allocation," in *Proceedings of the IEEE Conf. Decision and Control, Nassau, Bahamas*, pp. 3435–3440, 2004.

[11] J. Tjønnås and T. A. Johansen, "Optimizing Adaptive Control Allocation with Actuator Dynamics," in *Proceedings of the IEEE Conf. on Decision and Control, New Orleans, USA*, 2007.

[12] T. A. Johansen, T. I. Fossen, and P. Tøndel, "Efficient Optimal Constrained Control Allocation via Multiparametric Programming," *Journal of Guidance, Control and Dynamics*, vol. 28, no. 3, 2005.

[13] P. Tøndel and T. A. Johansen, "Control Allocation for Yaw Stabilization in Automotive Vehicles using Multiparametric Nonlinear Programming," in *Proceedings of the American Control Conference, Portland, USA*, 2005.

[14] J. Spjøtvold and T. A. Johansen, "Fault Tolerant Control Allocation for a Thruster-Controlled Floating Platform using Parametric Programming," in *Proceedings of the IEEE Conf. Decision and Control, Shanghai, People's Republic of China*, 2009.

[15] M. Bodson, "Evaluation of Optimization Methods for Control Allocation," *Journal of Guidance, Control and Dynamics*, vol. 25, no. 4, 2002.

[16] M. Bodson and J. A. M. Petersen, "Constrained Quadratic Programming techniques for Control Allocation," in *Proceedings of the IEEE Conf. Decision and Control, Maui, Hawaii*, 2003.

[17] O. Härkegård, "Dynamic Control Allocation Using Constrained Quadratic Programming," *Journal of Guidance, Control, and Dynamics*, vol. 27, pp. 1028–1034, 2004.

[18] E. Ruth and A. J. Sørensen, "A Solution to the Nonconvex Linearly Constrained Quadrating Thrust Allocation Problem," in *Proceedings of 8th Conference on Manoeuvring and Control of Marine Craft, Guaruj, Brasil*, pp. 195–200, 2009.

[19] T. A. Johansen, T. I. Fossen, and S. P. Berge, "Constrained Nonlinear Control Allocation with Singularity Avoidance using Sequential Quadratic Programming," *IEEE Trans. Control Systems Technology*, vol. 12, pp. 211–216, 2004.

[20] J. Mattingley and S. Boyd, "Automatic Code Generation for Real-Time Convex Optimization." Chapter in Convex Optimization in Signal Processing and Communications, Y. Eldar and D. Palomar, Eds., Cambridge University Press,, 2009.

[21] J. Mattingley, Y. Wang, and S. Boyd, "Code Generation for Receding Horizon Control," *Proceedings IEEE Multi-Conference on Systems and Control, Yokohama, Japan*, pp. 985–992, 2010.

[22] M. Hanger, "Control Allocation," tech. rep., Department of Engineering Cybernetics, Norwegian University of Science and Technology, 2010.

[23] J. Cloutier, J. Evers, and J. Feeley, "Assessment of Air-to-Air Missile Guidance and Control Technology," *IEEE Control Systems Magazine*, vol. 9, 1989.

[24] W. Durham, "Attainable Moments for the Constrained Control Allocation Problem," *Journal of Guidance, Control and Dynamics*, vol. 17, no. 6, 1994.

[25] R. E. Beck, *Application of Control Allocation Methods to Linear Systems with Four or More Objectives.* PhD thesis, Virginia Polytechnic Institute and State University, 2002.

[26] Maciejowski, *Predictive Control with Constrains.* Prentice Hall, 2002.

[27] J. Mattingley and S. Boyd, "CVXGEN: A Code Generator for Embedded Convex Optimization," *CVXGEN Website*, 2010. `http://cvxgen.com/`.

[28] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*, ch. 5. SIAM, 2000.

[29] S. Boyd and L. Vandenberghe, *Convex Optimization.* Cambridge University Press, 2004.

[30] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming." `http://www.stanford.edu/~boyd/cvx/`, 2008.

[31] J. Löfberg, "YALMIP: A toolbox for modeling and optimization in MATLAB," in *Proceedings of the CACSD Conference, Taipei, Taiwan*, 2004. `http://control.ee.ethz.ch/~joloef/yalmip.php`.

[32] K. A. Bordignon, *Constrained Control Allocation for Systems with Redundant Control Effectors.* PhD thesis, Virginia Polytechnic Institute and State University, 1996.

[33] D. Enns, "Control Allocation Approaches," in *Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit, Reston VA*, pp. 98–108, 1998.

[34] J. M. Buffington, "Modular Control Law Design for the Innovative Control Effectors (ICE) Tailless Fighter Aircraft Configuration 101-3," tech. rep., US Airforce Research Lab., Wright Patterson AFB, OH., 1999.

[35] M. W. Oppenheimer, D. B. Doman, S. Yurkovich, A. Serrani, and Y. Luo, "Model Predictive Dynamic Control Allocation with Actuator Dynamics," in *Proceedings of the 2004 American Control Conference*, 2003.

[36] T. I. Fossen, "Mathematical Models for Control of Aircraft and Satellites," tech. rep., Department of Engineering Cynernetics, Norwegian University of Science and Technology, 2011.

[37] A. Skullestad, "Missile Dynamic and Control." Lecture notes in the course Mathematical modelling. Department of Computer Science, Electrical Engineering and Space technology, HiN Norway, 2003.

[38] T. I. Fossen, *Guidance and Control of Ocean Vehicles.* John Wiley & Sons Ltd., 1994.

[39] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control.* John Wiley & Sons Ltd., 2011.

Appendix A

Model Parameters

Tables A.1, A.2 and A.3 presents the model parameters used in the simulations.

Table A.1: Model Parameters

Symbol	*Value*	*Unit*
Q	55350	$\frac{kg}{ms^2}$
m	200	kg
I_{xx}	5	kgm^2
I_{yy}	150	kgm^2
I_{zz}	150	kgm^2
I_{xy}	1	kgm^2
I_{xz}	1	kgm^2
V	300	$\frac{m}{s}$
s	1	m^2
l	1	m

Table A.2: Aerodynamic Control Parameters

Symbol	*Value*	*Unit*
$C_{\mathcal{L}\delta_R}$	0.6	rad^{-1}
$C_{\mathcal{C}\delta_R}$	0.6	rad^{-1}
$C_{N\delta_R}$	0.6	rad^{-1}
$C_{\mathcal{L}\delta_P}$	0.5	rad^{-1}
$C_{\mathcal{C}\delta_P}$	0	rad^{-1}
$C_{N\delta_P}$	0	rad^{-1}
$C_{M\delta_P}$	0.5	rad^{-1}
$C_{\mathcal{L}\delta_Y}$	0	rad^{-1}
$C_{N\delta_Y}$	0.4	rad^{-1}
$C_{\mathcal{C}\delta_Y}$	0.4	rad^{-1}

Table A.3: Aerodynamic Parameters

Symbol	*Value*	*Unit*
$C_{\mathcal{L}\alpha}$	5.0	rad^{-1}
$C_{M\alpha}$	-5.0	rad^{-1}
$C_{\mathcal{L}\beta}$	0	rad^{-1}
$C_{\mathcal{C}\beta}$	0.8	rad^{-1}
$C_{N\beta}$	0	rad^{-1}
$C_{\mathcal{L}r}$	-0.7	rad^{-1}
C_{Nr}	-8.0	rad^{-1}
C_{Np}	0.1	rad^{-1}
$C_{\mathcal{L}p}$	-10.0	rad^{-1}

Appendix B

Missile Model Nomenclature

u	Longitudal (forward) velocity
v	Lateral (transverse) velocity
w	Vertical velocity
p	Roll rate
q	Pitch rate
r	Yaw rate
ϕ	Roll angle
θ	Pitch angle
ψ	Yaw angle
α	Angle of attack
β	Sideslip angle
x_E	Earth-fixed x position
y_E	Earth-fixed y position
z_E	Earth-fixed z position
v_t	Total speed
V	Speed relative to wind
ν	Velocity vector
η	Position and attitude vector
τ	Vector of forces and moments
X	Longitudal force
Y	Lateral force
Z	Vertical force
L	Roll moment
M	Pitch moment
N	Yaw moment
$\mathcal{D}$	Longitudal drag force
$\mathcal{C}$	Lateral drag forces
$\mathcal{L}$	Lift force

$\mathbf{R}_b^a$	Rotation matrix from b to a
$\mathbf{R}_{x,\phi}$	Euler angle rotation about x axis
$\mathbf{R}_{y,\theta}$	Euler angle rotation about y axis
$\mathbf{R}_{z,\psi}$	Euler angle rotation about z axis
m	Mass
a_i	Acceleration along axis i
I_{CG}	Inertia tensor
M_{RB}	Rigid body inertia matrix
C_{RB}	Rigid body damping matrix
τ_{RB}	Rigid body forcing vector
I_{ii}	Moment of inertia
I_{ij}	Product of intertia
g_i	Gravitational acceleration along axis i
f_G	Gravitational force
$F_{i,engine}^W$	Engine force along axis i
ρ	Air density
Q	Dynamic pressure
s	Reference surface area
l	Reference length
C_{Li}	Partial derivative of the roll moment coefficient w.r.t variable i
C_{Mi}	Partial derivative of the pitch moment coefficient w.r.t variable i
C_{Ni}	Partial derivative of the yaw moment coefficient w.r.t variable i
$C_{\mathcal{D}i}$	Partial derivative of the longitudal drag force coefficient w.r.t variable i
$C_{\mathcal{C}i}$	Partial derivative of the lateral drag force coefficient w.r.t variable i
$C_{\mathcal{L}i}$	Partial derivative of the lift force coefficient w.r.t variable i
δ_i	Control vector for actuator i
δ_R	Roll control vector
δ_P	Pitch control vector
δ_Y	Yaw control vector
x_{long}	Longitudal state vector
u_{long}	Longitudal input vector
x_{lat}	Lateral state vector
u_{lat}	Lateral input vector
δ	Actuator control vector
δ_{cmd}	Actuator commanded control vector
ζ	Actuator damping ratio
ω_0	Actuator natural frequency
d_1	Actuator position
d_2	Actuator velocity

Appendix C

CVXGEN Code

MPCA - x configuration

```
dimensions
        m = 3;  n = 8;  T = 4;
end

parameters
  A (n,n);  B (n,4);  C (m,n);  D (1,4);
  deltamax positive;  deltamin negative;
  alpha positive;
  delta[0] (n);
  u_des[t] (m), t=0..T+1
end

variables
  delta_cmd[t] (4), t=0..T
  delta[t] (n), t=1..T+1
  y[t] (m), t=0..T+1
end

minimize
  sum[t=1..T+1](square(y[t][1] - u_des[t][1])+square(y[t][2]
   - u_des[t][2])+square(y[t][3] - u_des[t][3]))
   + alpha*sum[t=0..T](square(D[1]*delta_cmd[t][1])
   + square(D[2]*delta_cmd[t][2])+square(D[3]*delta_cmd[t][3])
   + square(D[4]*delta_cmd[t][4]))
subject to
  delta[t+1] == A*delta[t] + B*delta_cmd[t], t=0..T
  y[t] == C*delta[t], t=0..T+1
  deltamin[1] <= delta[t][1] <= deltamax[1], t=1..T+1
  deltamin[2] <= delta[t][3] <= deltamax[2], t=1..T+1
  deltamin[3] <= delta[t][5] <= deltamax[3], t=1..T+1
  deltamin[4] <= delta[t][7] <= deltamax[4], t=1..T+1
```

LP CA - x configuration

```
dimensions
  m = 3;  n = 4;
end

parameters
  B(m,n);
  D(1,n);
  deltamax(1) positive; deltamin(1) negative;
  u(m);
  alpha positive;
end

variables
  delta(n);
end

minimize
  norm1(B*delta - u) + alpha*(norm1(D[1]*delta[1])
  + norm1(D[2]*delta[2]) + norm1(D[3]*delta[3])
  + norm1(D[4]*delta[4]))
subject to
  -delta >= -deltamax;
  delta >= deltamin;
end
```

MPCA - New configuration

```
dimensions
  m = 3;  n = 12;  T = 4;
end

parameters
  A (n,n);  B (n,6);  C (m,n);  D (1,6);
  deltamax (6) positive;
  deltamin (6) negative;
  alpha positive;
  beta positive;
  delta[0] (n);
  u_des[t] (m), t=0..T+1
end

variables
  delta_cmd[t] (6), t=0..T
  delta[t] (n), t=1..T+1
  y[t] (m), t=0..T+1
end
minimize
  sum[t=1..T+1](square(y[t][1]-u_des[t][1])
  +square(y[t][2]-u_des[t][2])+square(y[t][3]-u_des[t][3]))
  +alpha*sum[t=0..T](square(D[1]*delta_cmd[t][1])
  +square(D[2]*delta_cmd[t][2])+square(D[3]*delta_cmd[t][3])
  +square(D[4]*delta_cmd[t][4])+square(D[5]*delta_cmd[t][5])
  +square(D[6]*delta_cmd[t][6])  )
subject to
  delta[t+1] == A*delta[t] + B*delta_cmd[t], t=0..T
  y[t] == C*delta[t], t=0..T+1
  deltamin[1] <= delta[t][1] <= deltamax[1], t=1..T+1
  deltamin[2] <= delta[t][3] <= deltamax[2], t=1..T+1
  deltamin[3] <= delta[t][5] <= deltamax[3], t=1..T+1
  deltamin[4] <= delta[t][7] <= deltamax[4], t=1..T+1
  deltamin[5] <= delta[t][9] <= deltamax[5], t=1..T+1
  deltamin[6] <= delta[t][11] <= deltamax[6], t=1..T+1
end
```

LP CA - New configuration

```
parameters
  B(3,6);  D(1,6);
  deltamax(1) positive;  deltamin(1) negative;
  u(3)
  alpha positive
end

variables
  delta(6)
end

minimize
  norm1(B*delta - u) + alpha*norm1(D*delta)
subject to
  -delta >= -deltamax;
  delta >= deltamin;
end
```

Appendix D

RP CA Embedded Matlab Code

RP CA - x configuration

```
function delta = redist_pseudoinv(u)

     B = [ 0.3000   -0.3000    0.4000   -0.4000 ;
          0.8000    0.8000    0.6000    0.6000 ;
         -0.6000    0.6000    0.5000   -0.5000 ];
     W = diag([1 1 1 1]);
     c = zeros(4,1);
     pseudoctrl = mypseudo(B,B,W,c,u);
     [sat,c,modmat] = satcheck(pseudoctrl,c);
     if sat
         while sat
             pseudoctrl = mypseudo(B*modmat,B,W,-c,u);
             [sat,c,modmat] = satcheck(pseudoctrl,c);
             if sat==2
                 pseudoctrl = c;
                 sat=0;
             end
         end
     end
     delta = pseudoctrl;
end

function psctrl = mypseudo(B,Bc,W,c,ref)
    psctrl = -c + pinv(B)*(ref+Bc*c);
end

function [sat,new_c,modmat] = satcheck(vec,c)
    new_c = zeros(size(c));
    maxVal=20*pi/180;
    sat=0;
    for i=1:length(vec)
```

```
        if c(i)==0
            if abs(vec(i))>=maxVal
                sat=1;
                new_c(i) = sign(vec(i))*maxVal;
            else
                new_c(i) = c(i);
            end
        else
            new_c(i) = c(i);
        end
    end
    if abs(new_c(1))==maxVal && abs(new_c(2))==maxVal ...
              && abs(new_c(3))==maxVal && abs(new_c(4))==maxVal
        sat=2;
        modmat=eye(4);
    else
        modmat = diag(double((abs(new_c)~=maxVal)));
    end
end
```

RP CA - New configuration

```
function delta = redist_pseudoinv(u)

     B = [ 0.3000   -0.3000    0.4000   -0.4000 0.9 -0.9;
          0.8000    0.8000    0.6000    0.6000 0.7 0.7;
         -0.6000    0.6000    0.5000   -0.5000 -0.3 0.3];
     W = diag([1 1 1 1 0.1 0.1]);
     c = zeros(6,1);
     pseudoctrl = mypseudo(B,B,W,c,u);
     [sat,c,modmat] = satcheck(pseudoctrl,c);
     if sat
         while sat
             pseudoctrl = mypseudo(B*modmat,B,W,-c,u);
             [sat,c,modmat] = satcheck(pseudoctrl,c);
             if sat==2
                 pseudoctrl = c;
                 sat=0;
             end
         end
     end
     delta = pseudoctrl;
end
```

```
function psctrl = mypseudo(B,Bc,W,c,ref)
    psctrl = -c + W\B'*pinv(B*inv(W)*B')*(ref+Bc*c);
end

function [sat,new_c,modmat] = satcheck(vec,c)
    new_c = zeros(size(c));
    maxVal=[ 20*pi/180 20*pi/180 20*pi/180 ...
             20*pi/180 10*pi/180 10*pi/180 ];
    sat=0;
    for i=1:length(vec)
        if c(i)==0
            if abs(vec(i))>=maxVal(i)
                sat=1;
                new_c(i) = sign(vec(i))*maxVal(i);
            else
                new_c(i) = c(i);
            end
        else
            new_c(i) = c(i);
        end
    end
    if abs(new_c(1))==maxVal(1) && abs(new_c(2))==maxVal(2) ...
        && abs(new_c(3))==maxVal(3) && abs(new_c(4))==maxVal(4) ...
        && abs(new_c(5))==maxVal(5) && abs(new_c(6))==maxVal(6)
        sat=2;
        modmat=eye(6);
    else
        new_c_p1 = new_c(1:4);
        new_c_p2 = new_c(5:6);
        modmat = diag( [ double((abs(new_c_p1')~=maxVal(2))) ...
        double((abs(new_c_p2')~=maxVal(6))) ] );
    end
end
```

Appendix E

Conference paper

This appendix contains the article "Dynamic Model Predictive Control Allocation using CVXGEN", written by Martin Hanger, Tor A. Johansen, Geir Kåre Mykland and Aage Skullestad. It includes the main ideas and results of this thesis, namely the development of the MPCA formulation, and the use of CVXGEN to solve the optimization. The paper is aimed for submission at the Ninth IEEE International Conference on Control and Automation in Santiago, Chile.

The article starts on the next page.

Dynamic Model Predictive Control Allocation using CVXGEN

Martin Hanger, Tor A. Johansen, Geir Kåre Mykland and Aage Skullestad

***Abstract*— Control allocation deals with the allocation of control among a redundant set of effectors, while taking into account the individual constraints. The use of model predictive control (MPC) for control allocation allows the response times of the actuators to be accounted for, and in particular to take advantage of predictions of the virtual control input as well as differences in dynamic control authority and cost of use among the actuators. The use of online quadratic programming (QP) is essential for implementation of the optimal constrained control allocation strategies. The main contributions of the present paper are the investigation of using the software system CVX-GEN and the MPC-based control allocation method. CVXGEN synthesizes a customized portable and library-free C-source code QP solver for the specific QP problem resulting from the MPC formulation, exploiting structural properties of the QP and optimizing the source code for execution speed. Two case studies, one being a missile auto-pilot, illustrates the benefits of using the MPC formulation, and the efficiency of CVXGEN.**

I. INTRODUCTION

Some control systems are designed with redundant actuator and effectors, for reasons such as fault tolerance and design issues related to cost, response-time, size, and flexibility. Examples include flight control systems [2], dynamic positioning systems for ships with using thrusters [8], and airjet controlled paper motion in machines [4].

Control algorithm design for systems with input redundancy is challenging since the same control effect (like a generalized force) can be generated by a number of different actuator settings, and actuator constraints should be accounted for. In order to systematically manage such control design challenges, one may decompose the control problem into two parts - a controller that commands a virtual control input of minimal dimension (like the generalized force), and a control allocation module that maps the virtual control input into the redundant actuator settings. Since there are more degrees of freedom available in the actuator system than virtual control variables, the available degrees of freedom in the actuator system can be used to satisfy actuator constraints and to meet secondary objectives such as fault tolerance, power consumption minimization, and actuator wear minimization. In general, the control allocation problem can be formulated as an optimization problem where certain objectives are minimized subject to actuator and effector constraints, and the constraint that the resulting control effect fulfills the requirements of the virtual control command. The main difference between different control allocation methods are related to how the optimization problem is formulated, which models are used, and which numerical algorithms is employed to solve it. This is reviewed in the next paragraphs.

M. Hanger and T. A. Johansen are with Department of Engineering Cyberentics, Norwegian University of Science and Technology, Trondheim, Norway. tor.arne.johansen@itk.ntnu.no

G. K. Mykland and A. Skullestad are with Kongsberg Defence Systems, Kongsberg, Norway

In the classic formulations of the constrained control allocation problem the actuator dynamics are neglected [2], under the assumption that all dynamic phenomena are accounted for by the controller that commands the virtual control to the control allocation module. This may in some cases be an unrealistic and inconvenient assumption when the actuator dynamics are limiting the control performance since response times and different dynamic authorities of the actuators are not taken into account. For systems where actuator dynamics are known, the interactions between the control allocation algorithm and the actuator dynamics working on the aircraft body become more complex, requiring a more sophisticated control allocation method. Actuators can have different response times, i.e. a fast actuator can be used to achieve fast transient response, while slow actuators can be used for steady state or trimmed flight, to improve power efficiency. A Model Predictive Control (MPC) allocation scheme will be able to optimally exploit such properties.

It is relatively straightforward to (re-)design a basic control allocation algorithm to comply with actuator rate constraints, e.g. [8], by incorporating this as a constraint on the change in control inputs from the previous sample to the current sample. More sophisticated dynamic actuator models may be incorporated by using the powerful MPC framework to solve the constrained control allocation problem [9], [10], [14], [20]. MPC is an optimization-based control algorithm which can be used in control allocation, beeing able to handle actuator dynamics as well as actuator saturation. MPC utilizes a model of the plant in predicting outputs and states, where in control allocation this model describes the actuator dynamics. Because of the predictive nature of the controller, the calculated control can pre-act to the actuator system dynamics to improve performance.

How to implement the numerical optimization for the optimal control allocaiton in real time, is a challenging task. Online optimization using off-the-shelf or customized quadratic programming (QP) solvers are studied in the context of linear actuator and effector models in [1],[15],[3], [16]. For nonlinear effector models, the use of sequential quadratic programming is proposed in [5]. Instead of demanding that the optimal control allocation is computed exactly at each sample, the dynamic online optimization appraoch in [6] will at each time instant move in the direction towards an optimal control allocation, but optimality is achieved only asymptotically. The method is extended to the case with actuator dynamics in [18]. While the dynamic online optimization approach reduces the online computational requirements, and at the same time guarantees that closed loop stability is not lost due to sub-optimality, one may also use multi-parametric programming to pre-compute an explicitly represented piecewise affine solution function. The remaining online computations corresponds to the evaluation of a piecewise linear function resulting from multi-parametric programming and explicit MPC [7],[19]. While this is highly attractive from the online processing point of view, its memory consumption and offline processing does not scale very well - in particular when considering control efficiency matrices that are time- or state-dependent due to nonlinear to time-varying characteristics like in fault tolerant control allocation [17].

This key idea of the present paper is to employ a family of highly customized QP solvers that are automatically generated using CVXGEN [13],[11] to solve MPC-based dynamic control allocation problems. CVXGEN has the unique feature that the C code of the customized solvers is completely standard and standalone, i.e. portable, and extremely efficient since the key structural properties of the QP problem is exploited in the automatic code generation that leads to code with only static data structures and almost branch-free code where for-loops are rolled out for efficiency and deterministic execution on pipeline processor architectures. Performance improvement also comes for low software overhead as the CVXGEN targets small-scale problems, in some contrast to most off-the-shelf solvers that target large-scale problems. Orders of magnitude faster execution compared to state-of-the-art off-the-shelf solvers have been reported on test problems, including MPC problems [13],[11]. This makes it interesting to study CVXGEN's performance in challenging control allocation problems that are of relatively small scale compared to typical MPC problems.

The paper is organized as follows. First the dynamic MPC-based control allocation problem formulation is introduced. Then the use of CVXGEN to address this problem is described, before the computational performance is assessed in a simulation benchmark study.

II. DYNAMIC CONTROL ALLOCATION

A. Optimization problem formulation

It is assumed that all control actuators have dynamics which can be approximately modelled as second order systems,

$$\ddot{\delta} - 2\zeta\omega_0\dot{\delta} - \omega_0^2\delta = \omega_0^2\delta_{cmd} \tag{1}$$

where δ_{cmd} is the commanded control input, and δ is the actuator response. ζ and ω_0 are the actuators relative damping ratio and natural frequency, respectively. Rewritten in state-space form, the model for actuator i will be on the form

$$\dot{\boldsymbol{\delta}}_i = A_{\delta_i}\boldsymbol{\delta}_i + B_{\delta_i}\boldsymbol{\delta}_{cmd,i} \tag{2}$$

For a system with K actuators and effectors, the model will be

$$\begin{bmatrix}\dot{\boldsymbol{\delta}}_1\\ \vdots\\ \dot{\boldsymbol{\delta}}_K\end{bmatrix} = \begin{bmatrix}A_{\delta_1} & 0 & \cdots & 0\\ 0 & A_{\delta_2} & \cdots & 0\\ \vdots & & \ddots & \\ 0 & & \cdots & A_{\delta_K}\end{bmatrix}\begin{bmatrix}\boldsymbol{\delta}_1\\ \vdots\\ \boldsymbol{\delta}_K\end{bmatrix} + \begin{bmatrix}B_{\delta_1}\\ \vdots\\ B_{\delta_K}\end{bmatrix}\begin{bmatrix}\boldsymbol{\delta}_{cmd,1}\\ \vdots\\ \boldsymbol{\delta}_{cmd,K}\end{bmatrix} \tag{3}$$

In a more compact form, this can be written

$$\dot{\boldsymbol{\delta}} = A_\delta\boldsymbol{\delta} + B_\delta\boldsymbol{\delta}_{cmd} \tag{4}$$

The corresponding MPC control allocation problem is posed as follows: For the constrained system

$$\begin{aligned}\dot{\boldsymbol{\delta}}(t) &= A_\delta\boldsymbol{\delta}(t) + B_\delta\boldsymbol{\delta}_{cmd}(t)\\ \boldsymbol{\tau}(t) &= B\boldsymbol{\delta}(t)\\ \delta_{min} \leq &\ \boldsymbol{\delta} \leq \delta_{max}\end{aligned} \tag{5}$$

find $\boldsymbol{\delta}_{cmd}(t)$ such that $\boldsymbol{\tau}(t)$ tracks $\boldsymbol{\tau}^*(t)$ as closely as possible, where $\boldsymbol{\tau}^*(t)$ is the virtual control input vector, B is the control efficiency matrix and δ_{min}, δ_{max} are the upper and lower saturation limits of the effectors or actuators, respectively.

The system (5) is used to predict the commanded control inputs $\boldsymbol{\delta}_{cmd}$, the control commands $\boldsymbol{\delta}$ and outputs y throughout the prediction horizon,

$$\hat{\boldsymbol{\delta}}_{cmd} = [\ \hat{\boldsymbol{\delta}}_{cmd}(k|k),\ \cdots,\ \hat{\boldsymbol{\delta}}_{cmd}(k+N-1|k)\ ] \tag{6}$$

$$\hat{\boldsymbol{\delta}} = [\ \hat{\boldsymbol{\delta}}(k+1|k),\ \cdots,\ \hat{\boldsymbol{\delta}}(k+N|k)\ ] \tag{7}$$

$$\hat{\boldsymbol{\tau}} = [\ \hat{\boldsymbol{\tau}}(k+1|k),\ \cdots,\ \hat{\boldsymbol{\tau}}(k+N|k)\ ] \tag{8}$$

where N is the length of the prediction horizon, and k is the current time step. The MPC algorithm finds the optimal set of $\hat{\boldsymbol{\delta}}_{cmd}$ by minimizing a cost function on the form

$$\begin{aligned}J(\cdot) = &\sum_{j=1}^{N} W(j)[\ \hat{\boldsymbol{\tau}}(k+j\ |k) - \boldsymbol{\tau}^*(k+j)\ ]^2\\ &+\alpha\sum_{j=1}^{N-1}\sum_{i=1}^{K} W_a(i)[\ \boldsymbol{\delta}_{cmd,i}(k+j-1|k)\ ]^2\end{aligned} \tag{9}$$

subject to (5).

In the cost function, W is a weight matrix weighing the importance of tracking $\boldsymbol{\tau}^*$ at time j. W_a weighs the relative cost of use of effector $i \in \{1\ldots K\}$. As before, K is the number of control actuators. $\alpha > 0$ weighs the relative importance between the tracking term and the effector penalty term, and is usually small. Only the first commanded control sample $\hat{\boldsymbol{\delta}}_{cmd}(k|k)$ is applied to the actuator. The whole algorithm is repeated when computing the consequtive $\hat{\boldsymbol{\delta}}_{cmd}(k+1|k+1)$.

B. CVXGEN Solver

The CVXGEN solver is currently available through a web interface `http://www.cvxgen.com`. An optimization problem specification can be entered through a MATLAB-like programming language. Syntax specifics can be found in CVXGEN's user manual [12]. The problem is entered in a fixed problem structure, specifying the problem's dimensions, parameters, variables, cost function and constraints.

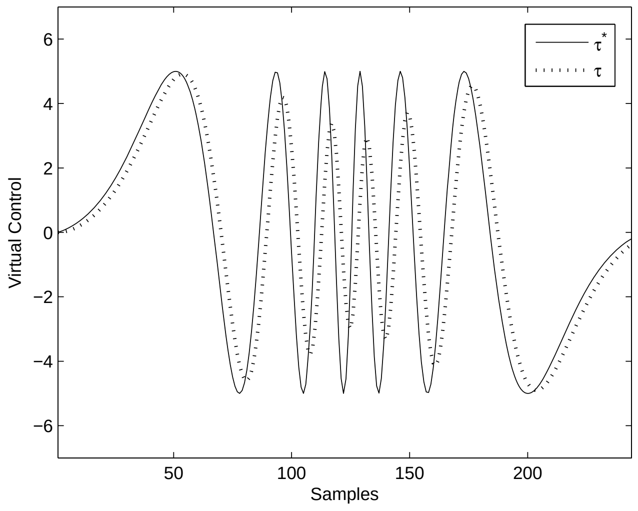

Fig. 1. QP CA Virtual Input Tracking

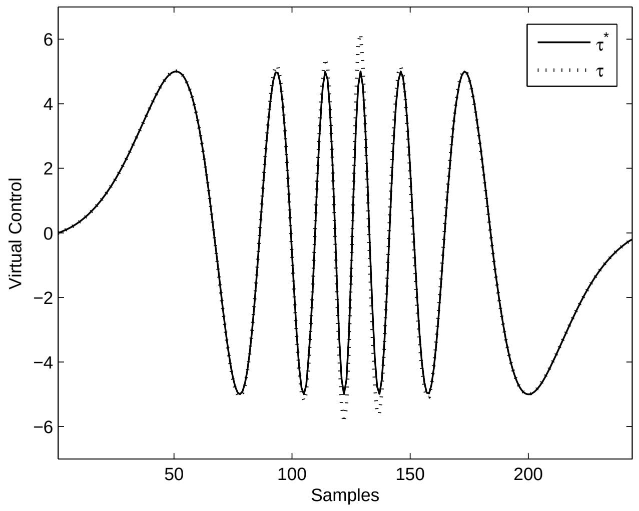

Fig. 2. MPCA Virtual Input Tracking

The library-free custom C solver is automatically generated. In addition to C code, a MATLAB interface is also available, making the custom solver available for e.g. prototyping and initial testing within the MATLAB environment.

The solver is used by calling a pre-made function, with the problem instance's specific parameters as function input. Solver settings can also be entered when calling the solver. After the call, the solver solves the convex optimization problem with respect to the instance parameters, and outputs the globally optimal solution.

CVXGEN lends itself naturally to MPC problems, see [11] for a detailed overview.

III. CASE STUDIES

The examples will illustrate performance tradeoffs between control performance, accuracy and cost of actuation (power, wear,...) that can be systematically adressed with dynamic predictive control allocation. Furthermore, computational performance characteristics of the CVXGEN implementation are reported.

A. Simple test - actuators/effectors with different cost and dynamic response

First, a simple test is conducted, comparing the performance of similar MPCA and QP formulations. The virtual control command τ^* is one-dimensional, consisting of a sine with increasing and then decreasing frequency. There are two actuators δ_1 and δ_2, with associated effectors, both modeled as second order systems. Actuator 1 will be fast but expensive to use, while actuator 2 will be slow and inexpensive. The actuator coefficients and corresponding cost weight are

$$\omega_{0,1} = 150, \quad \zeta_1 = 0.7, \quad W_1 = 1$$
$$\omega_{0,2} = 10, \quad \zeta_2 = 0.9, \quad W_2 = 0.1$$

This means that the control allocation module should use actuator/effector 1 only when necessary. In addition, effector 2 will be more efficient than effector 1, reflected in the control efficiency matrix $B = [\ 0.3\ \ 0.8\ ]$. The virtual input prediction for the MPCA is done using a second order extrapolation based on the current and most recent samples.

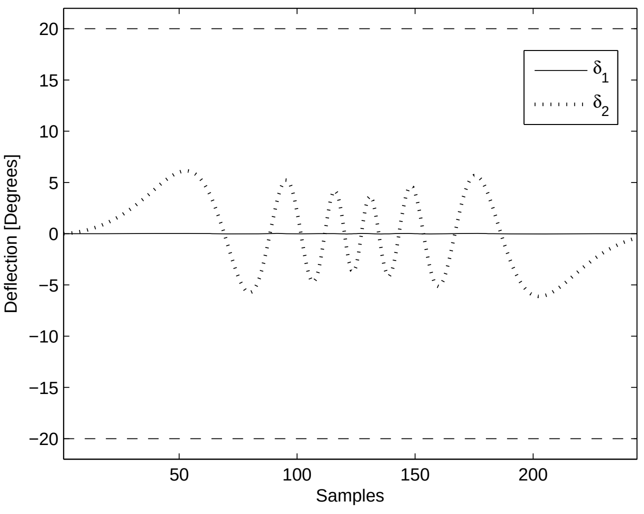

Fig. 3. QP CA Actuator Response

The virtual input tracking of the QP and MPCA methods can be seen in Figures 1 and 2 respectively. It is clear that the MPCA does a far better job than the similar QP formulation when it comes to tracking τ^*. This is because the QP CA ignores the actuator dynamics, leading to it commanding mostly the slow actuator δ_2 to deflect to track τ^*. As the frequency of the virtual input increases, actuator 2 can not follow, causing a larger tracking error. MPCA is aware of the actuator dynamics and optimally combines both actuators to meet the requirement of the virtual input. The actuator response δ_1 and δ_2 for the QP and MPCA methods can be seen in Figures 3 and 4, respectively. In these plots the actuator saturation limits are shown as dashed lines.

A comparison of the cost is shown in Figure 5, which summarizes the two methods' performance.

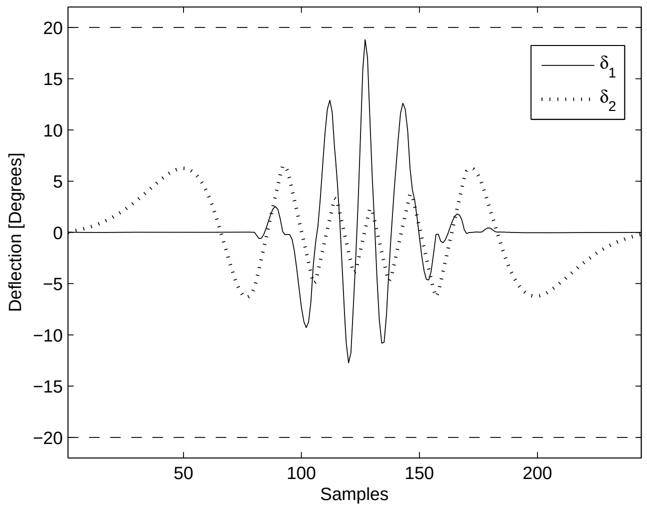

Fig. 4. MPCA Actuator Response

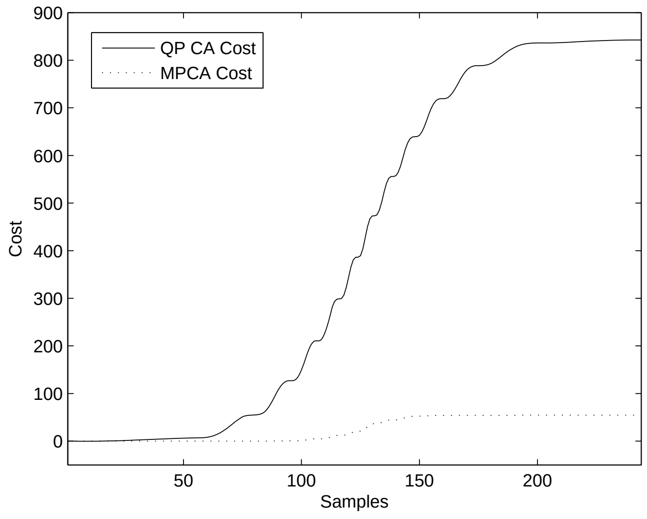

Fig. 5. Cumulative cost for QP CA and MPCA.

B. Missile auto-pilot

MPCA is also tested in a more realistic setup, as a part of a missile flight control system. The missile dynamics are approximated using decoupled longitudal and lateral models [22]. Such models are valid for small angles, but this is assumed to be sufficient for testing control allocation.

The models are on the form

$$\dot{x}_{long} = A_{long}x_{long} + B_{long}u_{long} \tag{10}$$
$$\dot{x}_{lat} = A_{lat}x_{lat} + B_{lat}u_{lat} \tag{11}$$

where

$$x_{long} = [\ \alpha\ q\ \theta\ ]^T, \quad u_{long} = \delta_P$$
$$x_{lat} = [\ \beta\ p\ r\ \phi\ \psi\ ]^T, \quad u_{lat} = [\delta_R\ \delta_Y\ ]^T$$

Subscripts denote longitudal and lateral models, and symbols are summarized in Table I.

The simulated missile has a mass of 200kg, flying at

p	Roll Rate	α	Angle of Attack
q	Pitch Rate	β	Sideslip
r	Yaw Rate	δ_R	Roll Control Moment
θ	Pitch Angle	δ_P	Pitch Control Moment
ϕ	Roll Angle	δ_Y	Yaw Control Moment
ψ	Yaw Angle		

TABLE I
SYMBOLS IN MISSILE MODEL

constant speed 300m/s, and has an inertia matrix

$$I = \begin{bmatrix} 5 & 1 & 1 \\ 0 & 150 & 0 \\ 0 & 0 & 150 \end{bmatrix} kgm^2$$

A new effector configuration is developed, tailored to be used with MPCA. It combines tail control, four fins are placed in an x-configuration, and wing control, where each wing has an aileron. All actuators $\delta_{1...6}$, are modeled as second order systems (1). Subscrips 1-4 denote the tail fins, while subscrips 5-6 represent the two wing ailerons. The actuator characteristics and cost are summarized below.

$$\omega_{0,1} = 150, \quad \zeta_1 = 0.7, \quad W_1 = 1$$
$$\omega_{0,2} = 150, \quad \zeta_2 = 0.7, \quad W_2 = 1$$
$$\omega_{0,3} = 150, \quad \zeta_3 = 0.7, \quad W_3 = 1$$
$$\omega_{0,4} = 150, \quad \zeta_4 = 0.7, \quad W_4 = 1$$
$$\omega_{0,5} = 10, \quad \zeta_5 = 0.9, \quad W_5 = 0.1$$
$$\omega_{0,6} = 10, \quad \zeta_6 = 0.9, \quad W_6 = 0.1$$

The new configuration has two main actuator groups spanning different dynamic authorities. The slow and inexpensive wing ailerons are thought to be used while in trimmed flight, while the fast, expensive tail fins will be added on in agile flight.

The control allocation is part of a flight control system together with a bank-to-turn autopilot, designed to follow lateral and longitudal references. The autopilot design has two loops. The outer loop is controlling z and y position, while beeing fed back missile lateral and longitudal accelerations. This loop uses a bank-to-turn design to command the inner angular control loop. All controllers within these loops are PI- or P-controllers. The autopilot's virtual control output $\tau^* = [\ \delta_R^*\ \ \delta_P^*\ \ \delta_Y^*\ ]^T$ is the input to the control allocation module, which computes a commanded control $\delta_{cmd,i},\ i \in \{1,2,3,4,5,6\}$, which is applied to the actuators. The actual actuator response $\delta_i,\ i \in \{1,2,3,4,5,6\}$ is mapped with the control efficiency matrix B to form the control vector $\tau = [\ \delta_R\ \delta_P\ \delta_Y\ ]^T$. This vector is in turn input to the missile model.

$$B = \begin{bmatrix} 0.3 & -0.3 & 0.4 & -0.4 & 0.8 & -0.8 \\ 0.8 & 0.8 & 0.6 & 0.6 & 0.7 & 0.7 \\ -0.6 & 0.6 & 0.5 & -0.5 & -0.3 & 0.3 \end{bmatrix}$$

An MPCA formulation like the one described in II-A is used, and a QP control allocation problem is used as a comparison. The prediction for the MPCA is created by

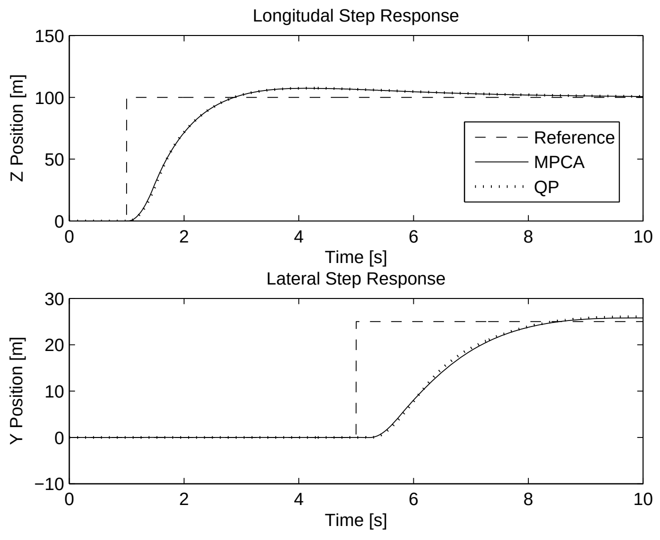

Fig. 6. Missile Step Response

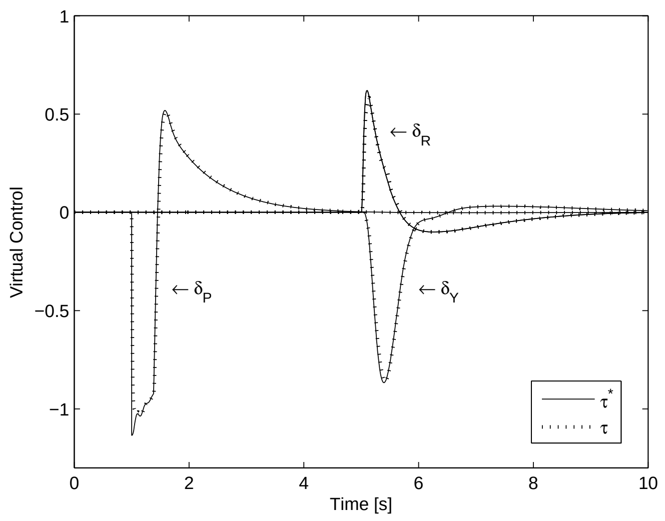

Fig. 7. MPCA Virtual Control Tracking

holding the current value throughout the prediction horizon N, which spans five samples. The MPCA is designed with sample time $0.05s$, making the horizon $0.2s$ long.

The lateral and longitudal trajectory references are steps, and the missile step responses are seen in Figure 6. The responses for MPCA and QP CA are relatively similar, though the MPCA performs somewhat better in the longitudal step case.

Looking at the virtual control tracking, the MPCA and QP CA cases are shown in Figures 7 and 8, respectively. It is clear that the MPCA, being aware of the actuator dynamics, provides significantly better tracking of τ^* than the QP CA case. Also note that the choice of control allocation method affects how the virtual control looks. The QP CA induces undesirable oscillations and larger amplitudes in the virtual control. Because of this the choice of control allocation method should not be arbitrary, but a consideration of method complexity, present constraints, actuator dynamics

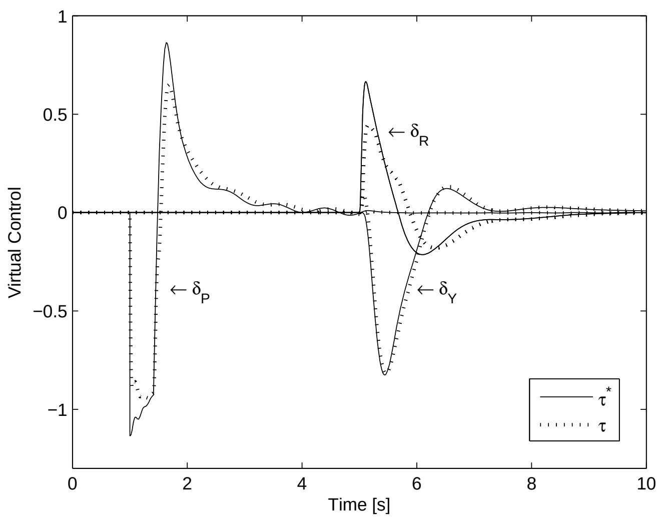

Fig. 8. QP Virtual Control Tracking

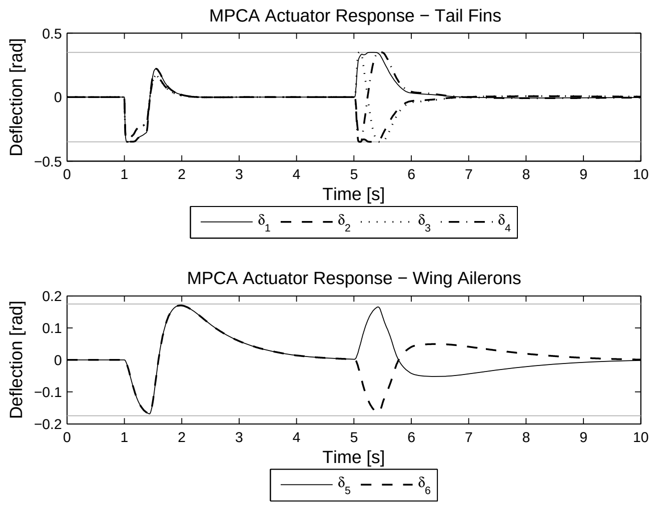

Fig. 9. MPCA Actuator Response

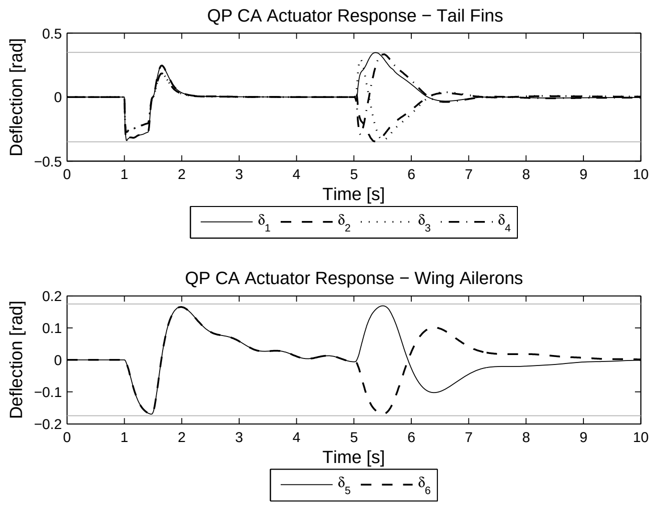

Fig. 10. QP Actuator Response

Parameter entries	155
Original variables	78
Variables in solver	114
Equalities in solver	94
Inequalities in solver	80

TABLE II

MPCA OPTIMIZATION PROBLEM SIZE STATISTICS FOR MISSILE EXAMPLE

and configuration. The actuator responses for MPCA and QP are shown in Figures 9 and 10. Actuator limits are shown as grey lines. Both methods use the wing ailerons δ_5 and δ_6 actively, which is a good thing since these are inexpensive. The mentioned oscillations also show up in the actuator response in the QP case. These cause the QP method to allocate the tail fins more, which the MPCA can avoid. Beeing aware of the actuator dynamics, the MPCA can exploit its knowledge to allocate actuators more efficiently, leading to the previously mentioned improved virtual control tracking.

Lastly, the cost is compared. The cumulative cost is shown in Figure 11. As expected, the MPCA cost is well below that of QP CA, mainly because of the latter methods excessive actuator use and delay in virtual control tracking.

CVXGEN is used during this simulation, and it is interesting to review the time consumption of the solver. During the 10-second simulation, 1045 calls are made to the model predictive control allocation function calculating the commanded control input δ_{cmd}. By isolating MATLAB on one CPU and using the program's *profiler* utility, it is found that these calls took a total of 0.434 seconds (CPU time), making each call on average consume 0.00041531 seconds CPU time. This is considered to be very fast, taking the large problem size into account. The MPCA problem size statistics are summarized in Table II. The solve time also scales well when using CVXGEN. A nearly linear relationship between solve time and complexity was found in an experiment where MPCA horizon was incrementally increased from $N = 1$ to $N = 7$, see Figure 12. In the step from $N = 6$ to $N = 7$, the complexity exceeds the recommended limit of CVXGEN (4000 nonzero KKT entries), and the solve time increases nonlinearily.

IV. CONCLUSIONS

It is shown that the dynamic constrained allocation problem for typical configurations can be solved efficiently using MPC and CVXGEN.

The MPC formulation leads to improved overall control performance compared to a more conventional static control allocation method, and it is able to exploit an actuator configuration with different dynamic properties. The developed MPCA provides better virtual control tracking aswell as allocating actuators more efficiently than classical formulations.

The use of CVXGEN leads to a customized quadratic programming solver that typically require less than 1 millisecond computation time on a powerful processor. This may

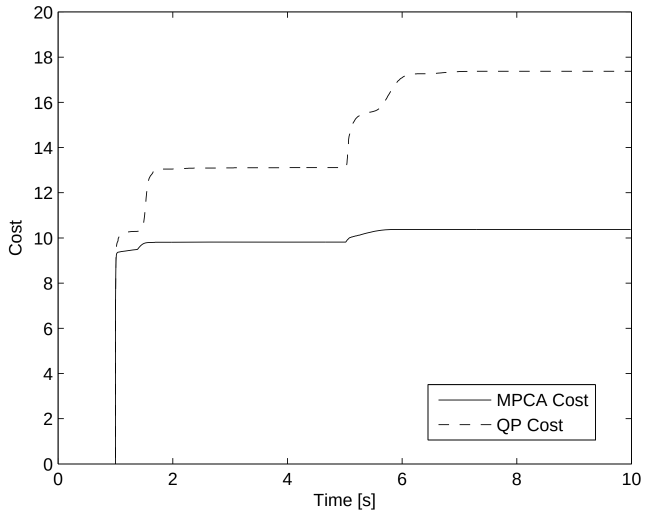

Fig. 11. Cumulative Cost, Step Response

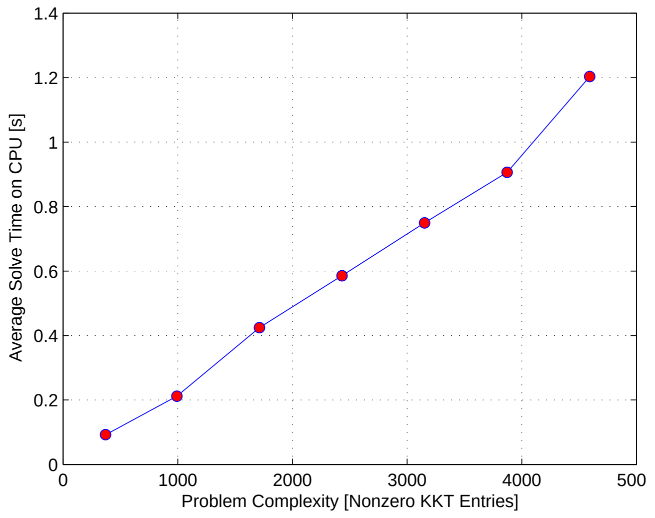

Fig. 12. CVXGEN Solve Time Scaling

considered computationally feasible for implementation in a flight control system, although important aspects such as software code verifiability needs to be addressed carefully.

REFERENCES

[1] M. Bodson, Evaluation of optimization methods for control allocation, *J. Guidance Control and Dynamics*, Vol. 25, 2002

[2] W. C. Durham, Constrained control allocation, *J. Guidance, Control, and Dynamics*, Vol. 16, pp. 717-725, 1993

[3] O. Härkegård, Dynamic control allocation using constrained quadratic programming *Journal of Guidance, Control, and Dynamics*, Vol. 27, pp. 1028-1034, 2004.

[4] W. Jackson, M. P. J. Fromherz, D. K. Biegelsen, J. Reisch, and D. Goldberg, Constrained optimization based control of real time largescale systems: Airjet object movement system, in Proc. IEEE Conf. Decision and Control, Orlando, 2001.

[5] T. A. Johansen, T. I. Fossen, Svein P. Berge, Constrained Nonlinear Control Allocation with Singularity Avoidance using Sequential Quadratic Programming, *IEEE Trans. Control Systems Technology*, Vol. 12, pp. 211-216, 2004

[6] T. A. Johansen, Optimizing nonlinear control allocation, IEEE Conf. Decision and Control, Nassau, Bahamas, pp. 3435-3440, 2004

[7] T. A. Johansen, T. I. Fossen, P. Tøndel, Efficient Optimal Constrained Control Allocation via Multi-Parametric Programming, *J. Guidance, Control and Dynamics*, Vol. 28, pp. 506-515, 2005

[8] T. A. Johansen, T. P. Fuglseth, P. Tøndel, T. I. Fossen, Optimal constrained control allocation in marine surface vessels with rudders, *Control Engineering Practise*, Vol. 16, pp. 457-464, 2008

[9] Y. Luo, A. Serrani, S. Yurkovich, D. B. Doman, M. W. Oppenheimer, Model predictive dynamic control allocation with actuator dynamics, American Control Conference, Boston, pp. 1695-1700, 2004

[10] Y. Luo, A. Serrani, S. Yurkovich, D. B. Doman, M. W. Oppenheimer, Dynamic control allocation with asymptotic tracking of time-varying control input commands, American Control Conference, Portland, pp. 2098-2103, 2005

[11] J. Mattingley, Y. Wang and S. Boyd, Code Generation for Receding Horizon Control, Proceedings IEEE Multi-Conference on Systems and Control, pages 985992, Yokohama, Japan, September 2010

[12] J. Mattingley and S. Boyd, CVXGEN: A Code Generator for Embedded Convex Optimization, CVXGEN Website http://www.cvxgen.com, 2010

[13] J. Mattingley and S. Boyd. Automatic Code Generation for Real-Time Convex Optimization, chapter in Convex Optimization in Signal Processing and Communications, Y. Eldar and D. Palomar, Eds., Cambridge University Press, 2009

[14] M. W. Oppenheimer and D. B. Doman, Control allocation for overactuated systems, Proc. Mediterranean Conf. Control and Automation, 2006

[15] J. A. M. Petersen and M. Bodson, Constrained Quadratic Programming tecniques for Control Allocation, Proc. IEEE Conf. Decision and Control, Maui, Hawaii, 2003

[16] E. Ruth and A. J. Sørensen, A solution to the Nonconvex Linearly Constrained Quadrating Thrust Allocation Problem. In Proceedings of 8th Conference on Manoeuvring and Control of Marine Craft (MCMC2009), pp. 195-200, Guaruj, Brazil, 2009

[17] J. Spjøtvold, T. A. Johansen, Fault Tolerant Control Allocation for a Thruster-Controlled Floating Platform using Parametric Programming, IEEE Conf. Decision and Control, Shanghai, 2009

[18] J. Tjønnås and T. A. Johansen, Optimizing Adaptive Control Allocation with Actuator Dynamics, Proc. IEEE Conf. Decision and Control, New Orleans, 2007

[19] P. Tøndel, and T. A. Johansen, Control allocation for Yaw Stabilization in Automotive Vehicles using Multiparametric Nonlinear Programming, American Control Conference, Portland, 2005

[20] C. Vermillion, J. Sun and K. Butts, Model predictive control allocation for overactuated systems - stability and performance, IEEE Conf. Decision and Control, New Orleans, pp. 1251 - 1256, 2007.

[21] K. A. Bordignon, Constrained Control Allocation for Systems with Redundant Control Effectors, Virginia Polytechnic Institute and State University, 1996.

[22] T. I. Fossen, Mathematical Models for Control of Aircraft and Satellites, Technical Report, Department of Engineering Cynernetics, Norwegian University of Science and Technology, 2011.